INDEX

A

Abrams, Lawrence, 245
acceleration checks, 121–122
access tokens, 253–254
AddAtom function, 209, 424
address space layout randomization (ASLR), 15–16, 62–63
AddVectoredExceptionHandler function, 192, 424
AdjustTokenPrivileges function, 269, 424
administrator account, 249
Agent Tesla, xxiv
alertable state, 208
Al-Khaser, 415–416
Allievi, Andrea, 436
Alternate Data Streams (ADS), 307–309
Andriesse, Dennis, 436
anti-attach techniques, 167, 180
anticuckoo, 143
anti-debugging, 167
    checking parent processes, 173
    countering, 181
    in Joe Sandbox, 34
    timing checks, 171–172
anti-disassembly, 151
anti-forensics, 289–290, 306–317
anti-hooking, 137–144, 238
    toolsets, 143–144
anti-malware exclusions, 236
anti-malware software, 224
    cloud-based sandboxing, 226
    hash-based detection in, 225
    heuristic-based detection in, 226
    limitations and challenges of, 226–227
    signature-based detection in, 225
anti-unpacking, 383–385
antivirus (AV) software, 224
Any.Run, 20, 437
APC (asynchronous procedure call) injection, 207–208
API. *See* application programming interface
apihashes plug-in, 165
API Monitor, 201–202, 206–207. *See also* tracing API calls
application logs, 313
application programming interface (API). *See also* hooking; Windows API
    call obfuscation, 160–165
    hammering, 135
    indirect calls, 161–163
    tracing calls, 69–71
applications, 4
AppLocker, 221, 294
Arbitrary Code Guard, 221
Arntz, Pieter, 246
artifacts
    filesystem, 234
    hiding, 306–312
    operating system, 75
    process, 233
    registry, 234
ASCII, 27
ASLR (address space layout randomization), 15–16, 62–63
    memory corruption, 15–16
aspnet_compiler.exe, 305
assembly code
    x64 (64-bit), 44
    x86 (32-bit), 44
assembly instructions
    arithmetic operations, 48–49
    call and return operations, 50
    jump operations, 50

assembly instructions *(continued)*
move operation, 49
no-operation, 51, 69
stack operations, 47–48
value comparisons, 49–50
ASUS, xxii
asynchronous procedure call (APC) injection, 207–208
AsyncRAT, xxiii
atoms, 209
Autochk, 279–280
AV (antivirus) software, 224
Ave Maria, 25

B

Background Intelligence Transfer Service (BITS), 299
backward link (blink), 9, 273–274
banking trojans, xxiii
bare-metal analysis systems, 419
base64dump.py, 322, 405
Base64 encoding, 319–322
Bashlite, 257
Bataille, Adrien, 309
`Bcrypt*` functions, 30, 335–336, 343, 424
beacons, 295
`BeingDebugged` value, 10, 168
bespoke malware, xxiv
binary instrumentation, 382, 420–421
dynamic, 420
static, 420
Binwalk, 310
BITS (Background Intelligence Transfer Service), 299
BITSAdmin (*bitsadmin.exe*), 299
bit manipulation, 310. *See also* steganography
bit test (`bt`) instruction, 125
Black Hat conference, 212, 246
blink (backward link), 9, 273–274
block ciphers, 326
`BlockInput` function, 179, 424
BluePill, 421
bootkits, xxiv, 285
bootstrap code, 204
botnets, 261
BPHs (bulletproof hosters), 262
Bratus, Sergey, 437
breakpoints
in debuggers, 58
detecting and circumventing, 174–176
in hardware, 365
in memory, 66–68, 175–176
traps, 174, 177–187
Bring Your Own Vulnerable Driver (BYOVD) techniques, 271–272
browsing history, 96
`bt` (bit test) instruction, 125
bulletproof hosters (BPHs), 262

C

C2 (command and control)
frameworks, 136, 259
servers, 39
cache, 96
callback functions, 106, 186
`CallNamedPipe` function, 86
CAPA, 56–57, 336–337, 405
CAPE, 31
Carbanak backdoor, 217
carry flag register, 125
Case, Andrew, 436
central dispatch code block, 157
certificate trust and signing abuse, 245–246
certificate trust store, 245
certmgr, 246
certutil.exe, 290, 299–300
CFF Explorer, 23, 62, 350, 353
CGG, 305
Chappell, Geoff, 7
`CheckRemoteDebuggerPresent` function, 168–169, 424
checksums, 178–179
Chlumecký, Martin, 284
Chrome browser file locations, 96
`CloseHandle` function, 170, 424
Cloudburst, 146
cmd.exe, 29, 250–251, 300–301, 369
`CmRegisterCallback(Ex)` callback, 231, 284–285
Cobalt Strike, 204, 295–297
`CoCreateInstance` command, 253

