INDEX

A

Abbati, Arnaud, 117
ABI (application binary interface), 128
Activity Monitor tool, 64
adware
  adware-related hijacks and injections, 54–56
  Bitdefender Adware Removal Tool, 49
  Crossrider, 43
  5mLen, 70, 80–81
  Pitchofcase, 40
`aevt_decompile` decompiler, 87
Amnesty International, 63
analysis. *See* dynamic analysis; static analysis
analysis tool detection
  detecting debugger, 209–210
  preventing debugging, 210–211
  targeting security tools, 208–209
analyzing EvilQuest malware
  anti-analysis logic, 233–242
  command line parameters, 231–233
  confirming file type, 223–224
  extracting contents, 224–225
  extracting embedded information, 229–231
  file encryption logic, 275–277
  file exfiltration logic, 271–275
  infection vector, 221–223
  local viral infection logic, 253–263
  persistence, 243–252
  remote communications logic, 263–271
  repersistence logic, 252–253
  Suspicious Package utility, 225–228
analyzing scripts
  AppleScript, 82–88
  bash shell scripts, 76–78
  Perl scripts, 88–89
  Python scripts, 78–82
anti-analysis logic, 176–177
  anti-dynamic-analysis approaches, 204–216
  anti-static-analysis approaches, 188–204
  environmentally generated keys, 216–217
  EvilQuest malware, 233–242
    debugging-thwarting logic, 234–238
    obfuscated strings, 238–242
    virtual machine-thwarting logic, 233–234
  overview, 187
anti-dynamic-analysis approaches
  analysis tool detection, 208–211
    detecting debugger, 209–210
    preventing debugging, 210–211
    targeting security tools, 208–209
  bypassing, 211–216
  execution environment, modifying, 213
  modifying instruction pointer, 214–215
  modifying register value, 216
  patching binary image, 213–214
  virtual machine detection, 204–207
    counting logical and physical CPUs, 206
    MAC address, checking, 207
    overview, 204–205

anti-dynamic-analysis
approaches *(continued)*
SIP status, 208
system model name, checking, 205–206
anti-infection protection mechanisms
application notarization requirements, 4–5
File Quarantine, 4
Gatekeeper, 4
anti-static-analysis approaches
code-level obfuscations, 199–204
binary encryptors, 202–204
overview, 199–201
packers, 201–202
string-based obfuscation, 188–199
encrypted strings, 189–191
finding deobfuscation code, 193–194
forcing malware to execute decryption routine, 197–199
locating obfuscated strings, 191–192
sensitive strings disguised as constants, 188–189
via Hopper script, 194–196
Apparency application, 91–92
`append_ei` function, EvilQuest malware, 257–259
Appify tool, 115, 116–117
Apple Disk Images (.dmg), 72
AppleJeus malware, 75–76, 135, 151–152
AppleScript, 82–88
Apple Silicon, 102, 126
application binary interface (ABI), 128
application bundles
Contents/_CodeSignature file, 93
Contents/ directory, 93
Contents/Info.plist file, 93–95
Contents/MacOS directory, 93
Contents/Resources directory, 93
defined, 91
WindTail malware, 91–95
applications
Apparency application, 91–92
application bundles, 91–95
application notarization requirements, 4–5
cracked, 9–10
fake, 7–8
pirated, 9–10
trojanized, 8–9
Art of Computer Virus Research and Defense, The (Szor), 253
assembly, 126–130
calling conventions, 127–128
defined, 126
instructions, 127
`objc_msgSend` function, 128–130
overview, 126
registers, 126–127
assembly (ASM) mode, Hopper, 146

B

`backtrace` command, LLDB debugger, 175
bash shell scripts, 76–78
binary analysis
extracting nonbinary components, 116–122
Mach-O binary file format, 99–114
classifying, 107–114
data segments, 106–107
header, 100–103
load commands, 103–106
overview, 99–100
tools used to build binaries, 115–116
binary encryptors, 202–204
binary modifications, 42–43
BirdMiner malware, 10, 56, 155–156
BitcoinMagazine-Quidax _ InterviewQuestions_2018 document, 51
Bitdefender Adware Removal Tool, 49
`breakpoint` command, LLDB debugger, 174
`breakpoint delete` command, LLDB debugger, 174
`breakpoint enable/disable` command, LLDB debugger, 174
`breakpoint list` command, LLDB debugger, 174
`breakpoint modify` command, LLDB debugger, 174

breakpoints
 adding commands to, 173–174
 conditionally triggering, 172–173
 defined, 170
 managing, 174
 overview, 170–171
 setting on method names, 172
brute-forcing, 17
`__bss` section, Mach-O binary `__DATA` segment, 106
Bundlore malware, 117

