

# INDEX

## A

- A records, 104, 109–111
- Abstract Syntax Notation One (ASN.1)
  - encoding, 133–135, 137–138
- acme/autocert, 235
- Add(*int*), 27
- Address Resolution Protocol (ARP)
  - poisoning, 178
- Advanced Encryption Standard (AES)
  - algorithm, 242
- ancillary chunks, 302
- anonymous functions, 126
- API interaction
  - overview, 51–53
  - Bing scraping, 68–76
  - Metasploit, 59–68
  - Shodan, 51–59
- APIInfo struct, 55
- append() function, 11
- ARP (Address Resolution Protocol)
  - poisoning, 178
- ASN.1 (Abstract Syntax Notation One)
  - encoding, 133–135, 137–138
- assembly, 216
- asymmetric algorithms, 234
- asymmetric cryptography, 245. *See also* encryption
- Atom, GitHub, 4–5
- authentication, 67, 86–88, 239–241

## B

- backticks, 19
- base workspace directory, 2
- Base64 encoding, 215–216
- bcrypt hashing, 235, 237–239
- Beacon, 121
- Berkeley Packet Filter (BPF), 175, 181.  
*See also* tcpdump
- best practices
  - coding, 19, 49, 66, 185, 195, 329
  - security, 96, 236

- bin directory, 2
- binaries, 2
- binary data handling, 213–216
- Bing, 68–76
- bodyType parameter, 46
- braces, 14
- break statements, 14
- brute force, 252–261
- buffer overflow fuzzing, 188–192
- buffered channels, 29, 37–39
- bufio package, 38, 112–113, 197
- build command, 7
- build constraints, 7–8
- byte slices, 19
- bytes package, 197

## C

- C, 201–212, 290–293
- C transform, 213
- Caddy Server, 127
- .Call() method, 273
- canonical name (CNAME) records,  
109–111
- capture() function, 184
- CGO package, 291
- channels, 16–17
- Checker interface, 220–222
- Cipher Block Chaining (CBC)
  - mode, 242
- ciphertext, 234
- cleartext
  - overview, 234
  - passwords, 150
  - sniffing, 178–180
- client implants, 323–325, 327–329
- Client struct, 53–54
- cloned sites, 90–93
- Close() method, 25
- closed ports, 22
- Cmd, 41
- CNAME records, 109–111
- Cobalt Strike, 118–124, 278

- COFF File Header, 282–283
- collision, 234
- Command() function, 41
- commands
  - build command, 7
  - cross-compiling, 7–8
  - go commands, 6–9
  - set command, 3
- complex data types, 10–11
- concurrency, 16–17, 37
- concurrent scanning, 26–32
- Conn, 35–38
- connections, 24–25, 35, 327
- constraints, 7–8
- control structures, 14–16
- convenience functions, 46–47, 140
- Copy() function, 40
- createChunkCRC() method, 304–305
- CreateRemoteThread() Windows
  - function, 275–276
- credential-harvesting attacks, 90–93
- critical chunks, 302
- cross-compiling, 7–8
- cross-site scripting, 94
- crypto package, 197, 235
- cryptography
  - overview, 234–235
  - hashing, 234–239
- curl, 40, 79

**D**

- Data Directory, 285–287
- data mapping, 71–73, 125
- data types
  - channels, 16
  - maps, 11
  - primitive, 10–11
  - slices, 11
- database miners, 161–170
- debug package, 197
- decoder function, 300
- decoding process, 308
- decryption, 234. *See also* encryption
- DefaultServerMux, 78–79
- defer, 49
- DELETE requests, 47–48
- dep tool, 9
- development environment set up, 1–10
- Dial() method, 24
- dialects, 132–133
- directives, 19

- Dirty COW, 201–204
- DNS clients, 104–117
- DNS proxies, 124–127
- DNS servers, 117–129
- DNS tunneling, 121
- do loops, 15
- Docker, 90, 118–122, 154–158
- document metadata, 69
- Document Object Model (DOM), 74
- domain fronting, 98
- DOS Header, 281
- DWORD, 271

