



ETHICAL HACKING AND PENETRATION TESTING GUIDE

RAFAY BALOCH



CRC Press
Taylor & Francis Group

AN AUERBACH BOOK

ETHICAL HACKING AND PENETRATION TESTING GUIDE

ETHICAL HACKING AND PENETRATION TESTING GUIDE

RAFAY BALOCH



CRC Press

Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
AN AUERBACH BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

© 2015 by Taylor & Francis Group, LLC
CRC Press is an imprint of Taylor & Francis Group, an Informa business

No claim to original U.S. Government works
Version Date: 20140320

International Standard Book Number-13: 978-1-4822-3162-5 (eBook - PDF)

This book contains information obtained from authentic and highly regarded sources. Reasonable efforts have been made to publish reliable data and information, but the author and publisher cannot assume responsibility for the validity of all materials or the consequences of their use. The authors and publishers have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission to publish in this form has not been obtained. If any copyright material has not been acknowledged please write and let us know so we may rectify in any future reprint.

Except as permitted under U.S. Copyright Law, no part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, please access www.copyright.com (<http://www.copyright.com/>) or contact the Copyright Clearance Center, Inc. (CCC), 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. CCC is a not-for-profit organization that provides licenses and registration for a variety of users. For organizations that have been granted a photocopy license by the CCC, a separate system of payment has been arranged.

Trademark Notice: Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

Visit the Taylor & Francis Web site at
<http://www.taylorandfrancis.com>

and the CRC Press Web site at
<http://www.crcpress.com>

Contents

Preface	xxiii
Acknowledgments	xxv
Author	xxvii
1 Introduction to Hacking	1
Important Terminologies	2
Asset	2
Vulnerability.....	3
Threat	3
Exploit.....	3
Risk	3
What Is a Penetration Test?	3
Vulnerability Assessments versus Penetration Test	3
Preengagement.....	3
Rules of Engagement	4
Milestones	4
Penetration Testing Methodologies.....	5
OSSTMM.....	5
NIST	6
OWASP	7
Categories of Penetration Test.....	7
Black Box.....	7
White Box	7
Gray Box	7
Types of Penetration Tests	7
Network Penetration Test.....	8
Web Application Penetration Test	8
Mobile Application Penetration Test	8
Social Engineering Penetration Test	8
Physical Penetration Test	8
Report Writing	8
Understanding the Audience	9

Executive Class.....	9
Management Class	9
Technical Class.....	9
Writing Reports	10
Structure of a Penetration Testing Report.....	10
Cover Page.....	10
Table of Contents	10
Executive Summary.....	11
Remediation Report	12
Vulnerability Assessment Summary.....	12
Tabular Summary.....	13
Risk Assessment.....	14
Risk Assessment Matrix.....	14
Methodology	14
Detailed Findings.....	15
Description.....	15
Explanation	16
Risk	16
Recommendation	16
Reports	17
Conclusion.....	17
2 Linux Basics	19
Major Linux Operating Systems	19
File Structure inside of Linux.....	20
File Permission in Linux	22
Group Permission.....	22
Linux Advance/Special Permission	22
Link Permission.....	23
Suid & Guid Permission.....	23
Stickybit Permission	23
Chatter Permission	24
Most Common and Important Commands.....	24
Linux Scheduler (Cron Job)	25
Cron Permission	26
Cron Permission	26
Cron Files.....	26
Users inside of Linux	28
Linux Services.....	29
Linux Password Storage.....	29
Linux Logging.....	30
Common Applications of Linux	30
What Is BackTrack?.....	30
How to Get BackTrack 5 Running.....	31
Installing BackTrack on Virtual Box	31
Installing BackTrack on a Portable USB	35

Installing BackTrack on Your Hard Drive	39
BackTrack Basics	43
Changing the Default Screen Resolution	43
Some Unforgettable Basics.....	44
Changing the Password	44
Clearing the Screen	44
Listing the Contents of a Directory	44
Displaying Contents of a Specific Directory	44
Displaying the Contents of a File.....	45
Creating a Directory.....	45
Changing the Directories	45
Windows	45
Linux.....	45
Creating a Text File	45
Copying a File	45
Current Working Directory.....	45
Renaming a File	45
Moving a File	46
Removing a File.....	46
Locating Certain Files inside BackTrack.....	46
Text Editors inside BackTrack.....	46
Getting to Know Your Network	47
Dhclient.....	47
Services.....	48
MySQL.....	48
SSHD	48
Postgresql.....	50
Other Online Resources	51
3 Information Gathering Techniques.....	53
Active Information Gathering.....	53
Passive Information Gathering.....	53
Sources of Information Gathering	54
Copying Websites Locally.....	54
Information Gathering with Whois.....	55
Finding Other Websites Hosted on the Same Server.....	56
Yougetsignal.com	56
Tracing the Location	57
Traceroute.....	57
ICMP Traceroute.....	58
TCP Traceroute	58
Usage.....	58
UDP Traceroute	58
Usage.....	58
NeoTrace	59
Cheops-ng.....	59
Enumerating and Fingerprinting the Webservers.....	60

Intercepting a Response	60
Acunetix Vulnerability Scanner	62
WhatWeb	62
Netcraft	63
Google Hacking	63
Some Basic Parameters.....	64
Site.....	64
Example.....	64
TIP regarding Filetype.....	65
Google Hacking Database	66
Hackersforcharity.org/ghdb.....	67
Xcode Exploit Scanner.....	67
File Analysis.....	68
Foca.....	68
Harvesting E-Mail Lists	69
Gathering Wordlist from a Target Website	71
Scanning for Subdomains.....	71
TheHarvester	72
Fierce in BackTrack	72
Scanning for SSL Version	74
DNS Enumeration.....	75
Interacting with DNS Servers	75
Nslookup	76
DIG	76
Forward DNS Lookup.....	77
Forward DNS Lookup with Fierce.....	77
Reverse DNS	78
Reverse DNS Lookup with Dig.....	78
Reverse DNS Lookup with Fierce.....	78
Zone Transfers.....	79
Zone Transfer with Host Command	79
Automating Zone Transfers	80
DNS Cache Snooping.....	80
What Is DNS Cache Snooping?.....	81
Nonrecursive Method.....	81
Recursive Method.....	82
What Is the Likelihood of Name Servers Allowing Recursive/Nonrecursive Queries?	83
Attack Scenario.....	84
Automating DNS Cache Snooping Attacks	84
Enumerating SNMP.....	84
Problem with SNMP	84
Sniffing SNMP Passwords	84
OneSixtyOne.....	85
Snmpenum	85
SolarWinds Toolset.....	85
SNMP Sweep.....	86
SNMP Brute Force and Dictionary	86

SNMP Brute Force Tool	86
SNMP Dictionary Attack Tool	87
SMTP Enumeration	87
Detecting Load Balancers	88
Load Balancer Detector	89
Determining Real IP behind Load Balancers.....	89
Bypassing CloudFlare Protection.....	90
Method 1: Resolvers	90
Method 2: Subdomain Trick	92
Method 3: Mail Servers	92
Intelligence Gathering Using Shodan	93
Further Reading	95
Conclusion.....	95
4 Target Enumeration and Port Scanning Techniques.....	97
Host Discovery	97
Scanning for Open Ports and Services	100
Types of Port Scanning.....	100
Understanding the TCP Three-Way Handshake.....	101
TCP Flags.....	101
Port Status Types	102
TCP SYN Scan.....	102
TCP Connect Scan.....	103
NULL, FIN, and XMAS Scans.....	104
NULL Scan	104
FIN Scan	105
XMAS Scan.....	105
TCP ACK Scan	105
Responses	106
UDP Port Scan	106
Anonymous Scan Types.....	107
IDLE Scan.....	107
Scanning for a Vulnerable Host	107
Performing an IDLE Scan with NMAP	109
TCP FTP Bounce Scan	109
Service Version Detection	110
OS Fingerprinting	111
POF	111
Output.....	112
Normal Format.....	112
Grepable Format.....	112
XML Format	113
Advanced Firewall/IDS Evading Techniques	113
Timing Technique	114
Wireshark Output	114
Fragmented Packets	115
Wireshark Output	115

Source Port Scan.....	115
Specifying an MTU.....	116
Sending Bad Checksums	116
Decoys.....	117
ZENMAP.....	117
Further Reading	119
5 Vulnerability Assessment.....	121
What Are Vulnerability Scanners and How Do They Work?	121
Pros and Cons of a Vulnerability Scanner	122
Vulnerability Assessment with Nmap	122
Updating the Database	122
Scanning MS08 _ 067 _ netapi	123
Testing SCADA Environments with Nmap.....	123
Installation	124
Usage.....	124
Nessus Vulnerability Scanner.....	124
Home Feed	125
Professional Feed	125
Installing Nessus on BackTrack	125
Adding a User	125
Nessus Control Panel.....	126
Reports.....	126
Mobile.....	126
Scan	127
Policies.....	127
Users.....	127
Configuration.....	127
Default Policies.....	127
Creating a New Policy	128
Safe Checks.....	128
Silent Dependencies.....	128
Avoid Sequential Scans	128
Port Range.....	129
Credentials	129
Plug-Ins	129
Preferences	130
Scanning the Target.....	130
Nessus Integration with Metasploit.....	132
Importing Nessus to Metasploit.....	132
Scanning the Target.....	133
Reporting	133
OpenVas	133
Resource	134
Vulnerability Data Resources.....	134
Exploit Databases	135