C

calling conventions, assembly, 127–128
capabilities
 adware-related hijacks and injections, 54–56
 categorizing, 47–48
 cryptocurrency miners, 56–57
 defined, 1
 file encryption, 61–62
 memory execution, 58–59
 privilege escalation
  root privileges, 52–54
  sandboxes, 50–51
 reconnaissance logic, 48–49
 remote download/upload, 59–61
 remote process, 58
 remote shells, 57–58
 spyware, 64–65
 stealth, 62–64
 surveys, 48–50
`carve_target` function, EvilQuest malware, 276
C/C++ disassembly, 135–137
certificate file exfiltration, 272–275
CFG (control flow graph) mode, Hopper, 146
`CFStringGetCString` function, 137
Checkm8 exploit, 20
`class-dump` utility, 113–114
code-level obfuscations
 binary encryptors, 202–204
 defined, 188
 overview, 199–201
 packers, 201–202
code-signing authorities, 109
`codesign` utility, 108–111
Cohen, Frederick, 253
ColdRoot malware, 154
commands, adding to breakpoints, 173–174
`construct_plist_path` function, EvilQuest malware, 251
`__const` section, Mach-O binary `__TEXT` segment, 106
Contents/_CodeSignature file, 93
Contents/ directory, 93
Contents/Info.plist file, 93–95
Contents/MacOS directory, 93
Contents/Resources directory, 93
`continue` command, LLDB debugger, 169
control flow disassembly, 137–139
control flow graph (CFG) mode, Hopper, 146
CookieMiner malware, 56, 157
CPUMeaner malware, 56, 73–74
`cputype` member, Mach-O header, 101
cracked applications, 9–10
CreativeUpdate malware, 16, 56, 109, 117–118
cron jobs, 32–33
cross-references, Hopper, 144–145
Crossrider adware, 43
cryptocurrency
 cryptocurrency file exfiltration, 272–275
 uncovering cryptocurrency mining in App Store app, 180–185
cryptocurrency miners, 56–57
`_cstring` section, Mach-O binary `__TEXT` segment, 106
`CTRL-C` command, LLDB debugger, 169
custom URL schemes, 10–13
Cylance antivirus firm, 59

D

Dacls backdoor malware, 133–134, 159–160
DarthMiner malware, 56
`__data` section, Mach-O binary `__DATA` segment, 106
`__DATA` segment, Mach-O binary file format, 104, 107
`dataWithBytes:length:` method, Objective-C disassembly, 130

debugging
  LLDB debugger, 167–180
    breakpoints, 170–174
    controlling execution, 169–170
    displaying runtime information, 174–176
    modifying process state, 176–178
    overview, 167–168
    scripts, 178–180
    starting session, 168–169
  overview, 165
  power of, 166–167
  uncovering cryptocurrency mining in App Store app, 180–185
debugging-thwarting logic, 234–238
decompilation, 139–140
Decompiler tool, 80
Devadoss, Dinesh, 222, 275
DevilRobber malware, 84
directory listing exfiltration, 271–272
disassembly
  C/C++, 135–137
  control flow, 137–139
  Objective-C, 130–133
  Swift, 133–135
display modes, Hopper, 145–146
distribution packaging, extracting malicious files from
  Apple Disk Images, 72
  packages, 73–76
.dmg (Apple Disk Images) file extension, 72
.docm file extension, 89–90
Dok malware, 5
DoubleFantasy malware, 194–196
Dummy malware, 57
dumping (printing), debugging process, 166–167, 174
`DYLD_*` environment variables, 35–36
dylibs. *See* dynamic libraries
dynamic analysis
  debugging, 165–185
    LLDB debugger, 167–180
    overview, 165
    power of, 166–167
    scripts, 178–180
    uncovering cryptocurrency mining in App Store app, 180–185
  defined, 67
  file monitoring, 153–156
    FileMonitor utility, 155–156
    `fs_usage` utility, 154–155
    overview, 153–154
  network monitoring, 157–163
    Netiquette utility, 159–160
    network status monitors, 158–159
    network traffic monitors, 160–163
    overview, 157–158
  overview, 149–150
  process monitoring, 150–153
    overview, 150–151
    ProcessMonitor utility, 151–153
dynamic libraries (dylibs), 34–39
  `DYLD_*` environment variables, 35–36
  dylib hijacking, 37–39
  dylib proxying, 36–37
  overview, 34–35

E

EFI (Extensible Firmware Interface) exploits, 20
`ei_carver_main` function, EvilQuest malware, 275
`ei_forensic_sendfile` function, EvilQuest malware, 274
`ei_forensic_thread` function, EvilQuest malware, 271
`ei_loader_main` function, EvilQuest malware, 254
`ei_pers_thread` function, EvilQuest malware, 252
`ei_rfind_cnc and ei_getip` function, EvilQuest malware, 281
`ei_selfretain_main` function, EvilQuest malware, 249–250
`ei_str` function, EvilQuest malware, 239–240
8-bit registers, 127
`eiht_get_update` function, EvilQuest malware, 262–263

