

# Contents

<b>Foreword</b> . . . . .	<b>xxv</b>
<b>Part I Nessus Tools</b> . . . . .	<b>1</b>
<b>Chapter 1 The Inner Workings of NASL (Nessus Attack Scripting Language)</b> . . . . .	<b>3</b>
Introduction . . . . .	4
What Is NASL? . . . . .	4
Structure of a NASL Script . . . . .	4
The Description Section . . . . .	4
The Test Section . . . . .	6
Writing Your First Script . . . . .	7
Commonly Used Functions . . . . .	9
Regular Expressions in NASL . . . . .	11
String Manipulation . . . . .	12
How Strings Are Defined in NASL . . . . .	12
String Addition and Subtraction . . . . .	13
String Search and Replace . . . . .	13
Nessus Daemon Requirements to Load a NASL . . . . .	14
Final Touches . . . . .	14
<b>Chapter 2 Debugging NASLs</b> . . . . .	<b>15</b>
In This Toolbox . . . . .	16
How to Debug NASLs Using the Runtime Environment . . . . .	16
Validity of the Code . . . . .	16
Validity of the Vulnerability Test . . . . .	21
How to Debug NASLs Using the Nessus Daemon Environment . . . . .	28
Final Touches . . . . .	28

<b>Chapter 3 Extensions and Custom Tests</b> . . . . .	<b>29</b>
In This Toolbox . . . . .	30
Extending NASL Using Include Files . . . . .	30
Include Files . . . . .	30
Extending the Capabilities of Tests Using the Nessus Knowledge Base . . . . .	34
Extending the Capabilities of Tests Using Process Launching and Results Analysis . . . . .	35
What Can We Do with TRUSTED Functions? . . . . .	36
Creating a TRUSTED Test . . . . .	37
Final Touches . . . . .	42
<b>Chapter 4 Understanding the Extended Capabilities of the Nessus Environment</b> . . . . .	<b>43</b>
In This Toolbox . . . . .	44
Windows Testing Functionality Provided by the smb_nt.inc Include File . . . . .	44
Windows Testing Functionality Provided by the smb_hotfixes.inc Include File . . . . .	47
UNIX Testing Functionality Provided by the Local Testing Include Files . . . . .	50
Final Touches . . . . .	55
<b>Chapter 5 Analyzing GetFileVersion and MySQL Passwordless Test</b> . . . . .	<b>57</b>
In This Toolbox . . . . .	58
Integrating NTLM Authentication into Nessus' HTTP Authentication Mechanism . . . . .	58
NTLM . . . . .	58
Improving the MySQL Test by Utilizing Packet Dumps . . . . .	70
Improving Nessus' GetFileVersion Function by Creating a PE Header Parser . . . . .	79
Final Touches . . . . .	94
<b>Chapter 6 Automating the Creation of NASLs</b> . . . . .	<b>95</b>
In This Toolbox . . . . .	96
Plugin Templates: Making Many from Few . . . . .	96
Common Web Application Security Issues . . . . .	96

Server-Side Execution (SQL Injection, Code Inclusion) . . . . .	96
Client-Side Execution (Code Injection, Cross-Site Scripting, HTTP Response Splitting) . . . . .	98
Creating Web Application Plugin Templates . . . . .	99
Detecting Vulnerabilities . . . . .	100
Making the Plugin More General . . . . .	101
Parameterize the Detection and Trigger Strings . . . . .	101
Allow Different Installation dirs. . . . .	101
Allow Different HTTP Methods. . . . .	102
Multiple Attack Vectors. . . . .	103
Increasing Plugin Accuracy . . . . .	107
The “Why Bother” Checks. . . . .	107
Avoiding the Pitfalls . . . . .	108
The Final Plugin Template . . . . .	111
Rules of Thumb . . . . .	114
Using a CGI Module for Plugin Creation . . . . .	115
CGI . . . . .	115
Perl’s CGI Class . . . . .	115
Template .conf File . . . . .	116
Plugin Factory . . . . .	117
Final Setup . . . . .	124
Example Run . . . . .	124
Advanced Plugin Generation: XML Parsing for Plugin Creation . . . . .	126
XML Basics . . . . .	126
XML As a Data Holder. . . . .	127
Using mssecure.xml for Microsoft Security Bulletins . . . . .	128
The mssecure XML Schema . . . . .	128
The Plugin Template . . . . .	129
Ins and Outs of the Template. . . . .	130
Filling in the Template Manually . . . . .	132
General Bulletin Information . . . . .	132
The Finished Template . . . . .	134
The Command-Line Tool . . . . .	135
XML::Simple . . . . .	135