Using Exploit-db with BackTrack	136
Searching for Exploits inside BackTrack	137
Conclusion.....	138
6 Network Sniffing	139
Introduction	139
Types of Sniffing	140
Active Sniffing	140
Passive Sniffing	140
Hubs versus Switches	140
Promiscuous versus Nonpromiscuous Mode	141
MITM Attacks	141
ARP Protocol Basics	142
How ARP Works	142
ARP Attacks	143
MAC Flooding	143
Macof	143
ARP Poisoning	144
Scenario—How It Works	144
Denial of Service Attacks	144
Tools of the Trade	145
Dsniff	145
Using ARP Spoof to Perform MITM Attacks.....	145
Usage.....	146
Sniffing the Traffic with Dsniff.....	147
Sniffing Pictures with Drifnet.....	147
Urlsnarf and Webspy	148
Sniffing with Wireshark.....	149
Ettercap	150
ARP Poisoning with Ettercap	150
Hijacking Session with MITM Attack.....	152
Attack Scenario.....	152
ARP Poisoning with Cain and Abel.....	153
Sniffing Session Cookies with Wireshark.....	155
Hijacking the Session.....	156
SSL Strip: Stripping HTTPS Traffic.....	157
Requirements.....	157
Usage.....	158
Automating Man in the Middle Attacks.....	158
Usage.....	158
DNS Spoofing	159
ARP Spoofing Attack	159
Manipulating the DNS Records	160
Using Ettercap to Launch DNS Spoofing Attack.....	160
DHCP Spoofing	160
Conclusion.....	161

7	Remote Exploitation.....	163
	Understanding Network Protocols.....	163
	Transmission Control Protocol.....	164
	User Datagram Protocol.....	164
	Internet Control Messaging Protocol.....	164
	Server Protocols.....	164
	Text-Based Protocols (Important).....	164
	Binary Protocols.....	164
	FTP.....	165
	SMTP.....	165
	HTTP.....	165
	Further Reading.....	165
	Resources.....	166
	Attacking Network Remote Services.....	166
	Overview of Brute Force Attacks.....	166
	Traditional Brute Force.....	166
	Dictionary Attacks.....	166
	Hybrid Attacks.....	167
	Common Target Protocols.....	167
	Tools of the Trade.....	167
	THC Hydra.....	167
	Basic Syntax for Hydra.....	168
	Cracking Services with Hydra.....	168
	Hydra GUI.....	170
	Medusa.....	170
	Basic Syntax.....	170
	OpenSSH Username Discovery Bug.....	170
	Cracking SSH with Medusa.....	171
	Ncrack.....	171
	Basic Syntax.....	171
	Cracking an RDP with Ncrack.....	172
	Case Study of a Morto Worm.....	172
	Combining Nmap and Ncrack for Optimal Results.....	172
	Attacking SMTP.....	173
	Important Commands.....	174
	Real-Life Example.....	174
	Attacking SQL Servers.....	175
	MySQL Servers.....	175
	Fingerprinting MySQL Version.....	175
	Testing for Weak Authentication.....	175
	MS SQL Servers.....	176
	Fingerprinting the Version.....	177
	Brute Forcing SA Account.....	177
	Using Null Passwords.....	178
	Introduction to Metasploit.....	178
	History of Metasploit.....	178

Metasploit Interfaces.....	178
MSFConsole.....	178
MSFcli.....	179
MSFGUI.....	179
Armitage.....	179
Metasploit Utilities.....	179
MSFPayload.....	179
MSFEncode.....	179
MSFVenom.....	179
Metasploit Basic Commands.....	180
Search Feature in Metasploit.....	180
Use Command.....	181
Info Command.....	181
Show Options.....	181
Set/Unset Command.....	182
Reconnaissance with Metasploit.....	182
Port Scanning with Metasploit.....	182
Metasploit Databases.....	182
Storing Information from Nmap into Metasploit Database.....	183
Useful Scans with Metasploit.....	184
Port Scanners.....	184
Specific Scanners.....	184
Compromising a Windows Host with Metasploit.....	184
Metasploit Autopwn.....	188
db _ autopwn in Action.....	188
Nessus and Autopwn.....	189
Armitage.....	189
Interface.....	190
Launching Armitage.....	190
Compromising Your First Target from Armitage.....	191
Enumerating and Fingerprinting the Target.....	191
MSF Scans.....	192
Importing Hosts.....	192
Vulnerability Assessment.....	193
Exploitation.....	193
Check Feature.....	195
Hail Mary.....	196
Conclusion.....	196
References.....	196
8 Client Side Exploitation.....	197
Client Side Exploitation Methods.....	197
Attack Scenario 1: E-Mails Leading to Malicious Attachments.....	197
Attack Scenario 2: E-Mails Leading to Malicious Links.....	197
Attack Scenario 3: Compromising Client Side Update.....	198
Attack Scenario 4: Malware Loaded on USB Sticks.....	198

E-Mails with Malicious Attachments	198
Creating a Custom Executable.....	198
Creating a Backdoor with SET	198
PDF Hacking	201
Introduction	201
Header.....	202
Body.....	202
Cross Reference Table.....	202
Trailer.....	202
PDF Launch Action.....	202
Creating a PDF Document with a Launch Action	203
Controlling the Dialog Boxes	205
PDF Reconnaissance	205
Tools of the Trade.....	205
PDFINFO	205
PDFINFO “Your PDF Document”	206
PDFTK	206
Origami Framework	207
Installing Origami Framework on BackTrack.....	207
Attacking with PDF.....	208
Fileformat Exploits	208
Browser Exploits.....	208
Scenario from Real World.....	209
Adobe PDF Embedded EXE.....	210
Social Engineering Toolkit.....	211
Attack Scenario 2: E-Mails Leading to Malicious Links	213
Credential Harvester Attack	214
Tabnabbing Attack	215
Other Attack Vectors	216
Browser Exploitation.....	217
Attacking over the Internet with SET	217
Attack Scenario over the Internet.....	217
Using Windows Box as Router (Port Forwarding).....	220
Browser AutoPWN.....	220
Why Use Browser AutoPWN?.....	221
Problem with Browser AutoPWN.....	221
VPS/Dedicated Server	223
Attack Scenario 3: Compromising Client Side Update	223
How Evilgrade Works.....	223
Prerequisites.....	223
Attack Vectors	223
Internal Network Attack Vectors	223
External Network Attack Vectors	224
Evilgrade Console.....	224
Attack Scenario.....	224
Attack Scenario 4: Malware Loaded on USB Sticks.....	227

Teensy USB	229
Conclusion.....	229
Further Reading	229
9 Postexploitation.....	231
Acquiring Situation Awareness.....	231
Enumerating a Windows Machine	231
Enumerating Local Groups and Users	233
Enumerating a Linux Machine	233
Enumerating with Meterpreter	235
Identifying Processes	235
Interacting with the System.....	235
User Interface Command	235
Privilege Escalation.....	236
Maintaining Stability	236
Escalating Privileges.....	237
Bypassing User Access Control	238
Impersonating the Token.....	239
Escalating Privileges on a Linux Machine.....	241
Maintaining Access.....	241
Installing a Backdoor	241
Cracking the Hashes to Gain Access to Other Services	241
Backdoors	241
Disabling the Firewall.....	242
Killing the Antivirus.....	242
Netcat.....	243
MSFPayload/MSFEncode.....	244
Generating a Backdoor with MSFPayload	244
MSFEncode.....	245
MSFVenom	246
Persistence	247
What Is a Hash?	249
Hashing Algorithms	249
Windows Hashing Methods	250
LAN Manager (LM)	250
NTLM/NTLM2.....	250
Kerberos	250
Where Are LM/NTLM Hashes Located?	250
Dumping the Hashes.....	251
Scenario 1—Remote Access.....	251
Scenario 2—Local Access.....	251
Ophcrack.....	252
References.....	253
Scenario 3—Offline System	253
Ophcrack LiveCD	253
Bypassing the Log-In.....	253

References.....	253
Cracking the Hashes.....	253
Bruteforce.....	253
Dictionary Attacks.....	254
Password Salts.....	254
Rainbow Tables.....	254
John the Ripper.....	255
Cracking LM/NTLM Passwords with JTR.....	255
Cracking Linux Passwords with JTR.....	256
Rainbow Crack.....	256
Sorting the Tables.....	257
Cracking the Hashes with rcrack.....	258
Speeding Up the Cracking Process.....	258
Gaining Access to Remote Services.....	258
Enabling the Remote Desktop.....	259
Adding Users to the Remote Desktop.....	259
Data Mining.....	259
Gathering OS Information.....	260
Harvesting Stored Credentials.....	261
Identifying and Exploiting Further Targets.....	262
Mapping the Internal Network.....	263
Finding Network Information.....	264
Identifying Further Targets.....	265
Pivoting.....	266
Scanning Ports and Services and Detecting OS.....	267
Compromising Other Hosts on the Network Having the Same Password.....	268
psexec.....	269
Exploiting Targets.....	270
Conclusion.....	270
10 Windows Exploit Development Basics.....	271
Prerequisites.....	271
What Is a Buffer Overflow?.....	271
Vulnerable Application.....	272
How to Find Buffer Overflows.....	273
Methodology.....	273
Getting the Software Up and Running.....	273
Causing the Application to Crash.....	273
Skeleton Exploit.....	275
Determining the Offset.....	278
Identifying Bad Characters.....	280
Figuring Out Bad Characters with Mona.....	281
Overwriting the Return Address.....	283
NOP Sledges.....	285
Generating the ShellCode.....	286
Generating Metasploit Module.....	287
Porting to Metasploit.....	288

