INDEX

Symbols and Numbers
\ (backslash), 47, 220
/ (forward slash), 81, 220
- (minus sign), 55
+ (plus sign), 55
7-bit integer, 39–40
8-bit integer, 38–39
32-bit system, 263
32-bit value, 40–41
64-bit system, 263
64-bit value, 40–41
8086 CPU, 114

A
A5/1 stream cipher, 159
A5/2 stream cipher, 159
ABI (application binary interface), 123–124, 259–260
Abstract Syntax Notation 1 (ASN.1), 53–54
accept system call, 123
acknowledgment (DHCP packet), 72
acknowledgment flag (ACK), 41
active network capture, 20, 280–282.
See also passive network capture
add() function, 124
ADD instruction, 115
add_longs() method, 198
add_numbers() method, 197
Address Resolution Protocol (ARP), 6–7, 74–77
addresses, 4
  32-bit, 5
destination, 5
  MAC, 6–8, 74–77
  source, 5
address sanitizer, 243–244
address space layout randomization (ASLR)
bypassing with partial overwrites, 272–273
exploiting implementation flaws in, 271–272
memory information disclosure vulnerabilities, 270–271
Adleman, Leonard, 160
Advanced Encryption Standard (AES), 133, 150, 152
AJAX (Asynchronous JavaScript and XML), 57
algorithms
complexity of, 224–225
cryptographic hashing, 164–165
Diffie–Helman Key Exchange, 162–164
hash, 165
key-scheduling, 151
message digest (MD), 164
MD4, 165
MD5, 133, 165–167
RSA, 149, 160–162, 165
secure hashing algorithm (SHA), 164, 202
SHA-1, 133, 165–166
SHA-2, 165
SHA-3, 168
signature, 146
asymmetric, 165
cryptographic hashing algorithms, 164–165
message authentication codes, 166–168
symmetric, 166
AMD, 114
American Fuzzy Lop, 285–286
AND instruction, 115
antivirus, 23
application, 3
content parsers, 4
network communication, 4
passive network traffic capture, 11
user interface, 4
application binary interface (ABI), 123–124, 259–260
application layer, 3
apt command line utility, 31
arbitrary writing of memory, 253–254
ARM architecture, 42, 118
ARP poisoning, 74–77
ASCII
character encoding, 42
code pages, 44
control characters, 43
printable characters, 43
text-encoding conversions, 229–230
ASLR. See address space layout
randomization (ASLR)
ASN.1 (Abstract Syntax Notation 1), 53–54
assembler, 113, 258
assemblies, 138
assembly language, 113
assembly loading, 190–193
asymmetric key cryptography, 159–164.
See also symmetric key cryptography
private key, 160
public key, 160
RSA algorithm, 160–162
RSA padding, 162
trapdoor functions, 160
asymmetric signature algorithms, 165
Asynchronous JavaScript and XML
(AJAX), 57
AT&T syntax, 116
attributes (XML), 58
authentication bypass, 209
authorization bypass, 209–210
automated code, identifying, 133–134

B
backslash (\), 47, 220
base class library, 141
Base64, 60–61
Berkeley packet filter (BPF), 180
Berkeley Sockets Distribution (BSD), 15
Berkeley Sockets model, 15, 121
big endian, 42, 52, 122
Big-O notation, 225
binary conversions, 90–92
binary protocols. See also protocols
binary endian, 41–42
bit flags, 41
Booleans, 41
formats, 53–54
numeric data, 38–41
strings, 42–46
variable binary length data, 47–49
bind system call, 15
bit flags, 41
bit format, 38
block ciphers. See also stream ciphers
AES, 150, 152
common, 152
DES, 150–151
initialization vector, 154
modes, 152–155
cipher block chaining, 153–155
Electronic Code Book, 152
Galois Counter, 155
padding, 155–156
padding oracle attack, 156–158
Triple DES, 151
Blowfish, 152
Booleans, 41, 55
BPF (Berkeley packet filter), 180
breakpoints, 135, 137
BSD (Berkeley Sockets Distribution), 15
bss data, 120
Bubble Sort, 224
bucket, 225
buffer overflows
fixed-length, 211–213
heap, 248–249
integer, 214–215
stack, 246–248
variable-length, 211, 213–214
Burp Suite, 283–284
bytes, 38

C
C# language, 112, 189, 210
C++ language, 112, 132
cacert file, 203
CALL instruction, 115
Camellia, 152
Canape Core, 21–22, 25, 103–105, 280–281
Canape.Cli, xxiv, 202
CORBA (Common Object Request Broker Architecture), 22
counter mode, 159
CPU, 39
8086, 114
assembly language and, 113
exhaustion attacks, 224–226
instruction set architecture, 114–116
registers, 116–118
signed integers, 39
x86 architecture, 114–119, 125

Crashes
debugging, 238–240
example, 240–243
finding root cause of, 243–245
CreateInstance() method (.NET), 191
cron jobs, 254
cross-site scripting (XSS), 58
Crypt32.dll, 132
CryptoAllPermissionCollection.class, 142
cryptanalysis, 146
cryptography
asymmetric key, 159–164
configurable, 226
hashing algorithms, 164–165
libraries, 132
symmetric key, 149–159
CS register, 116, 118
ctypes library (Python), 195
curl command line utility, 31

data
controlling flow of, 2
capsulation, 4–7
endianness of, 41
formatting and encoding, 2
implicit-length, 48–49
inbound, 92
integrity, 164
numeric, 38–41
padded, 49
terminated, 47–48
transmission, 2, 6–7
variable-length, 56