## E

- echo servers, 32, 35–37
- Empire, 121
- Encode() method, 65
- encodeDecode() function, 308
- encoding package, 197
- encoding process, 308
- encryption, 234, 242–252
- endianness function, 299
- error handling, 17–18
- error messages, 51
- Exclusive OR (XOR), 307–312
- Executable and Linkable Format (ELF), 203
- exploitation, 196–212
- export address table (EAT), 279

## F

- field tags, 19–20, 139
- filesystems, 170–171
- filetype filter, 73
- filtered ports, 22
- filtering search results, 73–76
- firewalls, 22–23
- fixed field tag, 140
- Flusher, 42
- fmt package, 25
- FOCA, 69
- Foo struct, 19
- for loop, 15
- formatting
  - data, 38, 113–114
  - source code, 9
- Frida, 278
- fully qualified domain name (FQDN), 104
- fuzzing, 188–196

## G

- gaping security holes, 41
- Get() function, 46
- get() HTTP function, 227–229
- GetLoadLibAddress() function, 275
- GetProcessAddress() Windows function, 275
- getRegex() function, 163
- GetSchema() function, 163, 165
- Gieben, Miek, 104
- GitHub Atom, 4–5
- GNU Compiler Collection (GCC), 290
- go build command, 6–7
- Go DNS package, 104
- go doc command, 8
- go fmt command, 9
- go get command, 8–9
- Go Playground execution environment, 10
- go run command, 6
- Go Syntax
  - complex data types, 10–11
  - concurrency, 16–17
  - control structures, 14–16
  - data types, 10–11
  - interface types, 13
  - maps, 11
  - patterns, 12–14
  - pointers, 12
  - primitive data types, 10–11
  - slices, 11
  - struct types, 12–13
- go vet command, 9
- GOARCH constraint, 7–8
- GoLand, 5–6
- golint command, 9
- GOOS constraint, 7–8
- gopacket package, 174
- gopacket/pcap subpackage, 174–175
- GOPATH environment variable, 2–3
- goquery package, 69
- gorilla/mux package, 82–83, 84, 101
- gorilla/websocket package, 96
- GOROOT environment variable, 2–3
- goroutines, 16–17, 26–32
- gRPC framework, 316–319
- gss package, 138

## H

- HandleFunc() method, 82
- handler() function, 75–76

- handles, 271. *See also* tokens
- handshake process, 22–23
- hash-based authentication, 147–150
- hashing, 234–239
- Head() function, 46
- head() HTTP function, 226–227
- hex transform, 214
- hexadecimal 198, 281, 297
- HMAC (Keyed-Hash Message Authentication Code) standard, 240–241
- Holt, Matt, 127
- host search, 55–57
- HTTP clients
  - overview, 46–51
  - Bing scraping, 68–76
  - Metasploit interaction, 59–68
  - Shodan interaction, 51–59
- HTTP servers
  - overview, 78–90
  - credential-harvesting attacks, 90–93
  - multiplexing, 98–102
  - WebSocket API (WebSockets), 93–98
- http.HandleFunc(), 78–79

## I

- if statements, 18
- implant code, 323–325, 327–329
- import address table (IAT), 279
- indexing metadata, 68–76
- infinite loops, 37
- init() function, 101
- input/output (I/O) tasks, 32–35
- instreamset filter, 73
- integrated development environments (IDEs), 3–6
- interface{} type, 97
- interface types, 13
- io package, 32, 197
- io.Pipe() function, 43
- io.ReadCloser, 49
- io.Reader, 32–35, 46
- ioutil.ReadAll() function, 49
- io.Writer, 32–35

## J

- Java, 118–120
- JavaScript, 94–95
- JBoss, 198

JetBrains GoLand, 5–6  
jQuery package, 69  
JS Bin, 94  
JSON, 19, 50, 54, 139, 159

## K

Kerberos, 133  
Kernel32.dll, 275  
Keyed-Hash Message Authentication  
Code (HMAC) standard,  
240–241  
keylogging, 93–98  
Kozierok, Charles M., 22

## L

lab environments, 118–121  
len field tag, 140  
libraries, 2  
lightweight threads, 16–17  
loadLibraryA() function, 275  
Login() method, 66  
Logout() method, 66, 68  
loops, 15, 37  
Lua plug-ins, 225–232  
Luhn checks, 253–254