CoInitialize function, 253
coinminers, xxiv
COM. *See* component object model
command and control (C2)
  frameworks, 136, 259
  servers, 39
command line arguments, 119
committed memory, 13
commodity malware, xxiv
component object model (COM)
  abuse, 252
  class ID, 253
  COMAutoApproval property, 253
  interfaces, 253
  objects, 253
  registry tampering, 253
context awareness, xxii
CONTEXT structures, 176
control flow, 45, 50, 154–160
  flattening, 156
  maintaining with index and pointer registers, 45
  modifying with jump instructions, 50
  obfuscation, 154
    countermeasures, 159–160
Control Flow Guard, 221
converting code and data, 153
cookies, 96
Cornelis, 243
CosmicStrand, 285
covert execution, 185
cpuid instructions, 57, 125, 413
CPU registers, 44–51
cradles, PowerShell, 302–303
CRC-32 hash, 325
CreateFile function, 86, 211, 278, 424
CreateFileTransacted function, 213, 424
CreateMutex function, 85, 424
CreateProcess function, 173, 206–208, 213, 254, 296, 366, 370, 425
CreateProcessInternal function, 366, 425
CreateProcessNotifyRoutine function, 283, 432
CreateProcessWithToken function, 254, 425
CreateRemoteThread function, 30, 199–201, 203, 296, 367, 425
CreateService function, 269–270, 425
CREATE_SUSPENDED flag, 206–207
CreateToolhelp32Snapshot, function, 76, 145, 173, 233, 425
CreateTransaction function, 213, 425
CreateWindow(Ex) function, 7–8
credentials, extracting and reusing, 254
crypters, 346
Crypt* functions, 162–163, 332–335, 337, 343, 378, 425. *See also* decryption; encryption
CryptoAPI, 324–325, 332–335
  cryptographic provider types, 334
  cryptographic service provider (CSP) module, 332
  Cryptography API: Next Generation (CNG), 332, 335–336
cscript.exe, 300
csrss.exe, 249
Cuckoo, 20, 31–32, 75, 78–79, 115, 143–144, 437
CVE-2021-36934, 255
CVEdetails, 419
CyberChef, 322–324, 330, 339

D

d4rksystem, 410–411
data execution prevention, 221
DDoS (distributed denial of service), 261
DebugActiveProcess function, 58, 173, 180, 425
debug blocking, 180
debugging, 58–69, 167–180
  analysis with x64dbg, 62
  Arguments pane in x64dbg, 60
  breakpoints, 58
    detecting debuggers with, 174
    in hardware, 66–68
    in memory, 66–68, 175–176
    in software, 65–66
  choosing debuggers, 58
  CPU Registers pane in x64dbg, 60
  CPU tab in x64dbg, 59
  crashing, 179

debugging *(continued)*
  debug objects, 170
  disabling ASLR, 62–63
  Dump pane in x64dbg, 61
  exploiting, 179
  Memory Map tab in x64dbg, 61–62
  patching and modifying code, 68–69
  processes, 173
  running code in x64dbg, 64–65
  Stack pane in x64dbg, 61
  windows, 172–173
decoy data, 134
decoy network communication, 136
decryption, 325–330, 338–343
  asymmetric, 326–327
  dynamic, 339–343
  keys, 325–327, 333
  static, 339
Deep Freeze, 419
Dekel, Kasif, 287
delayed execution, 130–134
`DeleteFile` function, 307, 425
denial of service (DoS) bugs, 239
Desai, Deepen, 271
Detect It Easy (DIE), 160, 353
device drivers, 266
device enumeration, 88–90
  VirtualBox, 90
  VMware, 90
`DeviceIOControl` function, 270–272, 279, 425
DGAs (domain generation algorithms), 260–261
digital signatures, 245–246
  enforcement, 286–287
direct injection, 199
direct kernel object manipulation (DKOM), 272–274
directory and file enumeration, 98
  virtual machines, 78–79
direct syscalls, 239–243
DirtyMoe, 284
disabling host defenses, 235–236
disabling services, 237
disassemblers, 51
  choosing, 52–53
  Ghidra, 52–53
  IDA, 52–55
disassembly, 44, 51–52
  code vs. data problem, 52
  linear (aka linear sweeping), 51
  recursive (aka flow-oriented), 52
Dispatch Table, 275
disrupting manual investigation, 144–145
distributed denial of service (DDoS) attacks, 261
DKOM (direct kernel object manipulation), 272–274
`DllCharacteristics` field, 63
DLLs. *See* dynamic link libraries
DNS. *See* Domain Name System
domain configuration enumeration, 110
domain generation algorithms (DGAs), 260–261
Domain Name System (DNS)
  traffic, 40
  tunneling, 257–258
doppelganging, 212
DoS (denial of service), 239
double fast flux technique, 262
Dridex, 133, 177–178, 381
drivers
  enumeration, 88–90
  hardware device, 5
  non-hardware device, 5
Driver Signature Enforcement (DSE), 286–287
droppers, xxiii
DR registers, 175–176
`DsGetDcName` function, 110, 426
dummy code, 132–133, 156
`DuplicateToken(Ex)` function, 254, 426
Duqu, 309
dword (data type), 44
dynamically compiled malicious code, 304–305
dynamic API function resolution, 161
dynamic code analysis, 58–71
dynamic link libraries (DLLs), 8
  hijacking, 214–216, 251
  injection, 203
  module, 120
  search order, 214
DynamoRIO, 420

E

Eagle, Chris, 435–436
early launch anti-malware (ELAM) drivers, 231, 287
Eclypsium, 288
Edge browser file locations, 97
EDR. *See* endpoint detection and response
EFLAGS register, 46, 126
email protection technologies, 257
Emotet, 127, 297
Emsisoft, 333
emulation, code, 382, 420–421
emulation engine, 226
encoding, 319–323
encryption, 256–257, 319–320, 325–343
- asymmetric, 326–327
- decrypting encrypted malware data, 339–343
- locating and identifying cryptographic routines in malware, 336–338
- symmetric, 326–327

endpoint defenses
- actively circumventing, 235–239
- engine limitations in, 246–247
- history, 224
- identifying, 232–235
- passively circumventing, 239
- vulnerabilities in, 238–239

endpoint detection and response (EDR), 224, 227–246, 262–263, 272, 283
- agent, 227–230
- filesystem artifacts, 234
- kernel drivers and callbacks, 230–231
- logging and analysis, 231–232
- process artifacts, 233
- registry artifacts, 234
- telemetry, 227

endpoint protection platforms (EPPs), 224
entropy, 352–353
`EnumChildWindows` function, 186
`EnumDateFormatsEx` function, 186
`EnumDisplayMonitors` function, 106, 186, 426
enumeration, operating system
- devices and drivers, 88–90. *See also* hardware and device configuration enumeration
- directories and files, 78–79, 98
- domain configuration, 110
- installed software, 84–85
- IP address configuration, 109–110
- language settings, 90–92
- loaded module, 119–120, 173
- locale settings, 90–92
- mutex, 85–86
- open port, 115
- process, 76–78
- registry, 80–83
- run path, 118
- services, 83–84
- usernames and hostnames, 90
- version information, 92–93