Data Encryption Standard (DES), 150–151
data execution prevention (DEP), 267–268
data expansion attack, 217
DataFrame, 108
datagram, 5
datagram socket, 122
Datagram Transport Layer Security (DTLS), 172
data section, 120
dates, 49–50, 55
 .dll extension, 137–138
default or hardcoded credentials, 218
dehcup, 111, 134–137, 236–240, 243–245, 258–259
debugging, 236–243
analyzing crash in, 238–240
applications, 236
default or hardcoded credentials, 218
shell code, 258–259
starting, 236–237
debugging symbols package (dSYM), 131
dEC instruction, 115
decimal numbers, 55
decompilation, 113
decryption. See also encryption
asymmetric, 160
block cipher, 150
breakpoints, 137
cipher block chaining, 155, 157–158
dealing with obfuscation, 143–144
padding, 155–157
RSA, 161, 165
TLS, 200–202
Triple DES, 151
default credentials, 218
default gateway, 8, 66
defined memory pools, 252–253
delimited text, 56
denial-of-service, 208
DEP (data execution prevention), 267–268
DER (Distinguished Encoding Rules), 53
DES (Data Encryption Standard), 150–151
DES cracker, 151
destination address, 5
destination network address translation (DNAT), 24, 68–71
DHCP. See Dynamic Host Configuration Protocol (DHCP)
Diffie, Whitfield, 162
Diffie–Hellman Key Exchange (DH), 162–164
Digital Signature Algorithm (DSA), 165
disassembly, 113
discover (DHCP packet), 71
dissector() function, 99
dissector.lua file, 98
dissector
   creating, 97
   Lua, 99
   message packet parsing, 100–103
   Wireshark, 95–103
Distinguished Encoding Rules (DER), 53
DLL extension, 80, 120, 189
DNAT (destination network address translation), 24, 68–71
DNSMasq, 287
dnsspoof, 34
Domain Name System (DNS) protocol, 3
Dotfuscator, 143–144
dotnet binary, 81
downgrade attack, 176
DSA (Digital Signature Algorithm), 165
DS register, 116, 118
dSYM (debugging symbols package), 131
Dtrace, 16–18
Dynamic Host Configuration Protocol (DHCP), 63, 66
   packets, 71–72
   spoofing, 71–74
dynamic libraries, 130, 195–196
dynamic linking, 113–114, 121
dynamic reverse engineering
   breakpoints, 135, 137
   defined, 134
general purpose registers, 136

E

EAX register, 116, 123, 242, 258, 270
EBP register, 116–117, 124
EBX register, 116, 124
ECDH (Elliptic Curve Diffie–Hellman), 202
ECX register, 116, 124
EDI register, 116–117, 124
EDX register, 116, 123–124
EFALULT, 262
EFLAGS register, 117, 119, 136
EIP register, 116–117, 135
Electronic Frontier Foundation, 151
elements (XML), 58
ELF (Executable Linking Format), 120, 131, 144
Elliptic Curve Diffie–Hellman (ECDH), 202
elliptic curves, 160
encoding
   Base64, 60–61
   binary data, 59–61
   hex, 59–60
   percent, 60
encoding layer, 8–10
encryption, 20, 30. See also decryption
   AES, 133, 150, 152
   asymmetric, 160
   block cipher, 150
   breakpoints, 137
   cipher block chaining, 153–155
   DES, 150–151
   Electronic Code Book, 153
   HTTP connection to, 108
   key, 146
   libraries, 132
   magic constants, 133
   one-time pad, 148
   padding, 155
   public key. See asymmetric key cryptography
   RSA, 155, 161
   substitution ciphers, 147
   TLS, 175–176, 200–206
   Triple DES, 151
encryption libraries, 132
endianness, 41–42
errno, 262
errors
   codes, 262
   detecting and correcting, 2
   off-by-one, 213
   verbose, 221–222
ES register, 116, 118
ESI register, 116, 124
ESP register, 116–117, 124, 136, 270
eth0, 180
Ethernet, 3
   ARP poisoning, 74–75
   frame, 6, 8
   MAC addresses, 6, 74
   network routing, 7–8
Ethernet (continued)
passive network capture, 12–13
simple network, 6
Ettercap, 72–75, 287–288
executable codes
address space layout
  randomization, 272
file formats, 119–120
function calls in, 123
memory corruption and, 210, 246
partial overwrites, 272
repurposing, 188–199
  in .NET applications, 189–193
  in Java applications, 193–195
ROP gadgets, 269
system calls, 259
unmanaged, 195–199
executable file formats, 119–120, 137
Executable Linking Format (ELF), 120, 131, 144
.exe extension, 120, 137–138, 189
exit system call, 260–261
Extensible Markup Language (XML), 58
Extensible Messaging and Presence Protocol (XMPP), 58

