

# Inhaltsverzeichnis

<b>1</b>	<b>Netzwerk-Grundlagen</b>	<b>1</b>
1.1	Netzwerkarchitekturen und -protokolle	1
1.2	Die Internet-Protokoll-Suite	3
1.3	Datenkapselung	5
1.3.1	Header, Footer und Adressen	5
1.3.2	Datenübertragung	7
1.4	Netzwerk-Routing	8
1.5	Mein Modell für die Analyse von Netzwerkprotokollen	9
1.6	Am Ende dieses Kapitels	12
<b>2</b>	<b>Capturing von Anwendungsverkehr</b>	<b>13</b>
2.1	Passives Capturing von Netzwerkverkehr	13
2.2	Eine kurze Einführung in Wireshark	14
2.3	Alternative passive Capturing-Techniken	16
2.3.1	Tracing von Systemaufrufen	17
2.3.2	Das strace-Utility unter Linux	18
2.3.3	Netzwerkverbindungen mit DTrace verfolgen	19
2.3.4	Process Monitor unter Windows	21
2.4	Vor- und Nachteile passiven Capturings	22
2.5	Aktives Capturing von Netzwerkverkehr	23
2.6	Netzwerk-Proxys	23
2.6.1	Port-Forwarding-Proxy	24
2.6.2	SOCKS-Proxy	28
2.6.3	HTTP-Proxys	33
2.6.4	Forwarding eines HTTP-Proxys	33
2.6.5	HTTP-Reverse-Proxy	37
2.7	Am Ende dieses Kapitels	40

<b>3</b>	<b>Strukturen von Netzwerk-Protokollen</b>	<b>41</b>
3.1	Binäre Protokollstrukturen	42
3.1.1	Numerische Daten	42
3.1.2	Boolesche Werte	45
3.1.3	Bit-Flags	46
3.1.4	Binäre Bytereihenfolge (Endianness)	46
3.1.5	Text und menschenlesbare Daten	47
3.1.6	Binärdaten variabler Länge	52
3.2	Datum und Uhrzeit	55
3.2.1	POSIX/Unix-Zeit	55
3.2.2	Windows FILETIME	55
3.3	TLV-Muster	56
3.4	Multiplexing und Fragmentierung	57
3.5	Netzwerk-Adressinformationen	58
3.6	Strukturierte Binärformate	59
3.7	Strukturen textbasierter Protokolle	60
3.7.1	Numerische Daten	61
3.7.2	Boolesche Werte in Textform	61
3.7.3	Datum und Uhrzeit	61
3.7.4	Daten variabler Länge	62
3.7.5	Formate für strukturierten Text	63
3.8	Codierung binärer Daten	65
3.8.1	Hex-Codierung	66
3.8.2	Base64	66
3.9	Am Ende dieses Kapitels	68
<b>4</b>	<b>Fortgeschrittenes Capturing von Anwendungsverkehr</b>	<b>69</b>
4.1	Rerouting von Verkehr	69
4.1.1	Traceroute nutzen	70
4.1.2	Routing-Tabellen	71
4.2	Konfiguration eines Routers	72
4.2.1	Routing unter Windows aktivieren	73
4.2.2	Routing unter *nix aktivieren	73
4.3	Network Address Translation	74
4.3.1	SNAT aktivieren	74
4.3.2	SNAT unter Linux konfigurieren	75
4.3.3	DNAT aktivieren	76

4.4	Verkehr an ein Gateway weiterleiten	78
4.4.1	DHCP-Spoofing	78
4.4.2	ARP-Poisoning	81
4.5	Am Ende dieses Kapitels	85
<b>5</b>	<b>Analyse auf der Datenleitung</b>	<b>87</b>
5.1	Die Verkehr produzierende Anwendung: SuperFunkyChat	87
5.1.1	Den Server starten	88
5.1.2	Clients starten	88
5.1.3	Kommunikation zwischen Clients	89
5.2	Ein Crashkurs zur Analyse mit Wireshark	90
5.2.1	Netzwerkverkehr generieren und Pakete erfassen	91
5.2.2	Grundlegende Analyse	93
5.2.3	Inhalte einer TCP-Session lesen	94
5.3	Die Paketstruktur mit Hex Dump identifizieren	95
5.3.1	Einzelne Pakete betrachten	96
5.3.2	Die Protokollstruktur ermitteln	97
5.3.3	Unsere Annahmen überprüfen	99
5.3.4	Das Protokoll mit Python sezieren	100
5.4	Einen Wireshark-Dissector in Lua entwickeln	106
5.4.1	Den Dissector entwickeln	109
5.4.2	Sezieren mit Lua	110
5.4.3	Parsen eines Nachrichtenpakets	111
5.5	Einen Proxy zur aktiven Verkehrsanalyse nutzen	114
5.5.1	Den Proxy einrichten	115
5.5.2	Protokollanalyse mittels Proxy	117
5.5.3	Grundlegendes Parsen von Protokollen hinzufügen	119
5.5.4	Das Protokollverhalten ändern	120
5.6	Am Ende dieses Kapitels	122
<b>6</b>	<b>Reverse Engineering einer Anwendung</b>	<b>123</b>
6.1	Compiler, Interpreter und Assembler	124
6.1.1	Interpretierte Sprachen	124
6.1.2	Kompilierte Sprachen	125
6.1.3	Statisches und dynamisches Linking	125
6.2	Die x86-Architektur	126
6.2.1	Instruction Set Architecture	127
6.2.2	CPU-Register	128
6.2.3	Ablaufsteuerung	131