Conclusion.....	290
Further Resources.....	290
11 Wireless Hacking	291
Introduction	291
Requirements.....	291
Introducing Aircrack-ng.....	293
Uncovering Hidden SSIDs	293
Turning on the Monitor Mode	294
Monitoring Beacon Frames on Wireshark	294
Monitoring with Airodump-ng.....	295
Speeding Up the Process.....	296
Bypassing MAC Filters on Wireless Networks.....	296
Cracking a WEP Wireless Network with Aircrack-ng	298
Placing Your Wireless Adapter in Monitor Mode.....	298
Determining the Target with Airodump-ng.....	299
Attacking the Target.....	299
Speeding Up the Cracking Process	300
Injecting ARP Packets.....	300
Cracking the WEP	301
Cracking a WPA/WPA2 Wireless Network Using Aircrack-ng	302
Capturing Packets.....	303
Capturing the Four-Way Handshake	303
Cracking WPA/WAP2	304
Using Reaver to Crack WPS-Enabled Wireless Networks	305
Reducing the Delay	306
Further Reading	306
Setting Up a Fake Access Point with SET to PWN Users.....	306
Attack Scenario.....	309
Evil Twin Attack.....	310
Scanning the Neighbors.....	311
Spoofing the MAC.....	311
Setting Up a Fake Access Point.....	311
Causing Denial of Service on the Original AP.....	311
Conclusion.....	312
12 Web Hacking.....	313
Attacking the Authentication.....	313
Username Enumeration	314
Invalid Username with Invalid Password	314
Valid Username with Invalid Password.....	314
Enabling Browser Cache to Store Passwords.....	314
Brute Force and Dictionary Attacks.....	315
Types of Authentication	315
HTTP Basic Authentication.....	315
HTTP-Digest Authentication.....	316
Form-Based Authentication	317
Exploiting Password Reset Feature	319

Etsy.com Password Reset Vulnerability.....	319
Attacking Form-Based Authentication.....	320
Brute Force Attack.....	322
Attacking HTTP Basic Auth.....	323
Further Reading	326
Log-In Protection Mechanisms.....	326
CAPTCHA Validation Flaw.....	326
CAPTCHA Reset Flaw.....	328
Manipulating User-Agents to Bypass CAPTCHA and Other Protections.....	329
Real-World Example.....	330
Authentication Bypass Attacks.....	330
Authentication Bypass Using SQL Injection.....	330
Testing for SQL Injection Auth Bypass.....	331
Authentication Bypass Using XPATH Injection	333
Testing for XPATH Injection	333
Authentication Bypass Using Response Tampering	334
Crawling Restricted Links	334
Testing for the Vulnerability.....	335
Automating It with Burp Suite	336
Authentication Bypass with Insecure Cookie Handling.....	336
Session Attacks	339
Guessing Weak Session ID	339
Session Fixation Attacks	341
Requirements for This Attack	342
How the Attack Works	342
SQL Injection Attacks	342
What Is an SQL Injection?	342
Types of SQL Injection.....	342
Union-Based SQL Injection	343
Error-Based SQL Injection	343
Blind SQL Injection	343
Detecting SQL Injection	343
Determining the Injection Type	343
Union-Based SQL Injection (MySQL).....	344
Testing for SQL Injection	344
Determining the Number of Columns	345
Determining the Vulnerable Columns.....	346
Fingerprinting the Database	347
Enumeration Information.....	347
Information_schema.....	348
Information_schema Tables.....	348
Enumerating All Available Databases	348
Enumerating All Available Tables in the Database.....	349
Extracting Columns from Tables.....	349
Extracting Data from Columns	350
Using group _ concat	350
MySQL Version \leq 5	351

Guessing Table Names.....	351
Guessing Columns.....	352
SQL Injection to Remote Command Execution	352
Reading Files	353
Writing Files	353
Blind SQL Injection	355
Boolean-Based SQLi.....	355
True Statement	355
False Statement.....	356
Enumerating the DB User	356
Enumerating the MYSQL Version.....	358
Guessing Tables	358
Guessing Columns in the Table.....	359
Extracting Data from Columns	360
Time-Based SQL Injection	361
Vulnerable Application	361
Testing for Time-Based SQL Injection	362
Enumerating the DB User	362
Guessing the Table Names.....	363
Guessing the Columns.....	364
Extracting Data from Columns	365
Automating SQL Injections with Sqlmap	366
Enumerating Databases	367
Enumerating Tables.....	367
Enumerating the Columns	367
Extracting Data from the Columns	368
HTTP Header-Based SQL Injection	368
Operating System Takeover with Sqlmap	369
OS-CMD	369
OS-SHELL.....	369
OS-PWN.....	370
XSS (Cross-Site Scripting)	371
How to Identify XSS Vulnerability.....	371
Types of Cross-Site Scripting	371
Reflected/Nonpersistent XSS	372
Vulnerable Code	372
Medium Security.....	373
Vulnerable Code.....	373
High Security	373
Bypassing htmlspecialchars.....	374
UTF-32 XSS Trick: Bypass 1.....	375
Svg Craziiness: Bypass 2.....	375
Bypass 3: href Attribute	376
Stored XSS/Persistent XSS	377
Payloads.....	377
Blind XSS	378
DOM-Based XSS	378

Detecting DOM-Based XSS.....	378
Sources (Inputs).....	378
Sinks (Creating/Modifying HTML Elements).....	378
Static JS Analysis to Identify DOM-Based XSS.....	384
How Does It Work?.....	385
Setting Up JSPRIME.....	385
Dominator: Dynamic Taint Analysis.....	390
POC for Internet Explorer.....	394
POC for Chrome.....	394
Pros/Cons.....	395
Cross Browser DOM XSS Detection.....	395
Types of DOM-Based XSS.....	397
Reflected DOM XSS.....	397
Stored DOM XSS.....	397
Exploiting XSS.....	399
Cookie Stealing with XSS.....	399
Exploiting XSS for Conducting Phishing Attacks.....	402
Compromising Victim's Browser with XSS.....	404
Exploiting XSS with BeEF.....	405
Setting Up BeEF on BackTrack.....	405
Demo Pages.....	408
BeEF Modules.....	409
Module: Replace HREFs.....	409
Module: Getcookie.....	409
Module: Tabnabbing.....	410
BeEF in Action.....	412
Cross-Site Request Forgery (CSRF).....	413
Why Does a CSRF Attack Work?.....	413
How to Attack.....	413
GET-Based CSRF.....	414
POST-Based CSRF.....	414
CSRF Protection Techniques.....	415
Referrer-Based Checking.....	415
Anti-CSRF Tokens.....	415
Predicting/Brute Forcing Weak Anti-CSRF Token Algorithm.....	416
Tokens Not Validated upon Server.....	416
Analyzing Weak Anti-CSRF Token Strength.....	417
Bypassing CSRF with XSS.....	419
File Upload Vulnerabilities.....	421
Bypassing Client Side Restrictions.....	423
Bypassing MIME-Type Validation.....	423
Real-World Example.....	425
Bypassing Blacklist-Based Protections.....	425
Case 1: Blocking Malicious Extensions.....	425
Bypass.....	426
Case 2: Case-Sensitive Bypass.....	426
Bypass.....	426

Real-World Example	426
Vulnerable Code	426
Case 3: When All Dangerous Extensions Are Blocked	426
XSS via File Upload	427
Flash-Based XSS via File Upload	428
Case 4: Double Extensions Vulnerabilities	429
Apache Double Extension Issues	429
IIS 6 Double Extension Issues	429
Case 5: Using Trailing Dots	429
Case 6: Null Byte Trick	429
Case 7: Bypassing Image Validation	429
Case 8: Overwriting Critical Files	430
Real-World Example	431
File Inclusion Vulnerabilities	431
Remote File Inclusion	432
Patching File Inclusions on the Server Side	433
Local File Inclusion	433
Linux	434
Windows	434
LFI Exploitation Using /proc/self/envIRON	434
Log File Injection	436
Finding Log Files: Other Tricks	440
Exploiting LFI Using PHP Input	440
Exploiting LFI Using File Uploads	441
Read Source Code via LFI	442
Local File Disclosure Vulnerability	443
Vulnerable Code	443
Local File Disclosure Tricks	445
Remote Command Execution	446
Uploading Shells	448
Server Side Include Injection	452
Testing a Website for SSI Injection	452
Executing System Commands	453
Spawning a Shell	453
SSRF Attacks	454
Impact	455
Example of a Vulnerable PHP Code	456
Remote SSRF	457
Simple SSRF	457
Partial SSRF	458
Denial of Service	463
Denial of Service Using External Entity Expansion (XEE)	463
Full SSRF	464
dict://	464
gopher://	465
http://	465
Causing the Crash	466

Overwriting Return Address.....	467
Generating Shellcode.....	467
Server Hacking.....	469
Apache Server.....	470
Testing for Disabled Functions.....	470
Open _ basedir Misconfiguration.....	472
Using CURL to Bypass Open _ basedir Restrictions.....	474
Open _ basedir PHP 5.2.9 Bypass.....	475
Reference.....	476
Bypassing open _ basedir Using CGI Shell.....	476
Bypassing open _ basedir Using Mod _ Perl, Mod _ Python.....	477
Escalating Privileges Using Local Root Exploits.....	477
Back Connecting.....	477
Finding the Local Root Exploit.....	478
Usage.....	478
Finding a Writable Directory.....	479
Bypassing Symlinks to Read Configuration Files.....	480
Who Is Affected?.....	481
Basic Syntax.....	481
Why This Works.....	482
Symlink Bypass: Example 1.....	482
Finding the Username.....	482
/etc/passwd File.....	483
/etc/aliases File.....	483
Path Disclosure.....	483
Uploading .htaccess to Follow Symlinks.....	484
Symlinking the Configuration Files.....	484
Connecting to and Manipulating the Database.....	485
Updating the Password.....	486
Symlink the Root Directory.....	486
Example 3: Compromising WHMCS Server.....	487
Finding a WHMCS Server.....	487
Symlinking the Configuration File.....	488
WHMCS Killer.....	488
Disabling Security Mechanisms.....	490
Disabling Mod _ Security.....	490
Disabling Open _ basedir and Safe _ mode.....	490
Using CGI, PERL, or Python Shell to Bypass Symlinks.....	491
Conclusion.....	491

Preface

Ethical hacking strikes all of us as a subject that requires a great deal of prerequisite knowledge about things like heavy duty software, languages that includes hordes of syntaxes, algorithms that could be generated by maestros only. Well that's not the case, to some extent. This book introduces the steps required to complete a penetration test, or ethical hack. Requiring no prior hacking experience, the book explains how to utilize and interpret the results of modern day hacking tools that are required to complete a penetration test. Coverage includes Backtrack Linux, Google Reconnaissance, MetaGooFil, dig, Nmap, Nessus, Metasploit, Fast Track Autopwn, Netcat, and Hacker Defender rootkit. Simple explanations of how to use these tools and a four-step methodology for conducting a penetration test provide readers with a better understanding of offensive security.