Tool Usage . . . . .	136
The Source . . . . .	138
Conclusion . . . . .	146
Final Touches . . . . .	147
<b>Part II Snort Tools . . . . .</b>	<b>149</b>
<b>Chapter 7 The Inner Workings of Snort . . . . .</b>	<b>151</b>
In This Toolbox . . . . .	152
Introduction . . . . .	152
Initialization . . . . .	154
Starting Up . . . . .	154
Libpcap . . . . .	158
Parsing the Configuration File . . . . .	159
ParsePreprocessor() . . . . .	160
ParseOutputPlugin() . . . . .	161
Snort Rules . . . . .	162
Event Queue Initialization . . . . .	168
Final Initialization . . . . .	168
Decoding . . . . .	168
Preprocessing . . . . .	172
Detection . . . . .	174
Content Matching . . . . .	175
The Stream4 Preprocessor . . . . .	176
Inline Functionality . . . . .	176
Inline Initialization . . . . .	176
Inline Detection . . . . .	178
Final Touches . . . . .	179
<b>Chapter 8 Snort Rules . . . . .</b>	<b>181</b>
In This Toolbox . . . . .	182
Writing Basic Rules . . . . .	182
The Rule Header . . . . .	182
Rule Options . . . . .	184
Metadata Options . . . . .	185
sid . . . . .	185
rev . . . . .	185
msg . . . . .	185

reference . . . . .	186
classtype . . . . .	186
priority . . . . .	188
Payload Options . . . . .	188
content . . . . .	188
offset . . . . .	188
depth . . . . .	189
distance . . . . .	189
within . . . . .	189
nocase . . . . .	190
rawbytes . . . . .	190
uricontent . . . . .	190
isdataat . . . . .	190
Nonpayload Options . . . . .	190
flags . . . . .	190
fragoffset . . . . .	191
fragbits . . . . .	191
ip_proto . . . . .	192
ttl . . . . .	192
tos . . . . .	192
id . . . . .	192
ipopts . . . . .	192
ack . . . . .	193
seq . . . . .	193
dsize . . . . .	193
window . . . . .	193
itype . . . . .	193
icode . . . . .	193
icmp_id . . . . .	193
icmp_seq . . . . .	194
rpc . . . . .	194
sameip . . . . .	194
Post-detection Options . . . . .	194
resp . . . . .	194
react . . . . .	195
logto . . . . .	195

## xviii Contents

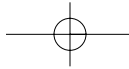
session . . . . .	195
tag . . . . .	196
Writing Advanced Rules . . . . .	196
PCRE . . . . .	196
Byte_test and Byte_jump . . . . .	205
byte_test. . . . .	205
byte_jump . . . . .	206
The Flow Options . . . . .	209
flow . . . . .	209
flowbits . . . . .	210
Activate and Dynamic Rules . . . . .	211
Optimizing Rules . . . . .	211
Ordering Detection Options . . . . .	211
Choosing between Content and PCRE . . . . .	212
Merging CIDR Subnets . . . . .	212
Optimizing Regular Expressions . . . . .	213
Testing Rules . . . . .	217
Final Touches . . . . .	219
<b>Chapter 9 Plugins and Preprocessors . . . . .</b>	<b>221</b>
In This Toolbox . . . . .	222
Introduction . . . . .	222
Writing Detection Plugins . . . . .	222
RFC 3514: The Evil Bit . . . . .	223
Detecting “Evil” Packets . . . . .	224
SetupEvilBit() . . . . .	225
EvilBitInit() . . . . .	226
ParseEvilBit() . . . . .	227
CheckEvilBit() . . . . .	228
Setting Up . . . . .	229
Testing . . . . .	230
Writing Preprocessors . . . . .	232
IP-ID Tricks . . . . .	233
Idle Scanning . . . . .	233
Predictable IP-ID Preprocessor . . . . .	235
SetupIPID() . . . . .	236
IPIDInit() . . . . .	236

IPIDParse()	237
RecordIPID()	238
Setting Up Prevention	241
Writing Output Plugins	242
GTK+	243
An Interface for Snort	244
Glade	244
Function Layout	248
AlertGTKSetup()	249
AlertGTKInit	249
AlertGTK	251
Exiting	251
Setting Up	253
Miscellaneous	254
Final Touches	254
<b>Chapter 10 Modifying Snort</b>	<b>255</b>
In This Toolbox	256
Introduction	256
Snort-AV	256
Active Verification	256
Snort-AV- Implementation Summary	257
Snort-AV Initialization	258
Snort.h	258
Snort.c	259
Parser.c	260
Signature.h	261
Detect.c	261
Snort-AV Event Generation	264
Snort-AV Event Verification	266
Setting Up	269
Snort-Wireless	269
Implementation	270
Preprocessors	272
Anti-Stumbler	272
Auth Flood	272