F
false, 55
fd argument, 261
Federal Information Processing Standard (FIPS), 151
Feistel network, 151
File Transfer Protocol (FTP), 24, 28
FILETIME (Windows), 50
Financial Information Exchange (FIX) protocol, 56
finished packet, 176
fixed-length buffer overflows, 211–213
floating-point data, 40–41
Follow Stream button (Wireshark), 85
Follow TCP Stream view (Wireshark), 88–89
footers, 4–5
format string vulnerability, 227
forward slash (/), 81, 220
forwarding HTTP proxy. See also reverse HTTP proxy
  advantages and disadvantages of, 31
  redirecting traffic to, 30–31
  simple implementation of, 30–31
fragmentation, 51–52
FreeBSD, 16
FreeCAP, 27
free-list, 251
frequency analysis, 147
FS register, 116, 118
FTP (File Transfer Protocol), 24, 28
function monitors, 111
fuzz testing
  defined, 234
  mutation fuzzer, 235
  simplest, 234
  test cases, 235–236
  tools
    American Fuzzy Lop, 285–286
    Kali Linux, 286
    Metasploit, 286
    Scapy, 287
    Sulley, 287

G
Galois Counter Mode (GCM), 155
gateway
  configuring, 66–67
  ARP poisoning, 74–77
  DHCP spoofing, 71–74
default, 8, 66
  forwarding traffic to, 71–77
  hops, 65
  nodes, 64
  routing tables on, 65–66
GB2312, 44
GCC compiler, 196
GCM (Galois Counter Mode), 155
GDB (debugger), 236–241
General Public License, 14
general purpose registers, 116–117, 136
GET request, 8, 29
GetConstructor method (.NET), 191
getDeclaredConstructor() (Java), 195
GetMethod() method (.NET), 192–193
Google, 170, 176–177
GS register, 116, 118
guard pages, 245
GUI registry editor, 67
GVSP protocol, 182
gzip, 217
handshake, 172
hardcoded credentials, 218
hash table, 225
hashed message authentication codes (HMAC), 168–169
hashing algorithms
  collision resistance, 164
cryptographic, 164–165
  nonlinearity of, 164
  pre-image resistance, 164
secure, 164–165, 202
SHA-1, 133, 165–166
SHA-2, 165
SHA-3, 168
HEAD, 29
Header, 4–5
  C, 17, 262
  Ethernet, 6
  HTTP, 24, 32–34
  IP, 6
    system call number, 260
  TCP, 5, 87
  UDP, 5
heap buffer overflows, 248–249
heap implementations, 250–251
heap memory storage, 253
Hellman, Martin, 162
Hex Dump (Wireshark), 86–95
  determining protocol structure in, 88–89
  information columns in, 87
  viewing individual packets in, 87
hex editor, 125
hex encoding, 59–60
Hex Rays, 125
high privileges, 254–255
HMAC (hashed message authentication codes), 168–169
Hopper, 289–290
hops, 65
host header, 24, 32–33
host order, 42
hosts file, 23, 34
Hping, 282
HTTP (HyperText Transport Protocol), 3, 56
  host header, 24
network protocol analysis, 8–10
proxies. See also protocols
  forwarding, 29–31
  reverse, 32–35
IBM, 151
ICS (Internet Connection Sharing), 69
IDA Pro, 289
  analyzing stack variables and arguments in, 128
  analyzing strings in, 132
  debugger windows, 135–136
    EIP window, 135
    ESP window, 136
  disassembly window, 127–128
  extracting symbolic information in, 129–131
  free version, 125–128
  graph view, 126
  identifying automated code in, 133–134
  Imports window, 131–132
  main interface, 127
  viewing imported libraries in, 131–132
  windows, 126–127
IEEE format, 40–41
IEEE Standard for Floating-Point Arithmetic (IEEE 754), 40
ILSpy, 138, 290
  analyzing type in, 140–141
  main interface, 139
  Search window, 139
implicit-length data, 48–49
in-band method, 253
inbound bytes, 89–92
inbound data, 92
INC instruction, 115
incorrect resource access, 220–223
  canonicalization, 220–221
  verbose errors, 221–222
inet_pton, 122–123
information disclosure, 209
initialization vector, 154
inner padding block, 168
instruction set architecture (ISA), 114–116
integer overflows, 214–215
integers
  signed, 39
text protocols, 55
unsigned, 38
  variable-length, 39–40
Intel, 114
Intel syntax, 116
Internet Connection Sharing (ICS), 69
Internet layer, 3
Internet Protocol (IP), 2
Internet Protocol Suite (IPS)
  data encapsulation, 4–7
  data transmission, 6–7
  defined, 3
  layers, 3
  network routing, 7–8
interpreted languages, 112
interpreters, 112
Invoke() method (.NET), 192–193
IP (Internet Protocol), 2
IP address
  32-bit, 24
  ARP poisoning, 74–77
  data transmission, 6–7
  destination, 18, 22
  DNAT, 69–71
  DNS spoofing, 34
hosts file, 34
NAT, 68
network routing, 7–8
reverse shell, 266
SNAT, 68
SOCKS connection, 25
ipconfig command, 69
iptables command, 69
IPS. See Internet Protocol Suite (IPS)
IPv4, 3, 5, 24, 52, 122
IPv6, 3, 5–6, 25, 52, 67
ISA (instruction set architecture), 114–116

J
Japanese characters, 44
Java, 112, 210
  applications, 141–142
  reflection types, 194
  repurposing codes in, 193–195
Java archive (JAR), 141, 193–194
Java byte code, 137
Java Decompiler, 288
Java Runtime, 27
JavaScript, 252
JavaScript Object Notation (JSON), 57–58
Java TCP client, 27
Jcc instruction, 115
JD-GUI, 142
JMP instruction, 115, 119

K
Kali Linux, 286
kernel mode, 14
key-scheduling algorithm, 151
Korean characters, 44
Krypto Analyzer, 134