Being an ethical hacker myself, I know how difficult it is for people who are new into hacking to excel at this skill without having any prior knowledge and understanding of how things work. Keeping this exigent thing in mind, I have provided those who are keen to learn ethical hacking with the best possible explanations in the most easy and understandable manner so that they will not only gain pleasure while reading, but they will have the urge to put into practice what have they learned from it.

The sole aim and objective of writing this book is to target the beginners who look for a complete guide to turn their dream of becoming an ethical hacker into a reality. This book elucidates the building blocks of ethical hacking that will help readers to develop an insight of the matter in hand. It will help them fathom what ethical hacking is all about and how one can actually run a penetration test with great success.

I have put in a lot of hard work to make this book a success. I remember spending hours and hours in front of my computer typing indefatigably, ignoring all the text messages of my friends when they asked me to come along and spend some time with them, which left me despondent, but now, when I see my book finally completed, it gives me immense pleasure that the efforts of a whole year have finally paid off.

This book came out as a result of my own experiences during my ethical hacking journey. Experiences that are worth sharing with all the passionate people out there.

It makes me elated to the core when I see my third book on the subject of hacking published, and I hope and pray that everyone likes it.

Best of luck to everyone out there.

Rafay Baloch

Acknowledgments

I am eternally indebted to the editor, Rich O’Hanley, for his encouragement and continuous support and my dear friend Prakhar Prasad for his help at various stages of this book.

I also thank Mohammed Ramadan for his help and support and Soroush Dallili for his ideas with file upload tricks. Many thanks to my friends Alex Infuhr and Giuseppe Trotta for their help with various sections of the “Web Hacking” chapter, Shahmeer Amir for his help with the “Wireless Hacking” chapter, and Tehseen Javed for his help with the “Linux Basics” chapter.

I also thank my mentors Prof. Asim Rizvi, David Vieira-Kurz, Ziaullah Mirza and last but not least, I thank the following keypersons: Mario Heiderich, Deepankar Arora, Nir Goldshlager, Britto Fleming Joe, Nishant Das Patnaik, Pepe Vila, Ray friedman, Armando Romeo, Tyler Borland, Zeeshan Haider, Nehal hussain, Rafael Souza, and Fatima Hanif.

I also thank my family members and relatives for always being supportive.

Author

Rafay Baloch is the founder/CEO of RHA InfoSec. He runs one of the top security blogs in Pakistan with more than 25,000 subscribers (<http://rafayhackingarticles.net>). He has participated in various bug bounty programs and has helped several major Internet corporations such as Google, Facebook, Twitter, Yahoo!, eBay, etc., to improve their Internet security. Rafay was successful in finding a remote code execution vulnerability along with several other high-risk vulnerabilities inside PayPal, for which he was awarded a huge sum of money as well as an offer to work for PayPal. His major areas of research interest are in network security, bypassing modern security defenses such as WAFs, DOM-based XSS, and other HTML 5–based attack vectors. Rafay holds CPTe, CPTC, CSWAE, CVA, CSS, OSCP, CCNA R & S, CCNP Route, and eWAPT certifications.

Chapter 1

Introduction to Hacking

There are many definitions for “hacker.” Ask this question from a phalanx and you’ll get a new answer every time because “more mouths will have more talks” and this is the reason behind the different definitions of hackers which in my opinion is quite justified for everyone has a right to think differently.

In the early 1990s, the word “hacker” was used to describe a great programmer, someone who was able to build complex logics. Unfortunately, over time the word gained negative hype, and the media started referring to a hacker as someone who discovers new ways of hacking into a system, be it a computer system or a programmable logic controller, someone who is capable of hacking into banks, stealing credit card information, etc. This is the picture that is created by the media and this is untrue because everything has a positive and a negative aspect to it. What the media has been highlighting is only the negative aspect; the people that have been protecting organizations by responsibly disclosing vulnerabilities are not highlighted.

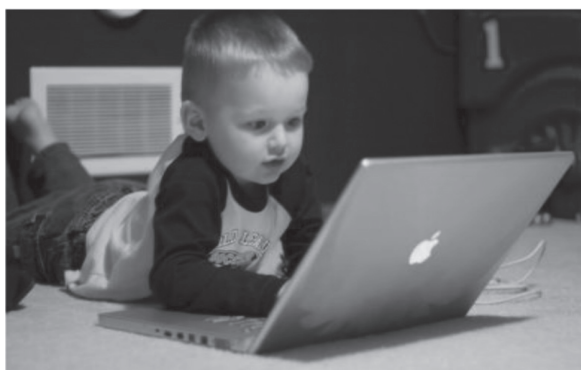
However, if you look at the media’s definition of a hacker in the 1990s, you would find a few common characteristics, such as creativity, the ability to solve complex problems, and new ways of compromising targets. Therefore, the term has been broken down into three types:

1. *White hat hacker*—This kind of hacker is often referred to as a security professional or security researcher. Such hackers are employed by an organization and are permitted to attack an organization to find vulnerabilities that an attacker might be able to exploit.
2. *Black hat hacker*—Also known as a *cracker*, this kind of hacker is referred to as a *bad guy*, who uses his or her knowledge for negative purposes. They are often referred to by the media as *hackers*.
3. *Gray hat hacker*—This kind of hacker is an intermediate between a white hat and a black hat hacker. For instance, a gray hat hacker would work as a security professional for an organization and responsibly disclose everything to them; however, he or she might leave a backdoor to access it later and might also sell the confidential information, obtained after the compromise of a company’s target server, to competitors.

2 ■ Ethical Hacking and Penetration Testing Guide

Similarly, we have categories of hackers about whom you might hear oftentimes. Some of them are as follows:

Script kiddie—Also known as *skid*, this kind of hacker is someone who lacks knowledge on how an exploit works and relies upon using exploits that someone else created. A script kiddie may be able to compromise a target but certainly cannot debug or modify an exploit in case it does not work.



(From <http://cdn.kaskus.com> and <http://the-gist.org>.)

Elite hacker—An elite hacker, also referred to as *l33t* or *1337*, is someone who has deep knowledge on how an exploit works; he or she is able to create exploits, but also modify codes that someone else wrote. He or she is someone with elite skills of hacking.

Hactivist—Hacktivists are defined as group of hackers that hack into computer systems for a cause or purpose. The purpose may be political gain, freedom of speech, human rights, and so on.

Ethical hacker—An ethical hacker is as a person who is hired and permitted by an organization to attack its systems for the purpose of identifying vulnerabilities, which an attacker might take advantage of. The sole difference between the terms “hacking” and “ethical hacking” is the permission.

Important Terminologies

Let’s now briefly discuss some of the important terminologies that I will be using throughout this book.

Asset

An asset is any data, device, or other component of the environment that supports information-related activities that should be protected from anyone besides the people that are allowed to view or manipulate the data/information.

Vulnerability

Vulnerability is defined as a flaw or a weakness inside the asset that could be used to gain unauthorized access to it. The successful compromise of a vulnerability may result in data manipulation, privilege elevation, etc.

Threat

A threat represents a possible danger to the computer system. It represents something that an organization doesn't want to happen. A successful exploitation of vulnerability is a threat. A threat may be a malicious hacker who is trying to gain unauthorized access to an asset.

Exploit

An exploit is something that takes advantage of vulnerability in an asset to cause unintended or unanticipated behavior in a target system, which would allow an attacker to gain access to data or information.

Risk

A risk is defined as the impact (damage) resulting from the successful compromise of an asset. For example, an organization running a vulnerable apache tomcat server poses a threat to an organization and the damage/loss that is caused to the asset is defined as a risk.

Normally, a risk can be calculated by using the following equation:

$$\text{Risk} = \text{Threat} * \text{vulnerabilities} * \text{impact}$$

What Is a Penetration Test?

A penetration test is a subclass of ethical hacking; it comprises a set of methods and procedures that aim at testing/protecting an organization's security. The penetration tests prove helpful in finding vulnerabilities in an organization and check whether an attacker will be able to exploit them to gain unauthorized access to an asset.

Vulnerability Assessments versus Penetration Test

Oftentimes, a vulnerability assessment is confused with a penetration test; however, these terms have completely different meanings. In a vulnerability assessment, our goal is to figure out all the vulnerabilities in an asset and document them accordingly.

In a penetration test, however, we need to simulate as an attacker to see if we are actually able to exploit a vulnerability and document the vulnerabilities that were exploited and the ones that turned out to be false-positive.