`EnumServiceStatus` function, 83, 426
`EnumSystemFirmwareTables` function, 108, 426
`EnumSystemLanguageGroups` function, 186
`EnumWindows` function, 99, 146, 172, 426
EPPs (endpoint protection platforms), 224
EPROCESS structures, 9, 180, 272
- backward link (blink), 9, 273–274
- forward link (flink), 9, 273–274

ESET
- software, 233, 236
- threat research, 248, 259, 271, 309

`EtwEventWrite` function, 238
evasion, xxii–xxvi. *See also* sandboxes
Event Tracing for Windows (ETW), 229, 238
exception handlers, 189. *See also* structured exception handler
exceptions, 124
- `EXCEPTION_ACCESS_VIOLATION` error, 191

exclusive or (XOR), 327–331, 340
executive layer, kernel, 5
Exeinfo, 160, 353–354

exfiltration of data, 39
ExitProcess function, 130, 426
ExitWindows function, 134, 426
exploits, hypervisor, 146
export address table, 14
exports, 30, 204
ExpressVPN, 112
extended detection and response (XDR), 232
external IP addresses, 112
  validation, 112–114

F

FakeDNS, 389
FakeNet, 389
fast flux technique, 261–262
Fiddler, 36, 39
file appending, 309
file command, 22–23
file emulation, 226
fileless malware, 289–294
  fileless attacks, 290
  paradox of, 292
file metadata, 247
FILE_OBJECT attribute, 214
filetime structures, 315
FileTimeToSystemTime function, 315
file type, identifying, 22
filter drivers, 266, 281–282
  altitudes of, 282
FIN7, 216–217
FindCrypt, 337–338
FindFirstFile function, 78, 426
FindFirstUrlCacheEntry function, 97, 426
FindNextFile function, 78, 426
FindNextUrlCacheEntry function, 97, 426
FindWindow function, 99, 172–173, 426
Firefox file locations, 96
firewall configurations, 237
firmware, 107
  FirmwareTablesView function, 107
  table enumeration, 107
flags, CPU register, 46–47
FLARE-VM, 401–402, 405
flinks (forward links), 9, 273–274
FLOSS, 29, 164, 342, 405
flow-oriented disassembly (aka recursive disassembly), 52
FltEnumerateFilters function, 283, 427
FltEnumerateInstances function, 283, 427
FltGetFilterFromName function, 283, 427
FltRegisterFilter function, 282–283, 427
foreground windows enumeration, 98–99
Formbook, 136–137
forward links (flinks), 9, 273–274
Fox-IT, 252
FreeLibrary function, 139
frequency, performance counters, 123
FRIDA, 420
FsRtlRegisterFileSystemFilterCallbacks function, 231
FudModule, 271
function pointer abuse, 159

G

Gafgyt, 257
geofencing, 259
GetAdaptersAddresses function, 109–111, 427
GetAsyncKeyState function, 101, 427
GetComputerName function, 110, 334, 427
GetCurrentProcess function, 130
GetCurrentProcessId function, 173
GetCursorPos function, 100, 427
GetDiskFreeSpace function, 105, 427
GetForegroundWindow function, 98–99, 427
GetKeyboardLayout function, 91, 427
GetKeyboardLayoutList function, 91, 427
GetLastError function, 169–170, 427
GetLocalTime function, 172, 428
GetLogicalDrives function, 7
GetLogicalProcessorInformation function, 104, 428
GetModuleFileName function, 118, 428
GetModuleHandle function, 120, 173, 203, 428
GetNativeSystemInfo function, 104
GetPhysicallyInstalledSystemMemory function, 105, 428
GetProcAddress function, 133, 140–141, 161–162, 165, 203–204, 349–351, 370, 428

GetProcessPreferredUILanguages function, 91, 429
GetSystemDefaultLCID function, 91, 429
GetSystemDefaultUILanguage function, 91, 429
GetSystemFirmwareTable function, 108, 428
GetSystemInfo function, 104, 428
GetSystemMetrics function, 106, 428
GetSystemTime function, 172, 428
GetTcpTable function, 114–115, 429
GetThreadContext function, 176, 429
GetThreadLocale function, 91, 429
GetTickCount64 function, 429
GetTickCount function, 101–102, 121–123, 171–172, 429
GetUserDefaultLCID function, 91, 429
GetUserDefaultUILanguage function, 91, 429
GetVersionEx function, 92–93
GetWindowText function, 146, 429
Ghidra, 52–53, 153, 160, 337–338, 402, 436
GhostHook, 287
GlobalAddAtom function, 209, 429
GlobalGetAtomName function, 209, 429
Go (Golang), 245
Goodin, Daniel, 309
Google, 25, 113, 131, 245, 259
Gould, Tara, 305
GuLoader, 180, 325

H

hacktools, xxiv
Hahn, Karsten, 209
HalPrivateDispatchTable structure, 287
Hand, Matt, 436
handles in Windows, 6
hardening, VM and hypervisor, 412–414. *See also* malware analysis lab
hardware abstraction layer, 5
hardware and device configuration enumeration, 103–115
- CPU, 104
- hard disks, 105–106
- monitor configurations, 106
- RAM, 105
- USB controllers, 107

hardware breakpoints
- circumventing, 175–176
- setting in x64dbg, 365

hash-based detection, 225
hasherezade, 271
hashing, 23
- cascading, 324
- collisions, 24
- data, 164–165, 323–325
- MD5, 23–24
- section, 178–179
- SHA-1, 23–24
- SHA256, 23–24