L
least significant bit (LSB), 38
length-extension attacks, 166–168
length-prefixed data, 48
lengths, 107
LibPCAP, 278–279
line feed, 56
line oriented protocols, 56
linking, 113–114
link layer, 3, 6
Linux, 120
  ASLR implementation flaws in, 272
  configuring SNAT on, 69
cron jobs, 254
debug symbols, 129
debugger, 236–241
dynamic libraries, 196
enabling routing on, 67
error codes, 262
executable file format, 131
loading library on, 197
SOCKS proxy, 27
strace, 16
little endian, 42, 122
LLDB (debugger), 236–241
Load() method (.NET), 190
LoadFrom() method (.NET), 190
local variables, corrupting, 274–275
localhost, 12
low-privileged file writes, 255
Lua, 95–103

M
MAC (Media Access Control)
  addresses, 6–7, 8, 74–77
machine code, 112–114, 120, 125
macOS, 16, 27–28, 120
debug symbols, 129
debugger, 236–241
dynamic libraries, 196
enabling routing on, 67
Mach-O format, 120, 131, 144
MACs. See message authentication codes (MACs)
magic constants, 132
mail application, 3
main thread, 121
Mallory, 281–282
malware, 23
man 2 syscall_name command, 16
managed languages
Java, 141–142
.NET applications, 137–141
reverse engineering, 137–144
man-in-the-middle proxy, 20, 201
masquerading, 68
master secret (TLS), 175
MD algorithm. See message digest (MD) algorithm
Media Access Control (MAC) addresses, 6–7, 8, 74–77
memory
arbitrary writing of, 253–254
heap memory storage, 253
information disclosure
vulnerabilities, 270–271
wasted, 250
memory canaries (cookies)
bypassing by corrupting local variables, 274–275
bypassing with stack buffer underflow, 275–276
detecting stack overflows with, 273–276
memory corruption. See also vulnerabilities
buffer overflows, 210–215
data expansion attack, 217
dynamic memory allocation failures, 217
exploit mitigations, 266–276
address space layout randomization, 270–273
data execution prevention, 266–267
return-oriented programming, 268–270
exploiting, 245–253
heap buffer overflows, 248–249
stack buffer overflows, 246–248
memory-safe vs. memory-unsafe languages, 210
off-by-one error, 213
out-of-bounds buffer indexing, 216–217
memory exhaustion attacks, 222–223
memory index registers, 117
memory sections, 120
memory-safe languages, 210
memory-unsafe languages, 210
Message Analyzer, 278
message authentication codes (MACs) collision attacks, 166–168
hashed, 168–169
length-extension attacks, 166–168
signature algorithms, 166–168
Message command, 101–102
message digest (MD) algorithm, 164
MD4, 165
MD5, 133, 165–167
message packet, 100–103
Metasploit, 286
accessing payloads, 265
advantages and disadvantages of, 265–266
executing payloads, 266
generating shell code with, 265–266
MethodInfo type (.NET), 192
Microsoft, 170
Microsoft Message Analyzer, 278
MIME (Multipurpose Internet Mail Extensions), 56–57
minus sign (−), 55
MIPS, 42, 137
Mitmproxy, 284–285
mnemonic instruction, 114
modulo arithmetic, 214
modulus, 161, 214
mono binary, 80
Mono Project, 137
most significant bit (MSB), 38
MOV instruction, 115
Mozilla Firefox, 26
MSCORLIB, 141
MS-DOS, 119
msfvenom tool, 265–266
multibyte character sets, 44
multiplexing, 51–52
Multipurpose Internet Mail Extensions (MIME), 56–57
multitasking, 120

namespace, 193
name-value pairs (XML), 58
namespace assembler, 256, 258, 263
NAT. See network address translation (NAT)
.NET applications
  base class library, 141
  file formats, 137–138
  ILSpy, 138–141
  reflection binding types, 192
  reflection types, 190
  repurposing codes in, 189–193
  repurposing executable codes in assembly loading, 190–193
  using Reflection APIs, 190
.NET Core, 80
.NET Reflector, 290–291
Netcat, 180–182, 234, 282
NetClientTemplate class, 184–185
netstat -r command, 65
Netwide Assembler, 256
network, 1
  connectivity and protocol testing tools
    Hping, 282
    Netcat, 282
    Nmap, 282–283
  monitoring connections with
    DTrace, 16–18
  proxies, 20–35
  routing, 7–8
  network address, 7, 20, 22, 52–53, 66, 71, 123
  network address translation (NAT), 68–71
  defined, 68
  destination, 24, 68
  source, 68–69
network communication, 4
  Berkeley Sockets model, 15
  layers, 3
  man-in-the-middle attack on, 20
  symmetric ciphers, 150
  user-to-kernel, 15
network interface, 121–124
  client connection to TCP server, 122
  TCP client connection to server, 121–122
Network News Transfer Protocol (NNTP), 59
network order, 42
newInstance() method (Java), 195
Nmap, 282–283
NNTP (Network News Transfer Protocol), 59
nodes, 1
  gateway, 64
  identifying through addressing, 2
  no-execute (NX) mitigation, 267
  nonlinearity, 165
  nonpersistent denial-of-service, 208
  NULL, 263–264
numeric data
  decimal numbers, 55
  floating-point data, 40–41
  integers, 55
  signed integers, 39
  text protocols, 55
  unsigned integers, 38
  variable-length integers, 39–40
  NX (no-execute) mitigation, 267