## M

madvise() function, 205  
magic bytes, 296  
main() function, 17  
main package, 6  
make() function, 11  
Mandatory Integrity Control, 271  
mapping data, 71–73, 125  
maps, 11  
Marshal() method, 19  
marshalData() method, 305  
marshaling interfaces, 135  
MD5 hashes, 236–237  
memory, 273–274  
message authentication, 239–241.  
*See also* authentication  
message authentication codes  
(MACs), 234  
MessagePack, 60  
metadata, 69, 138–139  
Metasploit Framework, 59–68, 213  
Meterpreter, 61, 98–102  
Microsoft API documentation, 263–265

Microsoft SQL (MSSQL) Server  
databases, 157–158, 160–161  
Microsoft Visual Studio Code, 5  
middleware, 80–81, 83–88  
MinGW-w64, 290  
mod tool, 9  
MongoDB databases, 154–156, 158–160  
MsfVenom, 213, 278  
Msg struct, 106–107  
MSYS2, 290  
multichannel communication, 30–32  
multiplexing, 98–102  
mutex, 129  
mutual authentication, 248–252  
MySQL databases, 156–157, 160–161

## N

named functions, 126  
native plug-ins, 218–224  
negrone package, 83–88  
Nessus vulnerability scanner, 217  
net package, 24–25, 197  
Netcat, 40–44  
net.Conn, 35  
net/http standard package, 46, 48  
New() helper function, 53–54  
NewProperties() function, 72–73  
NewRequest() function, 48  
Nmap, 225  
nonconcurrent scanning, 25–26  
NoSQL databases, 154, 158  
NT LAN Manager (NTLM)  
authentication, 150–151  
NTLM Security Support Provider  
(NTLMSSP), 133–135  
NTOWFv2, 148  
num transform, 214

## O

obfuscation, 307  
Office Open XML documents, 69  
offset field tag, 140  
offset values, 300  
omitempty, 62  
open ports, 22  
OPSEC, 329  
Optional Header, 284–285  
Oracle, 154  
os package, 197  
os/exec package, 41

## P

- packages, 2, 8–9
- packet capturing and filtering, 175–180
- panic() function, 107, 112
- parseTags() function, 140–142
- passive reconnaissance, 51, 59
- pass-the-hash authentication, 147–150
- passwords, 146–151, 222–224
- PATCH requests, 47
- payloads, 101, 302–307
- pcap, 175
- PDF files, 69
- PE (Portable Executable) format, 279–289
- PipeReader, 43
- PipeWriter, 43
- PKCS (Public Key Cryptography Standards), 242. *See also* public-key cryptography
- pkg directory, 2–3
- placeholders, 83, 89
- Plan 9 operating system, 216
- plug-ins
  - Lua, 225–232
  - native, 218–224
  - plugin package, 219
- PNG format, 296–307
- pointers, 12
- Portable Executable (PE) format, 279–289
- Portable Network Graphics (PNG) images, 296–307
- ports
  - availability, 24–25
  - handshake process, 22
  - port forwarding, 23, 39–40
  - port scanners, 180–185, 222–224
  - scanning, 23–32. *See also* scanners
- Post() function, 46–47
- PostForm() function, 47
- Postgres databases, 156–157, 160–161
- PostgreSQL databases, 156–157, 160–161
- PreProcessImage() function, 298
- primitive data types, 10–11
- process() function, 72–73
- Process Hacker, 278
- process injection, 268–269
- Process Monitor, 278
- ProcessImage() method, 302–303
- proclselfmem() function, 205
- project structure, 52–53, 60

- promisc variable, 177
- Protocol Buffers (Protobuf), 316
- PsExec, 131
- public-key cryptography, 242, 245.  
*See also* encryption
- PUT requests, 47–48
- Python, 197–201