Preengagement

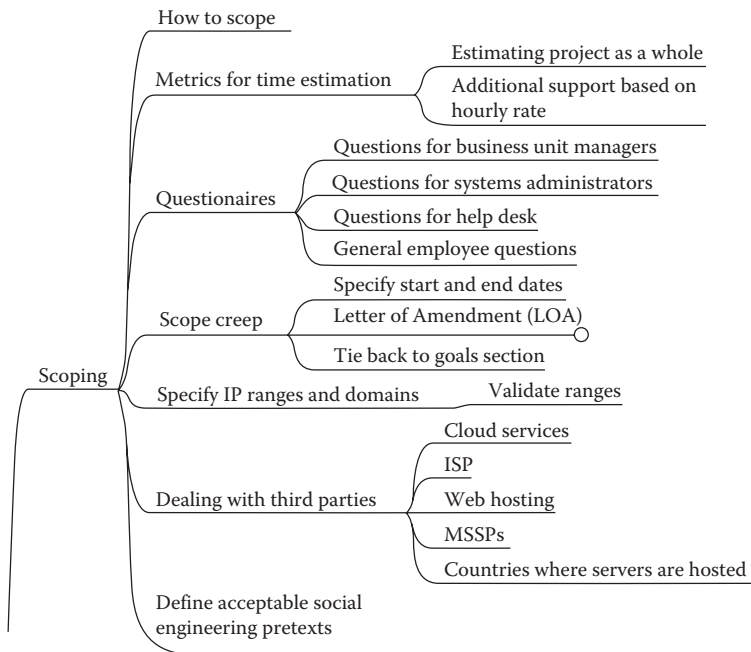
Before you start doing a penetration test, there is whole lot of things you need to discuss with clients. This is the phase where both the customer and a representative from your company would sit down and discuss about the legal requirements and the "rules of engagement."

Rules of Engagement

Every penetration test you do would comprise of a rules of engagement, which basically defines how a penetration test would be laid out, what methodology would be used, the start and end dates, the milestones, the goals of the penetration test, the liabilities and responsibilities, etc. All of them have to be mutually agreed upon by both the customer and the representative before the penetration test is started. Following are important requirements that are present in almost every ROE:

- A proper “permission to hack” and a “nondisclosure” agreement should be signed by both the parties.
- The scope of the engagement and what part of the organization must be tested.
- The project duration including both the start and the end date.
- The methodology to be used for conducting a penetration test.
- The goals of a penetration test.
- The allowed and disallowed techniques, whether denial-of-service testing should be performed or not.
- The liabilities and responsibilities, which are decided ahead of time. As a penetration tester you might break into something that should not be accessible, causing a denial of service; also, you might access sensitive information such as credit cards. Therefore, the liabilities should be defined prior to the engagement.

If you need a more thorough documentation, refer to the “PTES Pre-engagement” document (<http://www.pentest-standard.org/index.php/Pre-engagement>)



Milestones

Before starting a penetration test, it’s good practice to set up milestones so that your project is delivered as per the dates given in the rules of engagement.

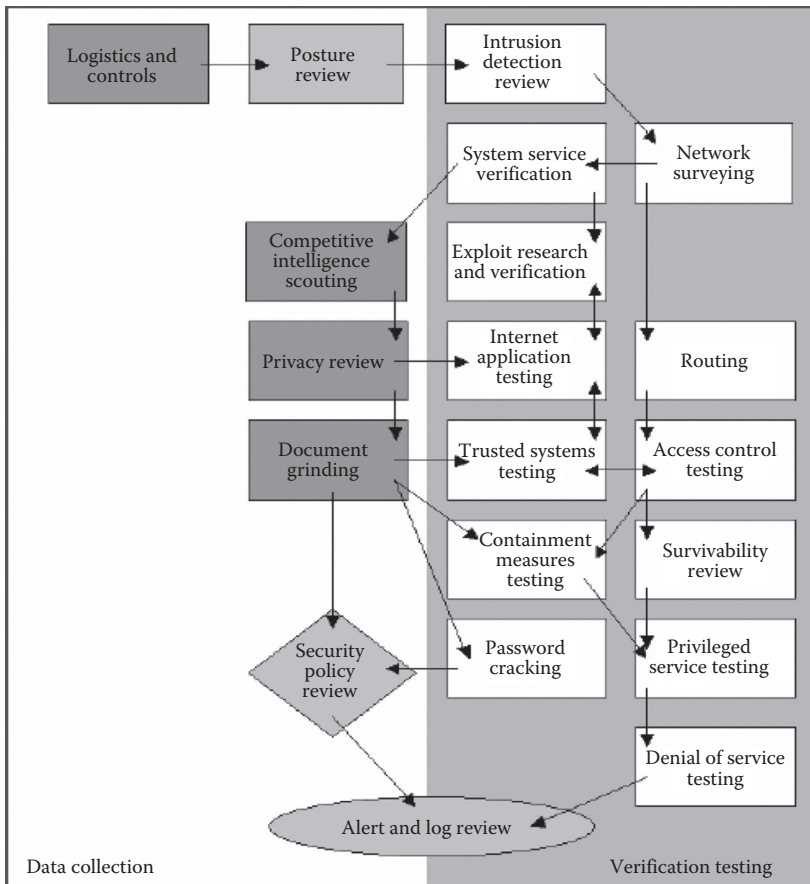
You can use either a GANTT chart or a website like Basecamp that helps you set up milestones to keep track of your progress. The following is a chart that defines the milestones followed by the date they should be accomplished.

Start	End	Month	Year	Phases
12th May 2013	18th	May	2013	Scope Definition
19th May 2013	27th	May	2013	Reconnaissance
28th May 2013	2th	June	2013	Scanning
3rd June 2013	16th	June	2013	Exploitation
17th June 2013	21th	June	2013	POST Exploitation
21st June 2013	28th	June	2013	Reporting

Penetration Testing Methodologies

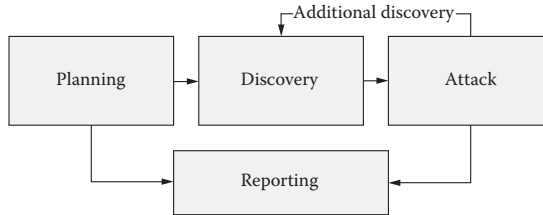
In every penetration test, methodology and the reporting are the most important steps. Let's first talk about the methodology. There are several different types of penetration testing methodologies that address how a penetration test should be performed. Some of them are discussed in brief next.

OSSTMM



An open-source security testing methodology manual (OSSTMM) basically includes almost all the steps involved in a penetration test. The methodology employed for penetration test is concise yet it's a cumbersome process which makes it difficult to implement it in our everyday life. Penetration tests, despite being tedious, demands a great deal of money out of company's budgets for their completion which often are not met by a large number of organizations.

NIST

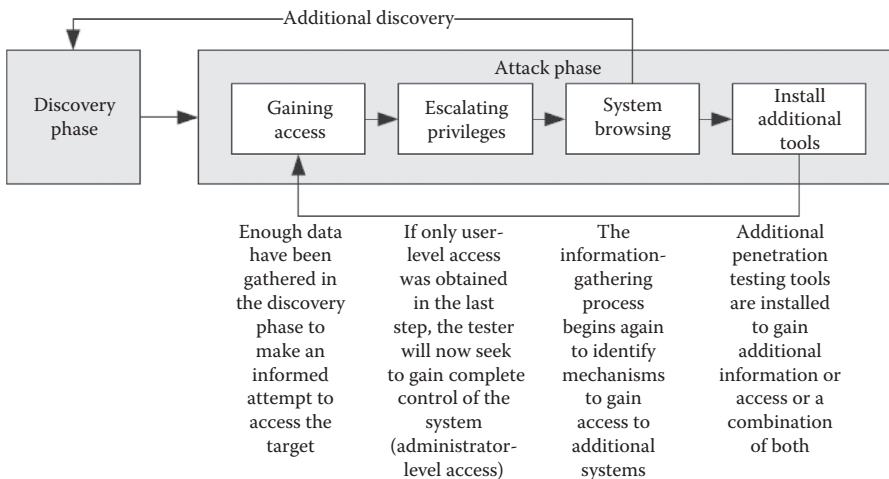


NIST, on the other hand, is more comprehensive than OSSTMM, and it's something that you would be able to apply on a daily basis and in short engagements. The screenshot indicates the four steps of the methodology, namely, planning, discovery, attack, and reporting.

The testing starts with the *planning* phase, where how the engagement is going to be performed is decided upon. This is followed by the *discovery* phase, which is divided into two parts—the first part includes information gathering, network scanning, service identification, and OS detection, and the second part involves vulnerability assessment.

After the discovery phase comes the *attack* phase, which is the heart of every penetration test. If you are able to compromise a target and a new host is discovered, in case the system is dual-homed or is connected with multiple interfaces, you would go back to step 2, that is, discovery, and repeat it until no targets are left. The indicating arrows in the block phase and the attack phase to the reporting phase indicate that you plan something and you report it—you attack a target and report the results.

The organization also has a more detailed version of the chart discussed earlier, which actually explains more about the *attack* phase. It consists of things such as “gaining access,” “escalating privileges,” “system browsing,” and “install additional tools.” We will go through each of these steps in detail in the following chapters.



OWASP

As you might have noticed, both the methodologies focused more on performing a network penetration test rather than something specifically built for testing web applications. The OWASP testing methodology is what we follow for all “application penetration tests” we do here at the RHA InfoSEC. The OWASP testing guide basically contains almost everything that you would test a web application for. The methodology is comprehensive and is designed by some of the best web application security researchers.