O
OAEP (Optimal Asymmetric Encryption Padding), 162
obfuscation, 143–144
octets, 38–40
octet-stream, 57
off-by-one error, 213
offer (DHCP packet), 71
one-time pad encryption, 148
open system call, 18
OpenSSL, 132
operands, 115
operating system
  application binary interface, 123–124
  executable file formats, 119–120
  networking interface, 121–124
  processes, 120–121
  sections, 120
  threads, 120–121
Optimal Asymmetric Encryption Padding (OAEP), 162
OR instruction, 115
outbound bytes, 89
outbound traffic, 89
outer padding block, 168
out-of-band method, 253
out-of-bounds buffer indexing, 216–217
output feedback mode, 159
overflow flag, 117
package-private scoped classes, 193

packets, 6
  calculating checksum of, 93–94
  capturing, 83–84
  finding, 87–88
  identifying structure with Hex Dump, 86–95
  sniffing, 12–14
  viewing, 87–88

packing tools, 134

padded data, 49

padding
  block ciphers, 155–156
  decryption, 155–157
  encryption, 155
  inner block, 168
  OAEP, 162
  oracle attack, 156–158
  outer block, 168
  RSA encryption, 155, 162

Page Heap, 244–245

parity flag, 117

Parser class, 106, 185

parser.csx script, 183–184

parsing
  binary conversion and, 90
  decimal numbers and, 55
  endianness of data and, 41
  HTTP header, 33
  message command, 101–102
  message packet, 100–103
  mutation fuzzer and, 235
  protocol, 107–108
  Python script for, 91
  traffic, 183
  URL, 230
  variable-length integers, 40

partial overwrites, 272–273

passive network capture
  advantages and disadvantages of, 19–20
  Dtrace, 16–18
  packet sniffing, 12–14
  Process Monitor tool, 17–18
  strace, 16
  system call tracing, 14–16

  tools
    LibPCAP, 278–279
    Microsoft Message Analyzer, 278
    TCPDump, 278–279
    Wireshark, 12–13, 279–280

path, 220

$pc, 239

PDB (program database) file, 129–131

PDP-11, 42

PDU (protocol data unit), 4

PE (Portable Executable) format, 120, 134, 144

PEiD, 134

PEM format, 202

percent encoding, 60

perfect forward secrecy, 177

permutation boxes (P-Box), 152

persistent denial-of-service, 208

PGP (Pretty Good Privacy), 169

PHP, 255

PKI. See public key infrastructure (PKI)

plain, 57

plaintext, 146

plus sign (+), 54

Point-to-Point Protocol (PPP), 3

POP3 (Post Office Protocol 3), 4

POP instruction, 115

port, 2

port numbers, 5

Portable Executable (PE) format, 120, 134, 144

port-forwarding proxy. See also proxies
  advantages and disadvantages of, 23–24
    binding to network addresses, 22
    redirecting traffic to, 22–23
    simple implementation of, 21–22