## Q

- query parameters, 73–76

## R

- race condition functions, 206
- Rapid7, 60
- RATs (remote access Trojans), 315–329
- raw transform, 215
- RC2, 252–261
- ReadString() function, 38
- reconnaissance, 51, 59
- redirectors, 98
- referential fields, 138–139
- reflect package, 139
- reflection, 132, 139
- regular expression (regex) values, 163
- remote access Trojans (RATs), 315–329
- remote procedure calls (RPCs), 59, 64–67, 316
- request/response cycles, 46, 62–64
- response handling, 48–51
- Rivest, Ron, 252
- RLock, 129
- Roundcube, 90
- routers, 79–80, 84–85
- rst packets, 22

## S

- salts, 234
- scanner package, 220, 223
- scanners, 23–32, 180–185, 217, 222–224. *See also* ports
- schema-less databases, 154
- scraping metadata, 68–76
- Search() function, 163
- search query templates, 73–76
- Section Table, 287–289
- security tokens, 133–134
- send() method, 65
- serveFile() function, 97
- Server Message Block (SMB), 132–147
- server multiplexers, 78–79

- ServerMux, 78–79
- SessionList() method, 66, 68
- set command, 3
- SHA-256 hashes, 236–237
- shellcode, 203–204, 213–216
- Shodan, 51–59
- signature validation, 245–248
- site filter, 73
- slices, 11, 106, 126, 144–145
- SQL injection fuzzing, 192–196
- SQLite databases, 328
- src directory, 3
- stateless protocols, 46
- static files, 93
- Status struct, 50–51
- steganography
  - overview, 295
  - PNG format, 296–307
  - XOR, 307–312
- strconv package, 25
- strlen() function, 17
- strToInt() method, 304
- structs
  - APIInfo struct, 55
  - Client struct, 53–54
  - encoding, 135
  - Foo struct, 19
  - handling, 142–143
  - Msg struct, 106–107
  - Status struct, 50–51
  - types of, 12–13, 19, 133–135
- structured data, 18–19, 50–51
- Stub, 281
- subdirectories, 2–3
- subdomains, 107–117
- switch statements, 14, 129, 143
- switched networks, 178
- symmetric algorithms, 234
- symmetric-key encryption, 242–245.
  - See also* encryption
- SYN cookies, 180–185
- syn packets, 22
- syn-acks, 22
- SYN-flood protections, 180–185
- syscall package, 197, 266–269
- Syscall16() function, 210

## T

- tabwriter package, 113–114
- Target breach, 154
- TCP flags, 180–181

- tcpdump, 102, 105, 175–178
- TCP/IP Guide* (Kozierok), 22
- teamservers, 121
- Telegram, 280
- Telnet, 41
- templates, 88–90
- Tenable, 217
- third-party packages, 8–9
- tokens, 61–63, 271
- “too fast” scanner, 26–27
- Tour of Go* tutorial, 10
- Transmission Control Protocol (TCP)
  - handshake process, 22–23
  - port scanners, 23–32
  - proxies, 32–44

## U

- Ubuntu VM, 118–120
- uint16 data types, 143–144
- uintptr type, 266
- unicode package, 197
- unmarshal() function, 141–142
- Unmarshal() method, 19
- unmarshaling interfaces, 136
- unsafe package, 197
- unsafe.Pointer type, 266–267
- USER property, 190
- utility programs, 67–68

## V

- `{{variable-name}}` convention, 89
- verbs, 47
- Vim text editor, 3–4
- vim-go plug-in, 3
- virtual machines (VMs), 118–120
- virtual memory, 273–274
- VirtualAllocEx, 273–274
- VirtualFreeEx() Windows function, 277–278
- VMWare Workstation, 118–120
- VS Code, 5
- vulnerability fuzzers, 188–196

## W

- WaitforSingleObject() Windows
  - function, 276–277
- waitForWrite() function, 206
- WaitGroup, 27–28
- walkFn() function, 171
- WebSocket API (WebSockets), 93–98

- while loops, 15
- Windows APIs, 263–265
- Windows DLL, 218–219
- Windows VM, 127
- winmods files, 270
- WINNT.H header, 285–286
- Wireshark, 102, 225
- worker functions, 28–30, 111–112
- wrapper functions, 136–137
- writeData() function, 305–307, 311

- WriteProcessMemory() function,  
274–275
- writer.Flush() function, 38
- WriteString() function, 38

## **X**

- XML, 19–20, 69
- XOR, 307–312