Categories of Penetration Test

When the scope of the penetration test is defined, the category/type of the penetration test engagement is also defined along with it. The entire penetration test can be Black Box, White Box, or Gray Box depending upon what the organization wants to test and how it wants the security paradigm to be tested.

Black Box

A black box penetration test is where little or no information is provided about the specified target. In the case of a network penetration test this means that the target’s DMZ, target operating system, server version, etc., will not be provided; the only thing that will be provided is the IP ranges that you would test. In the case of a web application penetration test, the source code of the web application will not be provided. This is a very common scenario that you will encounter when performing an external penetration test.

White Box

A white box penetration test is where almost all the information about the target is provided. In the case of a network penetration test, information on the application running, the corresponding versions, operating system, etc., are provided. In the case of a web application penetration test the application’s source code is provided, enabling us to perform the static/dynamic “source code analysis.” This scenario is very common in internal/onsite penetration tests, since organizations are concerned about leakage of information.

Gray Box

In a gray box test, some information is provided and some hidden. In the case of a network penetration test, the organization provides the names of the application running behind an IP; however, it doesn’t disclose the exact version of the services running. In the case of a web application penetration test, some extra information, such as test accounts, back end server, and databases, is provided.

Types of Penetration Tests

There are several types of penetration tests; however, the following are the ones most commonly performed:

Network Penetration Test

In a network penetration test, you would be testing a network environment for potential security vulnerabilities and threats. This test is divided into two categories: external and internal penetration tests.

An external penetration test would involve testing the public IP addresses, whereas in an internal test, you can become part of an internal network and test that network. You may be provided VPN access to the network or would have to physically go to the work environment for the penetration test depending upon the engagement rules that were defined prior to conducting the test.

Web Application Penetration Test

Web application penetration test is very common nowadays, since your application hosts critical data such as credit card numbers, usernames, and passwords; therefore this type of penetration test has become more common than the network penetration test.

Mobile Application Penetration Test

The mobile application penetration test is the newest type of penetration test that has become common since almost every organization uses Android- and iOS-based mobile applications to provide services to its customers. Therefore, organizations want to make sure that their mobile applications are secure enough for users to rely on when providing personal information when using such applications.

Social Engineering Penetration Test

A social engineering penetration test can be part of a network penetration test. In a social engineering penetration test the organization may ask you to attack its users. This is where you use spearphishing attacks and browser exploits to trick a user into doing things they did not intend to do.

Physical Penetration Test

A physical penetration test is what you would rarely be doing in your career as a penetration tester. In a physical penetration test, you would be asked to walk into the organization's building physically and test physical security controls such as locks and RFID mechanisms.

Report Writing

In any penetration test, the report is the most crucial part. Writing a good report is key to successful penetration testing. The following are the key factors to a good report:

- Your report should be simple, clear, and understandable.
- Presentation of the report is also important. Headers, footers, appropriate fonts, well-spaced margins, etc., should be created/selected properly and with great care. For example, if you are using a red font for the heading, every heading in the document should be in that style.
- The report should be well organized.

- Correct spelling and grammar is important too. A misspelled word leaves a very negative impact upon the person who is reading your report. So, you should make sure that you proofread your report and perform spell-checks before submitting it to the client.
- Always make sure that you use a consistent voice and style in writing a report. Changing the voice would create confusion in the reader; so you should choose one voice and style and stick to it throughout your report.
- Make sure you spend time on eliminating false-positives (vulnerabilities that are actually not present), because false-negatives will always be there no matter what you do. Eliminating the false-positives would enhance the credibility of the report.
- Perform a detailed analysis of the vulnerability to find out its root cause. A screenshot of a RAW http request or the screenshot that demonstrates the evidence of the finding would give a clear picture to the developer of the status.

Understanding the Audience

Understanding the audience that would be reading your penetration testing report is a very crucial part of the penetration test. We can divide the audience into three different categories:

1. Executive class
2. Management class
3. Technical class

While writing a report, you must understand which audience would read which part of your report; for example, the company's CEO would not be interested in what exploit you used to gain access to a particular machine, but on the flip side, your developers will probably not be interested in the overall risks and potential losses to the company; instead, they would be interested in fixing the code and therefore in reading about detailed findings. Let's briefly talk about the three classes.

Executive Class

This category includes the CEOs of the company. Since they have a very tedious schedule and most of the times have less technical knowledge, they would end up reading a very small portion of the report, specifically the executive summary, remediation report, etc., which we will discuss later in this chapter.

Management Class

Next, we have the management class, which includes the CISOs and CISSPs of the company. Since they are the ones who are responsible for implementing the security policy of the company, they would probably be a bit more interested in reading about overall strengths and weaknesses, the remediation report, the vulnerability assessment report, etc.

Technical Class

This class includes the security manager and developers, who would be interested in reading your report thoroughly. They would investigate your report as they are responsible for patching the weaknesses found and for making sure that the necessary patches are implemented.

Writing Reports

Now we are going to get into the essentials of the reporting phase, which will teach you about the structure of a report. We have discussed what a good report should look like. I pointed out that knowing your audience was essential. One of the key factors about a good report is that it should meet the needs for each audience and be presented in a clear and understandable manner.

The next major part of writing a report is the analysis, where we perform risk assessment and calculate the overall risk to the organization based upon our findings; along with this, your report should also provide remediation on how the risk can be averted.

Structure of a Penetration Testing Report

Let's look step by step on how a good report should be laid out. At the end of this chapter, I have provided links to some of the best reports which have been provided to the local mass.

Cover Page

We start with the cover page; this is where you would include details such as your company logo, title, and a short description about the penetration test. I would suggest you hire a good designer and work on a professional and appealing cover page because if your cover page looks great, it would make a good first impression upon the customer reading it.

Table of Contents

On the very next page, you should have an index so that the audience interested in reading a particular portion of the report can easily skip to that portion.

Table of Contents	
Executive Summary	3
Engagement Highlights	3
Vulnerability Report	4
Remediation Report	4
Findings Summary	5
Detailed Summary	5
E1 – DOM Based XSS Vulnerability	5
E2 – Stored Cross Site Scripting Vulnerability	6
E3 – Stored Cross Site Scripting Vulnerability	8
E4 – Blind XSS Vulnerability	10
E5 – Arbitrary File Upload Vulnerability	12
E6 – SOAP Based SQL Injection Vulnerability	13
E7 – Configuration File Disclosure	16
E8 – Administrative Login And Database Manipulation	17

Executive Summary

As the name suggests, an executive summary is the portion that is specifically addressed to executives such as the CEO or the CIO of the company. The executive summary is the most essential part of a penetration testing report; a good executive summary can make all the difference between a good report and a bad one.

Since the executive summary is specifically written to address the nontechnical audience, you should make sure that it's presented in such a way that it's easily comprehensible. Following are some of the essential points that you should take into consideration while writing an executive summary.

- Since executives are very busy, they have minimal time to invest in reading your reports. Therefore you should make sure that your executive summary is precise and to the point.
- Your executive summary should start with defining the purpose of the engagement and how it was carried out. Things such as the scope should be defined but very precisely.
- Next, you should explain the results of the penetration test and the findings.
- Following this, you should discuss the overall weaknesses in general and the countermeasures that were not implemented that caused the vulnerability in the first place.
- Next comes the analysis part; this is where you should write about the overall risk that was determined based upon our findings.
- And, finally, you should write about to what extent the risk would decrease after addressing the issues and implementing the appropriate countermeasures.

The following is an example of an executive summary that we wrote for a customer. I would suggest you spend some time reviewing the essential points discussed and compare them with the executive summary that follows.

EXECUTIVE SUMMARY

RHInfoSec conducted a full webapplication penetration test on **foonetworks**, the goal was to analyze the security posture of the Webapplications and suggest countermeasures for all the findings requiring remediation.

The Application Penetration test was conducted on foonetworks from January 2013 onwards. The target subdomains were also included in the scope of penetration test, which were not provided by default since it was a full black box penetration test.

As a result of the engagement we managed to find lots of high risk vulnerabilities which confirmed that the security posture of the application is very low and proper security countermeasures have not been implemented inside the environment.

This report contains detailed analysis about the vulnerabilities that we found during the engagement along with the report also contains a remediation report which would help you improve the overall security posture of your application. The report also contains a detailed explanation about every vulnerability found along with the detailed countermeasures to fix the vulnerability.

The overall risk of compromise was analyzed to be 70%. Addressing the security issues that present inside the report would significantly increase the overall risk of compromise.

Remediation Report

Next up we have the remediation report, which contains the overall recommendations that once implemented would increase the security of the organization. This is specifically an area of interest for the management class, as they are the ones that are going to enforce the security policies of an organization.

As mentioned earlier, these guys may or may not be technical; therefore our remediation report should be very precise and easy to understand. Things that could improve overall security such as implementing SDLC, a firewall, and an intrusion detection system should be recommended. The following is an example of how a remediation report should look like:

REMEDIATION

The security control environment for foonetworks was found very poor, as a result of which there are certain security countermeasures we would like to suggest. With the goal of protecting the Web application's infrastructure, we would recommend you to perform the following actions.