POSIX, 15

POSIX/Unix time, 50

POST, 29

Post Office Protocol 3 (POP3), 4

PowerPC, 38

PPP (Point-to-Point Protocol), 3

Practical Packet Analysis, 14

pre-image resistance (hashing algorithm), 165

pre-master secret (TLS), 175

Pretty Good Privacy (PGP), 169

printable characters (ASCII), 43

printf function, 227

private Connect() method (.NET), 192

private exponent, 161

private key, 161, 165

PRNGs (pseudorandom number generators), 149

Process() method, 275–276

Process Monitor tool, 17–18
processes, 120–121
processor architectures, 42
program database (PDB) file, 129–131
program flow, 118–119
ProGuard, 143–144
promiscuous mode, 12
PROT_EXEC flag, 257
protocol data unit (PDU), 4
protocol stack, 3
protocols
  analysis, 8–10, 105–106
  binary, 38–49
  changing behavior of, 108–109
  checksum, 93–94
  dates, 49–50
  determining structure of, 88–89
  fragmentation, 51–52
  functions of, 2
  multiplexing, 51–52
  network address, 52–53
  network connectivity and protocol testing
    Hping, 282
    Netcat, 282
    Nmap, 282–283
  parsing, 107–108
  security, 145–178
  structured binary formats, 53–54
  tag, length, value (TLV) pattern, 50–51
  text, 54–58
  times, 49–50
  unknown parts, 93
proxies
  HTTP, 29–35
  man-in-the-middle, 20
  port-forwarding, 21–24
  protocol analysis with, 105–106
  setting up, 103–105
  SOCKS, 24–29, 103
  traffic analysis with, 103–110
  Proxifier, 27
pseudo registers, 239
pseudorandom number generators (PRNGs), 149
public Connect() method (.NET), 192
public exponent, 161
public key, 160–161, 165
Public Key Cryptography Standard #1.5, 162
Public Key Cryptography Standard #7 (PKCS#7), 155–156
public key encryption. See asymmetric key cryptography
public key infrastructure (PKI), 169–172
  certificate chain verification, 170–172
  defined, 169
  web of trust, 169
  X.509 certificates, 169–170
PublicClass class, 189
PublicMethod() method, 189
PUSH instruction, 115
Python, 210
  binary conversions, 90–92
  calling functions with, 199
  ctypes library, 195
  data types, 198
  dissecting protocol with, 90–95
  loading library with, 197
  resending captured UDP traffic with, 182–183
  struct library, 90–92
Q
  quoted string, 47–48
R
  rand() function, 149
  random number generators, 149
  RAX register, 257–260
  RC4 stream cipher, 176
  RDP (Remote Desktop Protocol), 51
  read system call, 15, 18, 122
  read_bytes() function, 91
  ReadData() function, 108
  ReadOutbound() function, 109
  Real Time Messaging Protocol (RTMP), 29
  Receive() method (.NET), 193
  recv system call, 15, 122–123
  recvfrom system call, 15
  reflection, 189
registers
  control, 117
  CS, 116, 118
  DS, 116, 118
  EAX, 116, 123, 242, 258, 270
  EBP, 116–117, 124
  EBX, 116, 124
  ECX, 116, 124
EDI, 116–117, 124
EDX, 116, 123–124
EFLAGS, 117, 119, 136
EIP, 116–117, 135
ES, 116, 118
ESI, 116, 124
ESP, 116–117, 124, 136, 270
FS, 116, 118
general purpose, 116–117, 136
GS, 116, 118
memory index, 117
pseudo, 239
RAX, 257–260
scratch, 123
selector, 118
SS, 116
x86 architecture, 116–118
remote code execution, 208
Remote Desktop Protocol (RDP), 51
Remote Method Invocation (RMI), 29
Remote Procedure Call (RPC), 22
request (DHCP packet), 72
Request for Comments (RFCs), 42, 56–57
request line, 30
rerouting traffic, 64–66
RESP field, 25
RET instruction, 115
Ret2Libc, 269
RETN instruction, 115
return-oriented programming (ROP), 268–270
reverse engineering
dynamic, 134–137
managed languages, 137–144
obfuscation, 143–144
resources, 144
static, 125–134
tools
  Hopper, 289–290
  IDA Pro, 289
  ILSpy, 290
  Java Decompiler, 288
  .NET Reflector, 290–291
reverse HTTP proxy. See also forwarding HTTP proxy
advantages and disadvantages of, 35
host header, 32–33
redirecting traffic to, 34
simple implementation of, 33
reverse shell, 266
Rich Site Summary (RSS), 58
Rijndael, 152
Rivest, Ron, 160
RMI (Remote Method Invocation), 29
root certificate, 170
ROP (return-oriented programming), 268–270
route print command (Windows), 65
router, 7–8
  ARP poisoning, 75–77
  configuring, 66–67
  defined, 64
  enabling DNAT, 70
  enabling SNAT, 68–69
routing
  on Linux, 67
  on macOS, 67
  on Windows, 66
routing table, 8, 65–66
RPC (Remote Procedure Call), 22
RSA encryption, 149
  algorithm, 160–162
  padding, 155, 162
  signature algorithm, 165
RSS (Rich Site Summary), 58
Ruby, 210
Run() function, 187
runtime, 137

S
say_hello() method, 197
say_string() method, 197
say_struct() function, 199
Scan for Hosts (Ettercap), 76
Scapy, 287
scratch registers, 123
scripting languages, 112
sections (memory), 120
secure hashing algorithm (SHA), 164
  SHA-1, 133, 165–166
  SHA-2, 165
  SHA-3, 168
Secure Sockets Layer (SSL). See Transport Layer Security (TLS)
security, 145–178
  encryption, 146–149
  public key infrastructure (PKI), 169–172
  random number generators, 149
  requirements, 145–146
  signature algorithms, 164–169
security (continued)
  symmetric key cryptography, 149–159
Transport Layer Security, 172–177
segment, 5, 87
SELECT statement, 229
selector registers, 118
self-signed certificate, 170
Send() method (.NET), 192–193
send system call, 15, 122–123
sendfrom system call, 15
Serpent, 152
server random (TLS), 173
session key, 162
session state, 2
set detach-on-fork off command, 237
setAccessible() (Java), 195
SGML (Standard Generalized Markup Language), 58
SHA. See secure hashing algorithm (SHA)
Shamir, Adi, 160
shared key, 163
shell code
  accessing payloads, 265
debugging technique, 258–259
generating with Metasploit, 265–266
relative address on 32- and 64-bit systems, 263
reverse shell, 266
setting breakpoint on, 258–259
system calls, 259
  exit, 260–261
  write, 261–263
writing, 255–266
shell_bind_tcp, 265
Shift-JIS, 44
SHL instruction, 115, 119
SHR instruction, 115
sign flag, 117
signature algorithms, 146, 164–169
  asymmetric, 165
cryptographic hashing algorithms, 164–165
  DSA, 165
message authentication codes, 166–168
  RSA, 165
symmetric, 166
signed integers, 39
simple checksum, 93–94
Simple Mail Transport Protocol (SMTP), 3–4, 56, 59
Simple Network Management Protocol (SNMP), 53
sketches, 150
sniffing, 12–14, 73
sockaddr_in structure, 17, 122
socket system call, 15
SOCKS proxy, 103. See also proxies
  advantages and disadvantages of, 28–29
    Firefox proxy configuration, 26
    Java TCP client, 27
    overview, 24
    redirecting traffic to, 26–27
    simple implementation of, 25–26
    versions, 24–25
socksProxyHost system property, 27
socksProxyPort system property, 27
SOH (Start of Header), 56
Solaris, 16, 120
source address, 5
source code, 112
source network address translation (SNAT)
  configuring on Linux, 69
  enabling, 68–69
$sp, 239
SPARC architecture, 42, 118, 137
spoofing
  DHCP, 71–74
  DNS, 34
tools, 287–288
sprintf string function, 212
SQL. See Structured Query Language (SQL)
SS register, 116
stack buffer overflows, 246–248, 273–276
stack buffer underflow, 275–276
stack trace, 239–240
stack variables, 128
Standard Generalized Markup Language (SGML), 58
start address, 120
Start of Header (SOH), 56
static linking, 113–114
static reverse engineering, 125–134. See also reverse engineering
  analyzing strings in, 133
  extracting symbolic information in, 129–131
identifying key functionality in, 129–134
stack variables and arguments, 128