Hatching Triage, 31, 437
heap, memory, 11
- heap flags, 170

HermeticWiper, xxiv, 267–271
heuristic-based detection, 225
hidden threads, 193
HideProcess, 274
hiding artifacts and code, 306–312
Higgins, Kelly Jackson, 248
hijacking DLL and shims, 214–218, 251
Hinchliffe, Alex, 307
HiveNightmare, 255
hives, Windows registry, 16–17, 255–256, 292, 294
HKEY root keys, 16
hollowed process injection, 205–207
HollowsHunter, 379–380
Honig, Andrew, 436
hooking, 100–101, 121, 137–141, 218–222, 378, 433
- circumvention, sandbox evasion, 140
- detection, sandbox evasion, 138
- import address table, 221
- injection, mitigation for, 221
- inline, 219
- keyboard hooks, 101
- mouse hooks, 100
- removal, sandbox evasion, 139–140
- SetWindowsHookEx function, 100–101, 218–219, 433

host machine, malware lab, 388
hostname-changer, 411

hostnames, 90, 411
  enumeration, 90
HTML application (*.hta*) files, 298
HTTP traffic, 40
Hybrid Analysis, 31, 437
HyperHide, 182–183
hypervisors, 20
  exploits, 146
  lab setup, 388–389, 390–391

I

IDA Pro, 52–55, 160, 248, 436
  analyzing with, 53–55
IDA Signsrch, 337
IDT (interrupt descriptor table) register, 124
image (IMG) memory, 13
`ImpersonateLoggedOnUser` function, 254, 429
import address table (IAT), 14
  hooking, 221
imports
  packers, 350–352
  PE file, 30
indicators of compromise (IOCs), 35
indirect API calls, 161–163
indirect syscalls, 242
InetSim, 389
infinite loops, 132–133
InfinityHook, 287
infostealers, xxiii
`InitiateShutdown` function, 134, 429
`InitiateSystemShutdown` function, 134, 429
InjectDLL shim, 216
inline hooking, 219
  kernel hooks, 276–278
input/output (IO) ports, 124–125
inspecting malware network traffic, 39–40
installed software enumeration, 84–85
installing Windows, 393–394
InstallUtil (*installutil.exe*), 299
integrity levels, 249
Intel Pin, 420–421
interactive behavioral analysis, 36–40
`InternetConnect` function, 113, 378, 430
Internet Explorer file locations, 96
`InternetOpen` function, 54–55, 65–66, 69, 430
`InternetReadFile` function, 113, 430
interrupt descriptor table (IDT) register, 124
interrupts, 124
Intezer, 31
intrusion prevention systems (IPS), 256
IO manager, 278, 281. *See also* IO request packets ports, 124–125
Ionescu, Alex, 436
`IoRegisterBootDriverCallback` callback, 231
IO request packets (IRPs), 275–282
  codes, 278–279
  function table, 279
  hooking, 278
  interception (aka filtering), 281–282
`IoWMISetNotificationCallback` callback, 231
IP address configuration enumeration, 109–110
`IsDebuggerPresent` function, 161, 168, 430
`IsNativeVhdBoot` function, 106
`IsProcessorFeaturePresent` function, 104, 430

J

JavaScript, 263
Joe Sandbox, 20, 31–35, 297–298, 437
jump tables, 161–163

K

Kaspersky, 224–225, 233–234, 239, 285–286, 315
`KeBugCheckEx` function, 287
kernel, Windows
  callbacks, 283–285
  debuggers, 169
  drivers, 230, 266
  executive layer, 5
  hooking, 274–315
  kernel-mode APIs, 7
  Memory Manager, 5
  mode, 3

modules, 266
Process Manager, 5
kernel32.dll, 7–8, 14, 161–162, 202, 228, 240
kernelbase.dll, 8
Kernel-Based Virtual Machine (KVM), 390
Kernel Patch Protection (KPP), 274, 286–288
keylogging, xxiv, 255
KillAV, 235
killware, xxiii
`KiUserExceptionDispatcher` function, 192
Kleymenov, Alexey, 436
Kosayev, Uriel, 247, 437
Kuznetsov, Igor, 286
KVM (Kernel-Based Virtual Machine), 390

L

labs. *See* malware analysis lab
language identifiers, 91
language setting enumeration, 90–92
last-in, first-out (LIFO), 47–48
Lazarus Group, 271
`LdrLoadDll` function, 228
`lea` (load effective address) instruction, 88
least significant bit (LSB) technique, 311–312. *See also* steganography
Lechtik, Mark, 286
legacy filter drivers, 281. *See also* filter drivers
Legezo, Denis, 315
Lesueur, Jean-Pierre, 437
Levy, Jamie, 436
Ligh, Michael Hale, 436
linear disassembly (aka linear sweeping), 51
Linux, 389–390
Living Off The Land Binaries (LOLBins), 292, 294–306
loaded module enumeration, 119–120, 173
load effective address (`lea`) instruction, 88
loaders, xxiii
`LoadLibrary` function, 133, 142, 161, 164–165, 203, 219, 349–351, 370, 428, 430
local descriptor table (LDT), 124
locale and language enumeration, 90–92
Local Security Authority Subsystem Service (LSASS), 254
Lockbit, 325
logic bomb, 132
log tampering, 312–316
Lokibot, xxiv, 299
LOLBins (Living Off The Land Binaries), 292, 294–306
low-level functions, 141–142
lsass.exe, 254