De-auth Flood . . . . .	272
Mac-Spoof . . . . .	272
Rogue-AP . . . . .	273
Detection Plugins . . . . .	273
Wifi Addr4 . . . . .	274
BSSID . . . . .	274
Duration ID . . . . .	274
Fragnum . . . . .	274
Frame Control . . . . .	274
From DS . . . . .	274
More Data . . . . .	274
More Frags . . . . .	275
Order . . . . .	275
Power Management . . . . .	275
Retry . . . . .	275
Seg Number . . . . .	275
SSID . . . . .	275
Stype . . . . .	275
To DS . . . . .	276
Type . . . . .	276
WEP . . . . .	276
Rules . . . . .	276
Final Touches . . . . .	276
<b>Part III Ethereal Tools . . . . .</b>	<b>277</b>
<b>Chapter 11 Capture File Formats . . . . .</b>	<b>279</b>
In This Toolbox . . . . .	280
Using libpcap . . . . .	280
Selecting an Interface . . . . .	280
Opening the Interface . . . . .	283
Capturing Packets . . . . .	284
Saving Packets to a File . . . . .	287
Using text2pcap . . . . .	289
text2pcap Hex Dumps . . . . .	289
Packet Metadata . . . . .	290
Converting Other Hex Dump Formats . . . . .	292
Extending Wiretap . . . . .	295

The Wiretap Library . . . . .	295
Reverse Engineering a Capture File Format . . . . .	296
Understanding Capture File Formats . . . . .	296
Finding Packets in the File . . . . .	298
Adding a Wiretap Module . . . . .	308
The module_open Function . . . . .	308
The module_read Function . . . . .	312
The module_seek_read Function . . . . .	318
The module_close Function . . . . .	322
Building Your Module . . . . .	322
Final Touches . . . . .	322
<b>Chapter 12 Protocol Dissectors . . . . .</b>	<b>323</b>
In This Toolbox . . . . .	324
Setting up a New Dissector . . . . .	324
Built-in versus Plugin . . . . .	324
Calling Your Dissector . . . . .	330
Calling a Dissector Directly . . . . .	331
Using a Lookup Table . . . . .	332
Examining Packet Data as a Last Resort . . . . .	333
New Link Layer Protocol . . . . .	334
Defining the Protocol . . . . .	334
Programming the Dissector . . . . .	340
Low-Level Data Structures . . . . .	340
Adding Column Data . . . . .	343
Creating proto_tree Data . . . . .	345
Calling the Next Protocol . . . . .	349
Advanced Dissector Concepts . . . . .	350
Exceptions . . . . .	350
User Preferences . . . . .	352
Final Touches . . . . .	356

<b>Chapter 13 Reporting from Ethereal</b> . . . . .	<b>357</b>
In This Toolbox . . . . .	358
Writing Line-Mode Tap Modules . . . . .	358
Adding a Tap to a Dissector . . . . .	358
Adding a Tap Module . . . . .	361
tap_reset . . . . .	366
tap_packet . . . . .	367
tap_draw . . . . .	370
Writing GUI Tap Modules . . . . .	371
Initializer . . . . .	374
The Three Tap Callbacks . . . . .	377
Processing Tethereal's Output . . . . .	380
XML/PDML . . . . .	388
The PDML Format . . . . .	390
Metadata Protocols . . . . .	393
EtherealXML.py . . . . .	395
Final Touches . . . . .	400
<b>Appendix A Host Integrity Monitoring Using Osiris and Samhain</b> . . . . .	<b>401</b>
Introducing Host Integrity Monitoring . . . . .	402
How Do HIM Systems Work? . . . . .	403
Scanning the Environment . . . . .	403
Centralized Management . . . . .	405
Feedback . . . . .	406
Introducing Osiris and Samhain . . . . .	406
Osiris . . . . .	407
How Osiris Works . . . . .	407
Authentication of Components . . . . .	408
Scan Data . . . . .	409
Logging . . . . .	410
Filtering Noise . . . . .	411
Notifications . . . . .	411
Strengths . . . . .	412
Weaknesses . . . . .	412
Samhain . . . . .	413
How Samhain Works . . . . .	413



Authentication of Components . . . . .	415
Scan Data . . . . .	415
Logging . . . . .	415
Notifications . . . . .	416
Strengths . . . . .	417
Weaknesses . . . . .	417
Extending Osiris and Samhain with Modules . . . . .	418
Osiris Modules . . . . .	418
An Example Module: mod_hostname . . . . .	419
Testing Your Module . . . . .	421
Packaging Your Module . . . . .	423
General Considerations . . . . .	423
Samhain Modules . . . . .	423
An Example Module: hostname . . . . .	424
Testing Your Module . . . . .	430
Packaging Your Module . . . . .	431
<b>Index . . . . .</b>	<b>433</b>