`stdcall`, 199

storage exhaustion attacks, 223–224

`strcat` string function, 212

`strcpy` string function, 212

`strcpy_s` string function, 212

stream ciphers, 158–159. See also block ciphers

strings, 42–46
  analyzing, 132
  ASCII standard, 42–44
Strip tool, 131

`struct` library (Python), 90

Structure class, 199

structured binary formats, 53–54

Structured Query Language (SQL) injection, 228–229

Server, 229

structured text formats, 56–58

`SUB` instruction, 115

subroutine calling, 118–119

substitution boxes (S-Box), 152

substitution ciphers, 147

substitution-permutation network, 152

Sulley, 287

SuperFunkyChat
  analysis proxy
    captured traffic, 183–187
    simple network client, 184–186
    simple server, 186–188
  ChatClient, 81, 83–84, 106, 200
  ChatServer, 80, 106
  commands, 81
  communicating between clients, 81
dissector, 95–103
  parser code for, 107
  starting clients, 80–81
  starting the server, 80
  UDP mode, 97

switch device, 6

symbolic information, 129–131

symmetric key cryptography, 149.
  See also asymmetric key cryptography
  block ciphers, 150–159
  stream ciphers, 158–159
  symmetric signature algorithms, 166
  synchronize flag (SYN), 41

system API, 268

System assembly, 141

system calls
  accept, 123
  bind, 15
  connect, 15
  exit, 260–261
  open, 18
  read, 15, 18, 122
  recv, 15, 122–123
  recvfrom, 15
  send, 15, 122–123
  sendfrom, 15
  shell code, 259–262
  socket, 15
  tracing, 14–19
  Unix-like systems, 15–16, 122
  write, 15, 18, 122, 261–263

system function, 228

System.Activator class (.NET), 191

System.Reflection.Assembly class (.NET), 190

System.Reflection.ConstructorInfo class (.NET), 190

System.Reflection.FieldInfo class (.NET), 190

System.Reflection.MethodInfo class (.NET), 190

System.Reflection.PropertyInfo class (.NET), 190

System.Type class (.NET), 190

T
tag, length, value (TLV) pattern, 50–51, 89, 94–95

TCP. See Transmission Control Protocol (TCP)

TCPDump, 278–279

TCP/IP, 2, 9–10, 121, 262

`TCP/IP Guide`, 16

TcpNetworkListener (ILSpy), 140

terminated data, 47–48
terminated text, 56

`TEST` instruction, 115, 119
testy virtual buffer (TVB), 99
text protocols, 54
  Booleans, 55
dates, 55
numeric data, 55
structured text formats, 56–58
text protocols (continued)
  times, 55
  variable-length data, 55

variable-length data, 55

Text-encoding character replacement, 229–231

threads, 120–121

times, 49–50, 55

TLS. See Transport Layer Security (TLS)

*TLS Record* protocol, 172

TLV (tag, length, value) pattern, 50–51, 89, 94–95

*ToDataString()* method, 186

Token, 56

tools
  for active network capture and analysis
  Canape, 280–281
  Canape Core, 281
  Mallory, 281–282

fuzz testing
  American Fuzzy Lop, 285–286
  Kali Linux, 286
  Metasploit, 286
  Scapy, 286
  Sulley, 286

network connectivity and protocol testing
  Hping, 282
  Netcat, 282
  Nmap, 282–283

for network spoofing and redirection
  DNSMasq, 287
  Ettercap, 287–288

for passive network capture and analysis
  LibPCAP, 278–279
  Microsoft Message Analyzer, 278
  TCPDump, 278–279

reverse engineering
  Hopper, 289–290
  IDA Pro, 289
  IDA Pro, 289
  Java Decompiler, 288
  .NET Reflector, 290–291

for web application testing
  Burp Suite, 283–284
  Mitmproxy, 284–285
  Zed Attack Proxy, 284

traffic
  analysis using proxy, 103
  capturing
    active method, 20
    HTTP, 29–35
    man-in-the-middle, 20
    passive method, 12–20
    port-forwarding, 21–24
    proxies, 20–35
    SOCKS, 24–29
    system call tracing, 14–19
  capturing tools
    Dtrace, 17–18
    Netcat, 180–182
    Process Monitor tool, 18–19
    strace, 16
    generating, 83–84
    outbound, 89