M

MAC addresses, 110–112, 397–399
machine code, 44
macros, 295
magic bytes, 22
major function table, 279
MalShare, 438
Mal_Unpack, 380–381
malware
analysis overview, xxv, 19–21
behavioral analysis, 36–40
configuration extracted from, 35
detecting with Yara, 26–27
monitoring behaviors, 37–40
search engines for, 25–26, 28
triage, 21
types of, xxii–xxv
malware analysis lab, 387–422
advanced VM and hypervisor hardening, 412–414
architecture, 388–390
bare-metal analysis, 419–420
building and configuring, 390–414
operational security and effectiveness, 416–419
simulating network services, 416–418
stress-testing your VM, 414–416
MalwareBazaar, 25, 438
Malwarebytes, 213

Malware Traffic Analysis, 438
Mandiant, 56, 217, 309, 401
manually loading libraries and calling functions, 142–143
mapped memory, 13
masquerading, 247
master boot record (MBR), xxiv, 316
Matrosov, Alex, 437
McAfee, 22, 214, 224, 233
MD5, 23–24, 323
media access control (MAC) addresses, 110–112, 397–399
Mele, Gage, 305
`memcmp` function, 121
`memcpy` function, 140
memory
- bombing, 247
- breakpoints, 66–68, 175–176
- committed, 13
- corruption, 15–16
- deallocation, 362
- heap, 11
- image, 13
- mapped, 13
- page guards, 177
- physical, 11
- private, 13
- protections, 359, 363–364
- read-write-executable, 228
- reserved, 13
- strings in, 120
- virtual, 11–13

Memory Manager, kernel, 5
memory-resident malware, 289. *See also* fileless malware
metadata in a file, 30–31, 315
Microsoft Build Engine, 305
Microsoft Cryptography API. *See* CryptoAPI
Microsoft Defender, 232, 236, 401
Microsoft Hyper-V, 390
Microsoft Office, 229
Microsoft Security Center, 232, 235
Microsoft Vulnerable Driver Blocklist, 287
Mimikatz, 255
MinGW compilers, 305
minifilter drivers, 231, 267, 281
Mitre CVE database, 238
`MmProtectMdlSystemAddress` function, 277
modes in Windows, 3
`Module32First` function, 120, 173, 235, 430
`Module32Next` function, 120, 173, 235, 430
Mofang, 252
monitoring malware behaviors, 37–40
Moriya, 266
MosaicRegressor, 285–286
mouse coordinates, 100
`movdqa` (Move Aligned Double Quadword) instruction, 126
msbuild.exe, 305
msconfig.exe, 250–252
Mshta (*mshta.exe*), 298
`MsiEnumProducts` function, 84
multistage attacks, 262–263
mutation, 243–244
- code block reordering, 243
- engines, 244
- register reassignment, 244

mutexes, 6
- enumeration, 85–86
- VirtualBox, 86
- VMware, 86

`MZ` header, 22

N

Nance, Kara, 436
Native API, 7
NCC Group, 252
Necurs, 284
.NET framework, 144, 245, 305
netstat.exe, 114
network controls, 256
network defenses, circumvention, 256–262
network detection and response (NDR), 257
network intrusion detection systems (NIDS), 256–257
network traffic
- analysis in malware, 39–40
- obfuscating and obscuring, 257

Nim, 245
Nirsoft, 107
No Pill technique, 123–124
NordVPN, 112
Norton, 224
NtAllocateVirtualMemory function, 7, 139–140, 199, 240–241, 361–362
NtClose function, 170, 424
NtCreateFile function, 143, 219–221, 278–280, 424
NtCreateProcess function, 8, 211, 213, 425
NtCreateSection function, 143
NtCreateThreadEx function, 193, 200–201, 211–212, 425
NtCreateTransaction function, 430
NtCreateUserProcess function, 211, 228, 425
NtDeleteFile function, 425
ntdll.dll, 7–8, 14, 143, 161, 219–221, 228, 240–243
NTFS filesystem, 300, 307–308, 315
ntinternals, 7
NtLoadDriver function, 270, 430
NtMapViewOfSection function, 143, 242, 430
NtOpenDirectoryObject function, 88, 430
NtOpenProcess function, 199, 431
ntoskrnl.exe, 5, 7, 275–278, 280, 283
NtProtectVirtualMemory function, 141, 363
NtQueryDirectoryObject function, 88, 430
NtQueryInformationProcess function, 169, 430
NtQueryObject function, 170, 430
NtQuerySystemInformation function, 108, 169, 431
NtQuerySystemTime function, 172, 431
NtQueryVirtualMemory function, 176, 434
NtQueueApcThread function, 208–209, 432
NtReadVirtualMemory function, 138, 432
NtResumeThread function, 207, 242, 381, 432
NtSetInformationFile function, 316
NtSetInformationThread function, 179, 193, 431
NtUnmapViewOfSection function, 206, 229, 431
NtWriteVirtualMemory function, 200–201, 207, 242, 434
NumberOfProcessors value in PEB, 10
Nvidia, 246

O

objects in Windows, 6
OEP (Original Entry Point), 347–350, 371–374, 377
Office Recent Files list, 97
OllyDbg, 179
opaque predicates, 157–158
OpenMutex function, 85–86, 431
open port enumeration, 115
OpenProcess function, 145, 198–199, 201, 203–204, 208, 254, 431
OpenProcessToken function, 431
OpenService function, 83, 431
OpenThread function, 207–209, 431
operating system artifacts, 75
operational security (OPSEC), 416–419
Orion (SolarWinds), xxii
Osipov, Arnold, 251
OutputDebugString function, 169–170, 427, 431

P

packers, 244, 346–354
- architecture and functionality
  - resolving imports, 349–350
  - transferring execution to the OEP, 350
  - unpacking the malware payload, 348–349
- identifying packed malware, 350–354
  - automated packer detection, 353–354
  - entropy, 352–353
  - imports, 350–352
  - PE sections, 353
  - strings, 352
- types of, 346