6.3	Betriebssystem-Grundlagen	132
6.3.1	Dateiformate für Executables	132
6.3.2	Abschnitte	133
6.3.3	Prozesse und Threads	133
6.3.4	Netzwerkschnittstelle des Betriebssystems	134
6.3.5	Application Binary Interface	137
6.4	Statisches Reverse Engineering	138
6.4.1	Kurzanleitung für die Nutzung der IDA Pro Free Edition	139
6.4.2	Stackvariablen und Argumente analysieren	143
6.4.3	Schlüsselfunktionalitäten identifizieren	143
6.5	Dynamisches Reverse Engineering	150
6.5.1	Breakpunkte setzen	151
6.5.2	Debugger-Fenster	151
6.5.3	Wo setzt man Breakpunkte?	153
6.6	Reverse Engineering von Managed Code	153
6.6.1	.NET-Anwendungen	154
6.6.2	ILSpy nutzen	155
6.6.3	Java-Anwendungen	158
6.6.4	Mit Verschleierungstaktiken umgehen	160
6.7	Reverse-Engineering-Ressourcen	161
6.8	Am Ende dieses Kapitels	161
<b>7</b>	<b>Sicherheit von Netzwerkprotokollen</b>	<b>163</b>
7.1	Verschlüsselungsalgorithmen	164
7.1.1	Substitutionschiffre	165
7.1.2	XOR-Verschlüsselung	166
7.2	Zufallszahlengeneratoren	167
7.3	Symmetrische Verschlüsselung	168
7.3.1	Blockchiffre	168
7.3.2	Blockchiffre-Modi	171
7.3.3	Blockchiffre-Padding	174
7.3.4	Padding Oracle Attack	176
7.3.5	Stromchiffre	178
7.4	Asymmetrische Verschlüsselung	179
7.4.1	RSA-Algorithmus	180
7.4.2	RSA-Padding	182
7.4.3	Schlüsselaustausch nach Diffie-Hellman	182
7.5	Signaturalgorithmen	184
7.5.1	Kryptografische Hash-Algorithmen	185
7.5.2	Asymmetrische Signaturalgorithmen	186
7.5.3	Message Authentication Codes	187

7.6	Public-Key-Infrastruktur . . . . .	190
7.6.1	X.509-Zertifikate . . . . .	190
7.6.2	Verifikation einer Zertifikatskette . . . . .	192
7.7	Fallbeispiel: Transport Layer Security . . . . .	193
7.7.1	Der TLS-Handshake . . . . .	194
7.7.2	Initiale Aushandlungen . . . . .	195
7.7.3	Endpunkt-Authentifizierung . . . . .	195
7.7.4	Die Verschlüsselung aufbauen . . . . .	197
7.7.5	Sicherheitsanforderungen erfüllen . . . . .	198
7.8	Am Ende dieses Kapitels . . . . .	200
<b>8</b>	<b>Implementierung des Netzwerkprotokolls</b>	<b>201</b>
8.1	Replay von erfasstem Netzwerkverkehr . . . . .	201
8.1.1	Verkehr mit Netcat erfassen . . . . .	202
8.1.2	Replay von UDP-Verkehr mittels Python . . . . .	204
8.1.3	Unseren Analyse-Proxy wiederverwenden . . . . .	206
8.2	Ausführbaren Code wiederverwenden . . . . .	211
8.2.1	Code in .NET-Anwendungen wiederverwenden . . . . .	212
8.2.2	Code in Java-Anwendungen wiederverwenden . . . . .	217
8.2.3	Unmanaged Executables . . . . .	219
8.3	Verschlüsselung und der Umgang mit TLS . . . . .	224
8.3.1	Die verwendete Verschlüsselung ermitteln . . . . .	225
8.3.2	TLS-Verkehr entschlüsseln . . . . .	226
8.4	Am Ende dieses Kapitels . . . . .	232
<b>9</b>	<b>Die Hauptursachen für Sicherheitslücken</b>	<b>233</b>
9.1	Vulnerabilitätsklassen . . . . .	234
9.1.1	Remote Code Execution . . . . .	234
9.1.2	Denial-of-Service . . . . .	234
9.1.3	Offenlegung von Informationen . . . . .	235
9.1.4	Authentifizierung umgehen . . . . .	235
9.1.5	Autorisierung umgehen . . . . .	235
9.2	Verfälschung des Speichers . . . . .	236
9.2.1	Speichersichere und speicherunsichere Programmiersprachen . . . . .	236
9.2.2	Pufferüberlauf . . . . .	237
9.2.3	Out-of-Bounds-Indexierung . . . . .	242
9.2.4	Datenexpansion . . . . .	243
9.2.5	Fehler bei der dynamischen Speicherallokation . . . . .	244
9.3	Voreingestellte oder festcodierte Anmeldedaten . . . . .	244
9.4	Offenlegung von Benutzernamen . . . . .	245