Transmission Control Protocol (TCP), 2–3, 21
  bit flags, 41
  client connection to server, 121–123
  header, 5, 87
  HTTP proxy, 30
  packets, 87–88
  port numbers, 5
  port-forwarding proxy, 21–22, 201
  reading contents of sessions, 85–86
  reverse shell, 265–266
  SOCKS proxy, 24–28
  stream, 13–14

transport layer, 3, 6, 8–10

Transport Layer Security (TLS)
  certificate pinning, 177
  client certificate, 175
  decryption, 201–202
  encryption, 175–176, 200–201
  endpoint authentication, 174–175
  forcing TLS 1.2, 202
  handshake, 172–173
  initial negotiation, 173
  perfect forward secrecy, 177
  replacing certificate in, 202–206
  security requirements, 176–177
  *TLS Record* protocol, 172

trapdoor functions, 160

Triple DES, 151

true, 55

trusted root certification authorities, 204

Tshark, 180–182

TVB (testy virtual buffer), 99
Twofish, 152
two’s complement, 39

U
UCS (Universal Character Set), 44–45
UDP. See User Datagram Protocol (UDP)
UI (user interface), 4
uname command, 263–264
Unicode
character encoding, 44–45
character mapping, 44–45
UCS-2/UTF-16, 45
UCS-4/UTF-32, 45
Unicode Transformation Format (UTF), 44–45
Unified Sniffing mode (Ettercap), 76
Uniform Request Identifier (URI), 30, 32
uninitialized data, 120
Universal Character Set (UCS), 44–45
Unix-like systems, 5
  ASLR implementation flaws in, 272
  AT&T syntax, 116
  command injection, 228
  command line utilities on, 31
  configuring DNAT on, 70
  Dtrace, 16
  enabling routing on, 67
  error codes, 262
  executable format, 120
  hosts file, 23
  read and write calls, 122
  routing tables on, 65
  system calls, 15–16, 122
  traceroute, 64
Unk2 value, 93–95
unmanaged executables, 195–199
  dynamic libraries, 195–196
unsafe keyword, 210
unsigned integers, 38
UPX, 134
URI (Uniform Request Identifier), 30, 32
User Datagram Protocol (UDP), 3
  captured traffic, 182–183
dissectors, 98–99
  payload and header, 5
  port forwading, 21
  socket, 122
user interface (UI), 4
user mode, 14
user-after-free vulnerability, 249–250
UTF (Unicode Transformation Format), 44–45
UTF-8, 45–46

V
variable binary length data
  implicit-length data, 48–49
  length-prefixed data, 48
  padded data, 49
  terminated data, 47–48
variable-length buffer overflows, 211, 213–214
variable-length data, 56
variable-length integers, 39–40
verbose errors, 221–222
Verisign, 170
virtual function table, 242, 248–249
virtual hosts, 24
virtual machine, 137
VirtualAlloc, 250
Visual C++, 129
vulnerabilities
  authentication checking, 226
  classes
    authentication bypass, 209
    authorization bypass, 209–210
    denial-of-service, 208
    information disclosure, 209
    remote code execution, 208
    command injection, 228
  CPU exhaustion attacks
    algorithmic complexity, 224–225
    configurable cryptography, 224–225
  default or hardcoded credentials, 218
  exploiting
    arbitrary writing of memory, 253–254
    defined memory pool allocations, 252–253
    heap layout manipulation, 249–250
    heap memory storage, 253
    high-privileged file writes, 254–256
    low-privileged file writes, 255
memory corruption, 245–253
user-after-free vulnerability, 249–250
format string, 227
fuzz testing, 234–236
incorrect resource access
canonicalization, 220–221
verbose errors, 221–222
memory corruption
buffer overflows, 210–215
data expansion attack, 217
dynamic memory allocation
failures, 217
exploit mitigations, 267–268
memory-safe vs. memory-unsafe
languages, 210
out-of-bounds buffer indexing,
216–217
memory exhaustion attacks,
222–223
shell code, 255–266
SQL injection, 228–229
storage exhaustion attacks,
223–224
text-encoding character
replacement, 229–231
triaging, 236–245
user enumeration, 218–219

W
W3C, 58
web application testing tools, 283–285
Burp Suite, 283–284
Mitmproxy, 284–285
Zed Attack Proxy, 284
web of trust (WOT), 169
wget, 31
windll, 199
Windows
ASLR implementation flaws in, 272
calling functions with Python
on, 199
certificate manager, 203
debug symbols, 129
debugger, 236–241, 244–245
dynamic link libraries, 196
enabling routing on, 67
FILETIME, 50
loading library on, 197
Page Heap, 244–245
registry, 67
Winsock library, 121
XP SP2, 270
WinDump, 278
WinPcap, 278
Winsock, 121
Wireshark, 12–14, 81, 279–280
basic analysis, 84–85
capture interfaces dialog, 82–83
Conversations window, 84–85
dissectors, 95–103
generating network traffic in,
83–84
Hex Dump view, 86–95
main window, 82
reading contents of TCP sessions
in, 85–86
Tshark command line version,
180–182
WOT (web of trust), 169
write system call, 15, 18, 122, 261–263
writeData() function, 108
writePackets() method, 22
ws2_32.dll Windows network library,
130–131

X
X.509 certificates, 53–54, 169–171, 173
X.680 series, 53
x86 architecture, 42, 125
history, 114
instruction mnemonics, 115
instruction set architecture,
114–116
mnemonic forms, 115
program flow, 118–119
registers, 116–118
xcalc, 228
XML Schema, 58
XOR encryption, 108–109, 148–149,
153–154
XOR instruction, 115
XOR parameter, 108–109
xp_cmdshell function, 229
xxd tool, 90, 181

Z
Zed Attack Proxy (ZAP), 284
zero flag, 117
ZLib compression library, 132