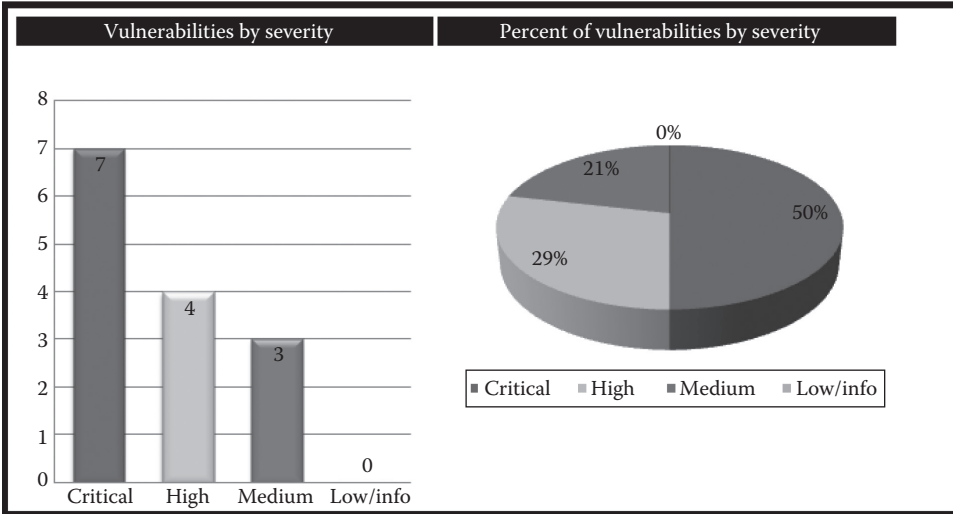
- A perfect plan for fixing the Critical, High, Medium, low risk vulnerabilities should designed and implemented. The vulnerabilities should be fixed in the descending order of priority.
- Secure development life cycle (SDLC) for developing web applications shall be implemented.
- A Web Application Firewall shall be implemented to detect, filter and block all the malicious packets.
- Security Audits shall be performed on the regular basis.
- Early security checks should be performed in the development process.

Vulnerability Assessment Summary

Next, we have the vulnerability assessment summary, sometimes referred to as “findings summary.” This is where we present the findings from our engagement. Things such as the overall strengths and weaknesses and risk assessment summary can also be included under this section.

“A picture speaks a thousand words” is a brilliant quotation that all of us remember from our childhood, don't we? Behold, for now it's time to see the actual use of it. It always helps to include charts in your report, which would give the audience a better understanding of the vulnerabilities that were found. Security executives might be interested in this portion of the report as they would need to enforce the countermeasures.

There are different ways for representing vulnerability assessment outputs in the form of graphical charts. Personally, I include two graphs; the first one classifies the vulnerability assessment on the basis of the severity and the second one on percentage.



Next, I include a “vulnerabilities breakdown” chart, where I talk about the findings for a particular host followed by the number of vulnerabilities that were found.

Vulnerabilities breakdown						
S #	IP Address	Hostname	Critical	High	Medium	Low/Info
1	192.254.236.66	Services.rafayhackingarticles.net	3	14	7	0
2	192.254.236.67	Tools.rafayhackingarticles.net	2	6	4	0

Tabular Summary

A tabular summary is also a great way to present the findings of a vulnerability assessment to a customer. The following screenshot comes directly from the “NII Report” and summarizes the vulnerability assessment based upon the number of live hosts and also talks about the number of findings with high, moderate, or low risk.

Category	Description			
Systems vulnerability assessment summary				
Number of live hosts	50			
Number of vulnerabilities	29			
High, medium, and info severity vulnerabilities	<table border="1"> <tr> <td>14</td> <td>6</td> <td>9</td> </tr> </table>	14	6	9
14	6	9		

Risk Assessment

Risk assessment as defined before is the analysis part of the report. It is very crucial for the customer because they would want to know the intensity of the damage the vulnerabilities are likely to cause; similarly, the security executives would also want to know how their team is performing.

Risk Assessment Matrix

When we talk about risk assessment analysis in terms of a penetration test, we compare the “likelihood of the occurring” and the “impact caused by the occurring.”

The following is a “hazard risk assessment matrix” derived from MIL-STD-882B; it’s an excellent method for demonstrating risk to the customer. In the following matrix the “frequency of occurrence,” that is, the likelihood of how often the vulnerability is occurring, is compared with the four hazard categories “catastrophic,” “critical,” “serious,” “minor,” and this is something you should definitely include in your penetration testing report.

Hazard risk assessment matrix

Frequency of Occurrence	Hazard Categories			
	1 Catastrophic	2 Critical	3 Serious	4 Minor
(A) Frequent	1A	2A	3A	4A
(B) Probable	1B	2B	3B	4B
(C) Occasional	1C	2C	3C	4C
(D) Remote	1D	2D	3D	4D
(E) Improbable	1E	2E	3E	4E

 Unacceptable
  High
  Medium
  Low

(From <http://www.sms-ink.com>.)

After including the risk assessment matrix, you should write a line or two describing the total risk.

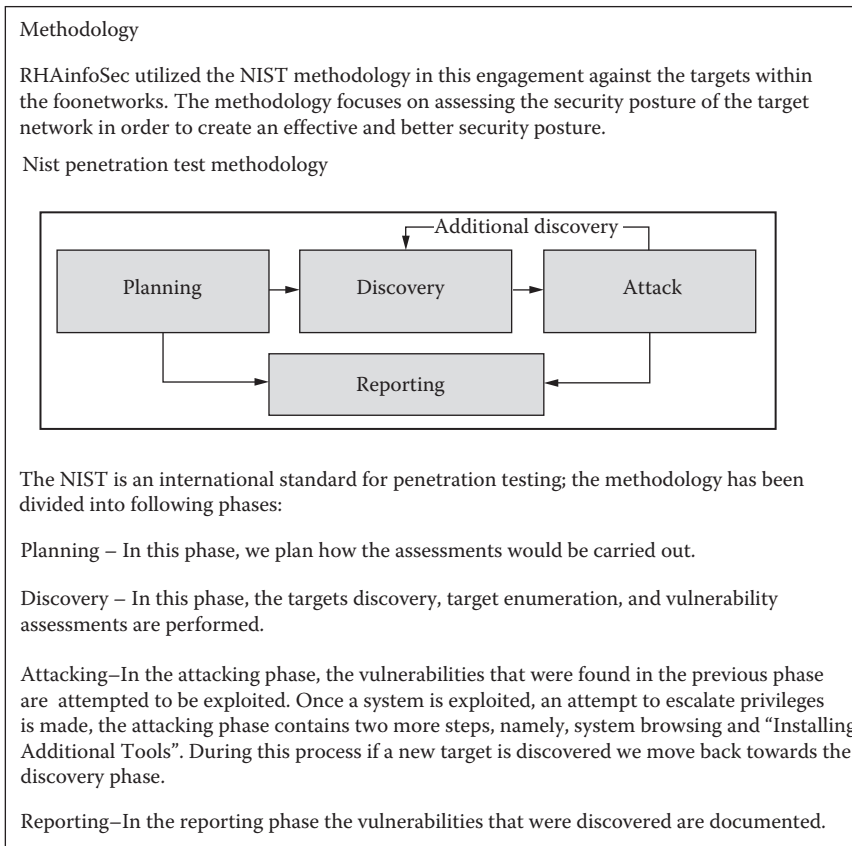
Based upon the comparison of the vulnerabilities that were determined, their likelihood and their impact we conclude the overall risk is high and the risk percentage was determined to be 82%.

Methodology

We have discussed a wide variety of methodologies and standards of penetration testing, such as OSSTMM, NIST, and OWASP. I would also like to include the methodology that was followed

for conducting the penetration test; though its inclusion in the report is optional, it could add great value to your penetration report. In a scenario where you have been asked to follow a certain standard, talking about the methodology and its steps is a good idea.

The following is a screenshot from one of our penetration testing reports where the NIST methodology was followed in order to conduct the penetration test. Notice that we include the flowchart on how the methodology works and explain each step precisely.



Detailed Findings

This is where you address the technical audience, specifically the security manager and the developers; also, this is where you are allowed to talk in depth about how the vulnerabilities were discovered, the root causes of the vulnerabilities, the associated risks, and the necessary recommendations.

Let’s now briefly talk about four essentials that should be included in the “Detailed Findings” section.

Description

This is where you talk about the vulnerability itself; a brief explanation should be provided in this section.

Explanation

This is the section where you reveal where the vulnerability was found, how it was found, the root cause of the vulnerability, the proof of concept, or the evidence of the finding.

Risk

This is where you talk about the risks and the likely impact that the vulnerability carries.

Recommendation

This is where you address the developers on how to fix the vulnerability; you may also include general suggestions to avoid that particular class of vulnerability in future.

The following screenshot comes directly from one of our penetration testing reports. Our finding was “DOM-based XSS” vulnerability. In the “Description” section we discussed the vulnerability. In the “Explanation” section, we talked about where the vulnerability was found and what line of the JavaScript code is the root cause of the vulnerability. We then talked about general risks and the impact and finally the general remediations to avoid vulnerabilities of a similar class.

DOM Based Cross Site Scripting Vulnerability
Affected Hosts: foonetworks.com
<i>Risk: Critical</i>
Description: A DOM Based XSS is a type of Cross site scripting vulnerability which occurs when the user supplied input passed through a source is not filtered/escaped before it's passed through a vulnerable sink.
Explanation: A dynamic file is being included which handles "location.hash" on the document object model (DOM). http://foonetworks.com/engine.js The following lines indicate the vulnerable code: Lines: 410 – 411: <pre>if(!=undefined){window.location.hash=t;}}); \$(window).bind("load", function() {if(window.location.hash){var _9=window.location.hash.substring(1);}</pre>
Risk Since javascript can access the DOM, an attacker can craft a special piece of javascript that would be able to steal the authentication cookies and send it the domain that he controls. In case of a DOM based XSS, the payload is always executed on the client side, this means this makes it difficult to trace the attacker from the forensics perspective, since the attack vector would not appear inside the log file.
Recommendations: Any user-generated input should be HTML-encoded at any point where it is copied into application responses. All HTML metacharacters should be replaced with the corresponding HTML entities.