9.5	Fehlerhafter Zugriff auf Ressourcen	247
9.5.1	Kanonisierung	247
9.5.2	Fehlermeldungen mit zu viel Information	249
9.6	Speicherüberlastung	250
9.7	Massenspeicherüberlastung	251
9.8	CPU-Überlastung	252
9.8.1	Algorithmische Komplexität	252
9.8.2	Konfigurierbare Kryptografie	254
9.9	Formatstrings	255
9.10	Befehlsinjektion	256
9.11	SQL-Injektion	257
9.12	Zeichenersetzung bei Textcodierung	258
9.13	Am Ende dieses Kapitels	259
<b>10</b>	<b>Sicherheitslücken aufspüren und ausnutzen</b>	<b>261</b>
10.1	Fuzzing	261
10.1.1	Der einfachste Fuzzing-Test	262
10.1.2	Mutations-Fuzzer	262
10.1.3	Testdatensätze generieren	263
10.2	Sicherheitslücken untersuchen	264
10.2.1	Debugging von Anwendungen	264
10.2.2	Die Chancen erhöhen, um die Hauptursache für einen Absturz zu ermitteln	271
10.3	Gängige Sicherheitslücken ausnutzen	274
10.3.1	Exploit von Speicherlücken	275
10.3.2	Willkürliche Schreiboperationen	283
10.4	Shell-Code entwickeln	286
10.4.1	Erste Schritte	286
10.4.2	Einfache Debugging-Technik	289
10.4.3	Systemaufrufe ausführen	290
10.4.4	Andere Programme ausführen	295
10.4.5	Shell-Code mit Metasploit generieren	296
10.5	Maßnahmen gegen Speicherlücken	298
10.5.1	Data Execution Prevention	298
10.5.2	Return-Oriented Programming	300
10.5.3	Address Space Layout Randomization (ASLR)	302
10.5.4	Stacküberläufe durch Canaries erkennen	305
10.6	Am Ende dieses Kapitels	309

## Anhang

<b>A</b>	<b>Toolkit für die Netzwerkprotokoll-Analyse</b>	<b>311</b>
A.1	Tools zum passiven Capturing und zur Analyse von Netzwerkprotokollen	311
A.1.1	Microsoft Message Analyzer	312
A.1.2	TCPDump und LibPCAP	313
A.1.3	Wireshark	314
A.2	Aktives Netzwerk-Capturing und Analyse	315
A.2.1	Canape	315
A.2.2	Canape Core	316
A.2.3	Mallory	316
A.3	Netzwerk-konnektivität und Protokolltests	316
A.3.1	Hping	316
A.3.2	Netcat	317
A.3.3	Nmap	317
A.4	Webanwendungen testen	318
A.4.1	Burp Suite	318
A.4.2	Zed Attack Proxy (ZAP)	319
A.4.3	Mitmproxy	319
A.5	Frameworks zum Fuzzing, zur Paketgenerierung und zur Entwicklung von Exploits	320
A.5.1	American Fuzzy Lop (AFL)	320
A.5.2	Kali Linux	321
A.5.3	Metasploit-Framework	321
A.5.4	Scapy	321
A.5.5	Sulley	322
A.6	Netzwerk-Spoofing und -Umleitung	322
A.6.1	DNSMasq	322
A.6.2	Ettercap	322
A.7	Reverse Engineering von Executables	323
A.7.1	Java Decompiler (JD)	323
A.7.2	IDA Pro	324
A.7.3	Hopper	325
A.7.4	ILSpy	325
A.7.5	.NET Reflector	326
	<b>Index</b>	<b>327</b>