Pafish, 414–416
page table in virtual memory, 11
Parshin, Yury, 286
PatchGuard, 274, 286–288
`PathFindFileName` function, 118
PEB. *See* Process Environment Block
PE-bear, 181, 350
PE Detective, 353
PE file format. *See* Portable Executable file format
persistence, 134, 292–294
PE-Sieve, 379–380
Pestr, 164
PEStudio, 28–29, 181, 187, 350
physical memory, 11
PID (process ID), 10
ping.exe, 131
pipes, 86–87
  pipelist, 86–87
  in VirtualBox, 87
  in VMware, 87
Portable Executable (PE) file format
  export address table, 14
  headers, 13
    DOS, 13
    optional, 13, 62–63
  import address table, 14
  loading process, 14–15
  sections, 13–14, 353
  triage tools, 23
  Windows PE loader, 13
`PostMessage` function, 145–146, 431
PowerShell, 29, 229–230, 236–237, 290–293, 302–305, 409
  cradles, 302–303
  execution policy, 237
  obfuscation, 303
  querying WMI, 303–304
primary data streams, 307
priority of a process, 236
private keys, 326
private (PRV) memory, 13
privilege elevation, 248
  vulnerabilities, 255–256
ProcDump, 296
`Process32First` function, 76, 173
`Process32Next` function, 76–77, 173
`ProcessDebugPort` function, 169
process doppelganging, 212–213
process enumeration, 76–78
Process Environment Block (PEB), 9–10, 104–105, 167–168, 170–171, 204
  anti-debugging, 167–168
    `BeingDebugged` function, 10, 168
    `NtGlobalFlag` function, 168
    `ProcessHeap` function, 168
  `NumberOfProcessors` value, 10
  offsets, 10
Process Explorer, 38
process ghosting, 214
Process Hacker, 12, 36, 38, 249, 267, 402
process herpaderping, 211–212
process hollowing, 205–207
process ID (PID), 10
process image manipulation, 197, 210–214
  doppelganging, 212–213
  herpaderping, 211–212
  ghosting, 214
  reimaging, 214
process injection, 138, 193, 197–210, 366–370
  asynchronous procedure call, 207–208
  atom bombing, 209
  direct, 199
  DLL, 203
  mitigations for, 221
  monitoring, 366–369
  process hollowing, 205–207
  reflective DLL, 204–205
    shellcode reflective, 205
    staged, 205
    stageless, 205
  self-injection, 199
  shellcode, 199–202
  thread hijacking, 207
  tracing, 369–370
Process Manager, kernel, 5
process reimaging, 214
process replacement, 205–207
process tree in a malware sandbox, 33
Procmon (Process Monitor), 36–38, 233–234, 402

programmable logic controller, xxi
programming languages, uncommon, 244–245
Proofpoint, 248, 259
provider signatures, 108
PSExec, 296
PsLookupProcessByProcessID function, 273, 431
PsSetCreateProcessNotifyRoutine function, 210, 212, 432
PsSetCreateThreadNotifyRoutine function, 231, 283–284, 432
PsSetLoadImageNotifyRoutine function, 231, 432
public keys, 326
PyCrypto, 338
Python, 245

Q

Qbot, 215
Qiling Framework, 421
QueryPerformanceCounter function, 123, 432
QueueUserAPC function, 208, 432
qwords (data type), 44

R

race condition, 6
RAM, 11
ransomware, xxiii
RATs (remote access trojans), xxiii
rdtsc (Read Time-Stamp Counter) instruction, 122–123, 171–172
ReadFile function, 138, 140
ReadProcessMemory function, 120–121, 138–139, 221, 432
read-write-executable (RWX) memory, 228
recursive disassembly (aka flow-oriented disassembly), 52
Red Pill technique, 123–124
Red Team Notes, 243
reflective DLL injection, 204–205
Regedit (Registry Editor), 17
regedit.exe, 82
RegEnumKey function, 81, 133, 141, 432
RegEnumValue function, 80–81, 432
registers, CPU, 44–51
registry, Windows. *See* Windows registry
RegOpenKey function, 80–81, 284, 432
RegQueryValue function, 38
regsvr32 (*regsvr32.exe*), 297–298
Remcos, xxii–xiii, 242
Remnux, 389, 404–408, 416–417
remote access trojans (RATs), xxiii
repairing unpacked executables, 375–377
reserved memory, 13
ResumeThread function, 206–207, 366, 381–382, 432
return pointer abuse, 158
RFLAGS register, 46
Rivest Cipher 4 (RC4), 327, 331–332, 340–241
Roccia, Thomas, 271, 437
Rodionov, Eugene, 437
rogue byte technique, 152–153. *See also* disassembly
RollbackTransaction function, 213, 432
rootkits, xxiv, 265–267
    defenses against, 286–288
    direct kernel object manipulation, 272–274
    inspecting EPROCESS blocks, 180
    installation, 269–270
ROR-13 hash, 164–165, 325
round-robin technique, 261
RtlAddVectoredExceptionHandler function, 424
RtlCopyMemory function, 277, 432
RtlCreateUserThread function, 200, 425
RtlDecompressBuffer function, 378
RtlQueryProcessDebugInformation function, 171
RtlQueryProcessHeapInformation function, 171, 433
RtlSetCurrentTransaction function, 213
RtlZeroMemory function, 307, 433
rundll32 (*rundll32.exe*), 37–38, 295–298
RunOnce keys, 293
run path enumeration, 118
RunPE, 205–207
    RunPE Unpacker, 381–382
Run registry keys, 293–294