Reports

Now that you know the basics and structure of how a penetration testing report is written, I would urge you to spend some time reviewing the following penetration testing sample reports.

- <http://www.offensive-security.com/penetration-testing-sample-report.pdf>
- http://www.niiconsulting.com/services/security-assessment/NII_Sample_PT_Report.pdf
- <http://pentestreports.com/>

Conclusion

In this chapter, we talked about basic terminologies that you will encounter on a daily basis as a penetration tester. We discussed about the types of penetration tests and the different penetration testing methodologies. We then talked about what makes a good penetration testing report. We also looked at how a penetration test report should be laid out in order to provide the target audience the necessary information.

Chapter 2

Linux Basics

In order to become a good ethical hacker or penetration tester, you need to be conversant with Linux, which is by far one of the most powerful operating systems. Linux is really good for ethical hacking and penetration testing because it is compatible with a wide variety of related tools and software, whereas other operating systems such as Mac and Windows support fewer of these software and tools. In this chapter, I will teach you some of the very basics of operating a Linux OS. If you are already familiar with Linux basics, you can skip this chapter.

One of the most common questions asked in many forums is “Which Linux distro should I use?” As there are tons of Linux distros such as Ubuntu, Fedora, Knoppix, and BackTrack you can use any Linux distro you want as all work in a similar manner. However, I suggest you use BackTrack if you really wish to dig deeper into this subject because it is all encompassing from a penetration tester’s perspective.

Major Linux Operating Systems

Before talking about BackTrack, let’s take a look at some of the Linux-based distros that you will encounter very often:

Redhat Linux—Used mostly for administration purpose.

Debian Linux—Designed for using only in open source software.

Ubuntu Linux—Designed mostly for personal use.

Mac OS X—Used in all Apple computers.

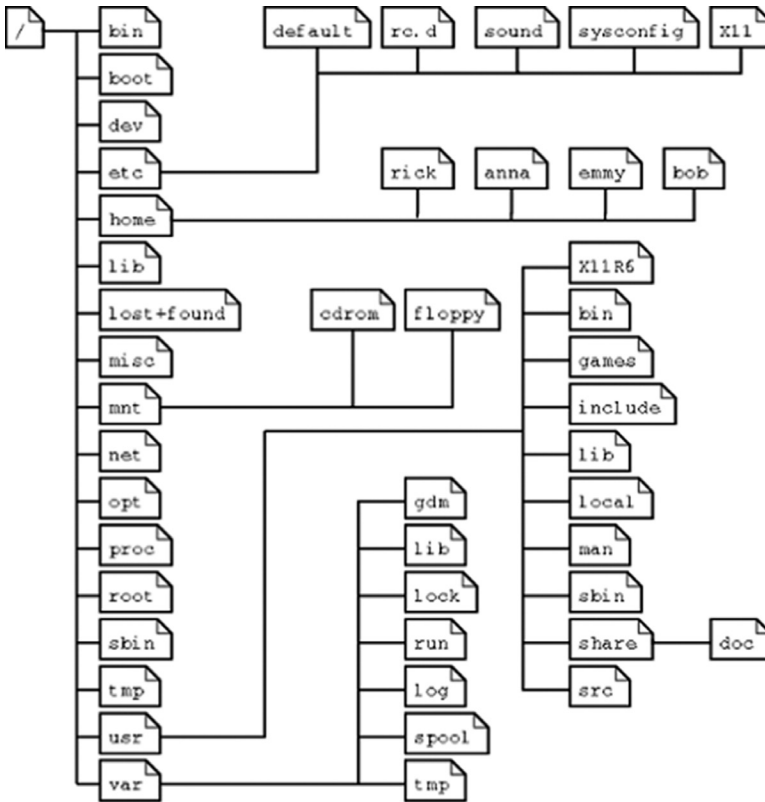
Solaris—Used in many commercial environments.

BackTrack Linux—Used mostly for penetration testing.

File Structure inside of Linux

On a Linux system, most everything is a file, and if it is not a file, then it is a process.

Here is a general diagram for file structure in Linux.



There are certain exceptions in a Linux file system

Directories—Files that are lists of other files.

Special file—The mechanism used for input and output. /dev are special files.

Links—A system to make file or directory visible in multiple parts of the systems.

Sockets—A special file type, similar to TCP/IP sockets providing inter-process networking.

Pipes—More or less like sockets; they form a way for process to communicate with each other with out using network socket.

File types in a long list:

Symbol	Meaning
-	Regular file
d	Directory
l	Link
c	Special file

s	Socket
p	Named pipe
b	Block device

Subdirectories of the root directory:

<i>Directory</i>	<i>Content</i>
/bin	Common programs, shared by the system, the system administrator, and the users.
/boot	The startup files and the kernel, vmlinuz. In some recent distributions also grub data. Grub is the GRand Unified Boot loader and is an attempt to get rid of the many different boot-loaders we know today.
/dev	Contains references to all the CPU peripheral hardware, which are represented as files with special properties.
/etc	Most important system configuration files are in/etc., this directory contains data similar to those in the Control Panel in Windows
/home	Home directories of the common users.
/initrd	(on some distributions) Information for booting. Do not remove!
/lib	Library files, includes files for all kinds of programs needed by the system and the users.
/lost+found	Every partition has a lost+found in its upper directory. Files that were saved during failures are here.
/misc	For miscellaneous purposes.
/mnt	Standard mount point for external file systems, for example, a CD-ROM or a digital camera.
/net	Standard mount point for entire remote file systems.
/opt	Typically contains extra and third-party software.
/proc	A virtual file system containing information about system resources. More information about the meaning of the files in proc is obtained by entering the command man proc in a terminal window. The file proc.txt discusses the virtual file system in detail.
/root	The administrative user's home directory. Mind the difference between /, the root directory and /root, the home directory of the root user.
/sbin	Programs for use by the system and the system administrator.
/tmp	Temporary space for use by the system, cleaned upon reboot, so don't use this for saving any work!
/usr	Programs, libraries, documentation, etc., for all user-related programs.
/var	Storage for all variable files and temporary files created by users, such as log files, the mail queue, the print spooler area, space for temporary storage of files downloaded from the Internet, or to keep an image of a CD before burning it.

File Permission in Linux

Although there are already a lot of good security features built into Linux-based systems, based upon the need for proper permissions, I will go over the ways to assign permissions and show you some examples where modification may be necessary. Wrong file permission may open a door for attackers in your system.

Group Permission

Owner—The Owner permissions apply only the owner of the file or directory; they will not impact the actions of other users.

Group—The Group permissions apply only to the group that has been assigned to the file or directory; they will not affect the actions of other users.

All User/Other—The All Users permissions apply to all other users on the system; this is the permission group that you want to watch the most.

Each file or directory has three basic permission types:

Read—The Read permission refers to a user’s capability to read the contents of the file.

Write—The Write permissions refer to a user’s capability to write or modify a file or directory.

Execute—The Execute permission affects a user’s capability to execute a file or view the contents of a directory.

Let’s see how it works.

File permission is in following format.

Owner Group Other/all

```
root@Net:~# ls -al
```

We will talk about aforementioned command later on in this chapter.

```
-rwxr-xr-x 1 net tut 77 Oct 24 11:51 auto run
drwx----- 2 ali tut 4096 Oct 25 2012 cache
```

File auto run permission

--No special permissions

rwx—Owner (net) having read, write, and execute permission while group (tut) having read and execute and other also having same permission.

File cache permission

d—Represent directory

rwx—Owner (ali) having read, write, and execute permission while group (tut) and other/all does not have any permission for accessing or reading this file.

Linux Advance/Special Permission

l—The file or directory is a symbolic link

s—This indicated the setuid/setgid permissions. Represented as a s in the read portion of the owner or group permissions.

- t—This indicates the sticky bit permissions. Represented as a t in the executable portion of the all users permissions
- i—chatter Making file unchangeable

There are two more which mostly used by devices.

- c—Character device
- b—Block device (i.e., hdd)

Let's go through some examples

Link Permission

```
root@net:~#ln -s new /root/link
root@net:~#ls -al
lrwxrwxrwx 1 ali ali 3 Mar 18 08:09 link -> new
link is created for a file name called new (link is symbolic for file name new)
```

Suid & Guid Permission

setuid (SUID)—This is used to grant root level access or permissions to users

When an executable is given *setuid* permissions, normal users can execute the file with root level or owner privileges. *Setuid* is commonly used to assign temporarily privileges to a user to accomplish a certain task. For example, changing a user's password would require higher privileges, and in this case, *setuid* can be used.

setgid (SGID)—This is similar to *setuid*, the only difference being that it's used in the context of a group, whereas *setuid* is used in the context of a user.

```
root@net:~#chmod u+s new
root@net:~#ls -al
-rwSr--r-- 1 ali ali 13 Mar 18 07:54 new
```

Capital **S** shows *Suid* for this file.

```
root@net:~#chmod g+s guid-demo
root@net:~#ls -al
-rw-r-Sr-- 1 ali ali 0 Mar 18 09:13 guid-demo
```

Capital **S** shows *Guid* for *guid-demo* file and capital **S** is in group section.

Stickybit Permission

This is another type of permission; it is mostly used on directories to prevent anyone other than the “root” or the “owner” from deleting the contents.

```
root@net:~#chmod +t new
root@net:~#ls -al
-rw-r--r-T 1 ali ali 13 Mar 18 07:54 new
```

Capital **T** shows that stickybit has been set for other user (only owner or root user can delete files)