Russinovich, Mark E., 436
Rust, 245
RWX (read-write-executable) memory, 228

S

Saini, Ankur, 271
Salinas, Rotem, 283
sandboxes, 20
  automated malware triage with, 31–36
  circumventing analysis, 144
  cloud-based, 226
  commercial, 31
  detection
    function execution timing for, 123
    performance and timing indicators for, 122–123
    risks of using detection techniques, 127
    unsupported instruction sets for, 125–126
  evasion
    countermeasures, 146–147
    defense evasion, 35
    logouts for, 134
    reboots for, 134
    self-termination for, 130
    unnecessary process spawning for, 136
  free and open source, 31
  process tree in, 33
Sandboxie, 120
Sandworm, 248
SANS SIFT Workstation, 389
Santos, Doel, 307
Santos, Roberto, 271
Scylla, 358–359, 374–377
ScyllaHide, 181–182
SDB Explorer, 217
`SeBackupPrivilege` function, 253
section hashing, 178–179
section object, 143
Secure Boot, 288
security logs, 313
Security Onion, 390
`SeDebugPrivilege` function, 254
SEH. *See* structured exception handler
self-healing techniques, 178–179
self-injection techniques, 199
`SeLoadDriverPrivilege` function, 254, 270
Service Control Manager, 83, 249
services, 83–84
  enumeration, 83–84
  VirtualBox, 84
  VMWare, 84
services.exe, 83, 249
`SeShutdownPrivilege` function, 254
`SetFileAttributes` function, 306, 433
`SetFileInformationByHandle` function, 316
`SetFileTime` function, 316, 433
`SetLastError` function, 169–170, 427
`SetPriorityClass` function, 236, 433
`SetThreadContext` function, 176, 206–207, 433
`SetUnhandledExceptionFilter` function, 178, 433
`SetWindowsHookEx` function, 100–101, 218–219, 433
SHA-1, 23–24, 323
SHA256, 23–24, 323
ShadowHammer, xxii, 111
shared folders, 79–80, 400–401, 418
shellcode
  injection, 199–202, 296
  reflective injection, 205
`SHEnumKeyEx` function, 141
shims, 214–216
  hijacking, 214, 216–218
shutdown.exe, 134
`sidt` instruction, 124
Siemens programmable logic controller, xxi
signature-based detection, 225
  circumventing, 243
signature of a file, 22
Sikander, Usman, 243
Sikorski, Michael, 436
`sldt` instruction, 124
`Sleep` function, 98–100, 121–122, 131, 433
sleep routine, 131, 247, 263, 411
snapshots, 388–389, 408–409

SocGholish, 418
social engineering, 247
sockets, 142
SolarWinds Orion, xxii
Solomon, David A., 436
Speakeasy, 382, 405, 421
spyware, xxiv
Squiblydoo, 298
SSDT (System Service Descriptor Table), 275–276
SSE instruction set, 126
stack, memory, 11, 47–48
stack strings, 163–164
stack unwinding, 193
Stancill, Blaine, 309
`StartService` function, 269–270, 433
static code analysis, 23, 51–57
static properties, analyzing, 27
Stegano, 309
steganography, 309–312
Steghide, 312
Stevens, Didier, 322, 329, 405
Stone-Gloss, Brett, 271
stream ciphers, 326
string obfuscation, 160–165
strings
  ASCII, 27
  extracting and analyzing, 27–28
  inspecting in malware sample, 352
  in memory, 120
  unicode (aka wide) strings, 27
StringSifter, 29–30
`strings` tool, 27–28
structs, 104
structured exception handler (SEH), 158–159, 178, 189–193
  frames, 189
  handler abuse, 158–159
  records, 189–191
Stuxnet, xxi, 92
Sunburst, xxii
`SuspendThread` function, 207, 433
symmetric encryption, 326–327
syscalls, 7
  direct, 239–243
  indirect, 242
  number (syscall ID), 241
  stub, 240
  sysenter, 241
`sysinfo` command, 102
Sysinternals suite, 37, 88
System Binary Proxy Execution, 296
system destruction, 316–317
`SystemFirmwareTableInformation` parameter, 108
system logs, 313
System Service Descriptor Table (SSDT), 275–276
system service number (SSN), 241
system uptime, 101

T

TA547, 259
tailing jump, 350, 372
tampering with logs and evidence, 312–316
tamper protection, 401
Tanner, Amanda, 307
TCP connection states, 114
telemetry, 227
`TerminateProcess` function, 91, 130, 145, 228, 235, 433
Thabet, Amr, 436
`THREAD_CREATE_FLAGS_HIDE_FROM_DEBUGGER` flag, 193
Thread Environment Block (TEB), 10–11, 186, 188–189
  offsets, 10
  process ID, 10
  thread ID, 11
  thread-local storage, 186
`ThreadHideFromDebugger` value, 179, 193
thread hijacking, 207
Thread Information Block (TIB). *See* Thread Environment Block
`ThreadInformationClass` parameter, 193
thread-local storage (TLS) callbacks, 186–189
ticks, 101, 122
time bomb, 132
timeout, sandbox evasion, 131
timestomping, 315
tiny_tracer, 421
token theft, 254

TOR (The Onion Router), 112
tracing API calls, 69–71
trampolines, 221
transacted hollowing, 213
Transactional NTFS, 212–213
trap flag (TF), 47, 50, 126, 181
traps, 174, 177–187
triage (techniques), 21, 23
Trickbot, 245, 251
Tripathi, Rahul Dev, 283
trojan horses (trojans), xxiv
  in banking, xxiii
  remote access, xxiii
Turla, 259

U

UAC. *See* User Account Control
UACME, 253
UEFI (Unified Extensible Firmware Interface), 285–288
unaligned function calling, 140–141
uncommon functions, 141–142
unhandled exceptions, 178
unhooking, 139, 238, 137
unicode (aka wide) strings, 27
Unicorn, 421
unnecessary code, 155–156
unnecessary jump statements, 154–155
unpacking
  analyzing without, 383
  anti-unpacking techniques, 383–385
  automated, 354–383
    fully automated, 355
    sandbox-assisted, 355–357
  manual dynamic, 357–382
  manual static, 382–383
  process overview, 348–349
  repairing executables, 375–377
UnpacMe, 31, 356, 437
Upclicker, 100
uptime, 101
uptime.exe, 102
UPX, 346, 348, 353
USB controllers, 107
user32.dll, 8, 138
User Account Control (UAC)
  bypassing, 248–254, 269
  prompts, 250, 253
user-mode APIs, 7
username enumeration, 90

V

values, Windows registry, 16
VBA (Visual Basic for Applications), 290
VBA macro-based malware, 295
VboxCloak, 403, 410
VboxHardenedLoader, 414
VBScript, 300
vectored exception handling (VEH), 192–193
version information enumeration, 92–93
  versions of Windows, 93
Virlock, 244
`VirtualAlloc` function, 7, 13, 199–208, 296, 359–362, 366, 434
VirtualBox, 79–81
  concealment, 410
  creating a virtual machine in, 392–400, 404–406
  default IP address range, 109–110
  devices, 90
  drivers, 88–89, 272
  Guest Additions Service, 83–86, 401–404
  hardening, 414
  in lab setup, 390–391
  mutexes, 86
  paths, 79
  pipes, 87
  prefixes, 107, 112, 397
  process names, 77
  registry keys, 82
  snapshots, 408, 419
  strings, 83, 108, 125, 410
  updating, 419
  verifying settings, 391
  in WMI output, 304
virtual machines (VMs), 20
  creation, 392–393
  directory and file enumeration, 78–79
  escaping, 146
  kernel-based, 390
  in packers, 385

processes in, 77
snapshots, 388–389, 408–409
tools, 402–404
virtual memory, 11–13
Virtual PC, 120, 126
virtual private networks (VPNs), 112, 418
`VirtualProtect` function, 139–141, 177, 221, 362–363
`VirtualQuery` function, 176, 434
viruses, xxiv
VirusShare, 438
VirusSign, 438
VirusTotal, xxix, 24–25, 31, 39, 119, 437–438
Visual Basic for Applications (VBA), 290
VMs. *See* virtual machines
VMProtect, 346
VMRay, 31, 356
VMware
Carbon Black, 233
creating a virtual machine in, 391, 393–400, 404–406
default IP address range, 109
devices, 90
drivers, 78, 89
Fusion, 390–391
Guest Authentication Service, 84
hardening, 412–414
IO ports, 124–125
mutexes, 85–86
paths, 79
pipes, 87
prefixes, 111, 397
process names, 77
registry keys, 81–82
snapshots, 408–409
strings, 108, 125, 410
Tools Service, 84, 402–404
updating, 419
vulnerabilities in, 146
Workstation, 80, 86–87, 89–90, 125, 390–391
VMwareCloak, 403, 410
VMX (*.vmx*) file, 413
VMX IO port, 124–125
volatility, 279–280
Volume Shadow Copy Service (VSS), 255–256
`vpcext` instruction, 126
VPNs (virtual private networks), 112, 418
vulnerabilities, exploiting for privilege elevation, 255–256
vx-underground, 438

W

wallpaper enumeration, 98
Walters, Aaron, 436
Warzone RAT, 26
watch expressions, 192
web proxy, 39
wide (aka unicode) strings, 27
windows, desktop, 98
Windows API (WinAPI), 4, 6–9
formats, 8
suffixes in function names, 8
Windows Command Line, 300. *See also cmd.exe*
Windows event logs, 312–315
Windows Event Viewer, 312
Windows image downloading, 391–392
Windows Management Instrumentation (WMI), 81, 231, 236, 279, 303–304
Windows Management Instrumentation Console (WMIC), 29, 83, 102, 304
Windows Native API
interaction with WinApi, 7
suffixes in function names, 8
Windows PE loader, 13
Windows registry
configuration for concealment, 409–410
enumeration, 80–83
registry keys, VirtualBox, 82
registry keys, VMware, 81
hives, 16
malware behavior analysis, 38
Registry Editor (Regedit), 17
registry-resident malware, 291–292
values, 16
Windows Script Host, 300
Windows Service Controller, 270

Windows Startup Repair, 316
Windows subsystem process, 249
Windows updates, 401
Windows Update service, 237
Winhttp.dll, 8, 30, 142
WinObj, 88
Winusb.dll, 107
wipers, xxiii
Wireshark, 36, 39
WMI (Windows Management Instrumentation), 81, 231, 236, 279, 303–304
WMIC (Windows Management Instrumentation Console), 29, 83, 102, 304
`WNetGetProviderName` function, 79
words (data type), 44
worms, xxiv
`Wow64SetThreadContext` function, 176, 429
`Wow64SuspendThread` function, 207, 433
`WriteFile` function, 30, 37
`WriteProcessMemory` function, 30, 175, 199–201, 203, 206–208, 220–221, 296, 434
wscript.exe, 300

X

x64 (64-bit) assembly code, 44, 47, 192–193
  64-bit SEH, 192–193
x64dbg. *See also* debugging
  adding a watch expression to, 192
  analyzing with, 62–68
  breakpoint settings, 175, 365
  dumping memory to file, 375
  patching, 68–69
  plug-ins, 181–183, 358, 375, 382
  starting a session in, 58–62
  and system artifacts, 172–173
  using with disassembler, 160
x86 (32-bit) assembly code, 44, 47
XDR (extended detection and response), 232. *See also* endpoint detection and response
XMM registers, 126
XOR (exclusive or), 327–331, 340
XORSearch, 329

Y

Yara, 24, 26–27, 32, 405
Yason, Mark Vincent, 180, 437
Yehoshua, Nir, 247, 437
Yomi, 437
Yosifovich, Pavel, 436

Z

zero flag (ZF), 47–50
Zeus, 309
`ZwCreateFile` function, 213
`ZwCreateTransaction` function, 213
`ZwNotifyChangeKey` function, 8