Eleanor malware, 150–151
Electron tool, 115–116, 120
embedded information, extracting, 229–231
Endpoint Security framework, 151
environmentally generated keys, 216–217
Equation Group, 217
ESET antivirus company, 10
Esser, Stefan, 54
event monitor rules, 41
EvilQuest malware, 43, 52, 62, 215
  analyzing
    anti-analysis logic, 233–242
    command line parameters, 231–233
    confirming file type, 223–224
    extracting contents, 224–225
    extracting embedded information, 229–231
    file encryption logic, 275–277
    file exfiltration logic, 271–275
    infection vector, 221–223
    local viral infection logic, 253–263
    persistence, 243–252
    remote communications logic, 263–271
    repersistence logic, 252–253
    Suspicious Package utility, 225–228
  invoking string decryption routine, 197–199
  updates, 277–281
exfiltration, 59–61
  certificate and cryptocurrency exfiltration, 272–275
  directory listing exfiltration, 271–272
exploits
  Checkm8, 18–19
  defined, 18
  EFI, 18–19
  zero-day, 18–19
Extensible Firmware Interface (EFI) exploits, 20
externally facing services, 17–18

F

fake applications, 7–8
fake security alerts, 6–7
fake updates, 7
`fat_header`, Mach-O header, 102
`52M_rj` function, EvilQuest malware, 278
`file` command
  identifying byte-compiled Python script with, 70–71
  identifying Office documents, 89
file encryption logic, 61–62, 275–277
file exfiltration logic, EvilQuest malware analysis
  certificate and cryptocurrency file exfiltration, 272–275
  directory listing exfiltration, 271–272
file monitoring
  FileMonitor utility, 155–156
  `fs_usage` utility, 154–155
  overview, 153–154
FileMonitor utility, 155–156
File Quarantine, 4
`filetype` member, Mach-O header, 101
`file` utility, 228
  EvilQuest malware, 223
  Mach-O header, 103
  WindTail malware, 71
FindCrypt plug-in, 194
FinFisher malware, 172
`finish` command, LLDB debugger, 169
FinSpy, 63
FireEye, 53
Flashback malware, 18
Flash zero-day, 19
FruitFly malware, 17, 64, 88–89, 151
`fs_usage` utility, 12, 154–155

G

garbage (spurious) instructions, 200
Gatekeeper, 4
`get_mediator` function, EvilQuest malware, 263–265
`get_targets` function, EvilQuest malware, 254, 272–274, 275
`getDeviceSerial` function, 135–136, 137, 139

GMERA, 63
Grant, Ari, 172
GravityRAT malware, 118–121

H

HackingTeam cyberespionage company, 19
Handbrake application, 16
handlers, AppleScript, 87
header, Mach-O binary file format, 100–103
- `cputype` member, 101
- `fat_header`, 102
- `filetype` member, 101
- `file` utility and, 103
- `lipo` tool and, 102–103
- `magic` member, 100–101
- `offset` member, 102
- `otool` utility and, 101

hexadecimal mode, Hopper, 146
`-h` flag, Mach-O header, 101
Homebrew package manager, 6
Hopper, 125, 139
- cross-references, 144–145
- Inspector view, 141–142
- Proc view, 142
- reverse engineering with, 140–146
  - creating binary to analyze, 140–141
  - display modes, 145–146
  - interface, 141–143
  - loading binary, 141
  - viewing disassembly, 143–145
- string-based obfuscation via Hopper script, 194–196
- Str view, 142

`hostname` command, 89
`--ignrp` command line parameter, EvilQuest malware, 233

I

infection vectors
- anti-infection protection mechanisms, 4–5
- cracked applications, 9–10
- custom URL schemes, 10–13
- defined, 1
- EvilQuest malware, 221–223
- exploits, 18–20
- externally facing services, 17–18
- fake applications, 7–8
- fake security alerts, 6–7
- fake updates, 7
- malicious emails, 5–6
- Office macros, 14–15
- overview, 3–4
- physical attacks, 19
- pirated applications, 9–10
- supply chain attacks, 16
- trojanized applications, 8–9
- Xcode projects, 15–16

Info.plist file
- defined, 11
- WindTail malware, 93–95

Inspector view, Hopper, 141–142
Intego security company, 7
interactive shells, 57
IOServiceGetMatchingService function, 136–137
IPStorm malware, 17–18
Iran Threats blog, 49
`is_debugging` function, EvilQuest malware, 235–236
`is_executable` function, EvilQuest malware, 255–257
iWorm malware, 10

J

jobs, 33

K

KeRanger ransomware, 61, 213–215
keychains, 49
`kill _unwanted` function, EvilQuest malware, 244
KnockKnock open source utility, 44–45
Komplex malware, 58, 130, 138, 210

L

launch agents, 26–32
launch daemons, 26–32
Lazarus Group, 58, 61, 91, 133, 217
- AppleJeus, 75–76, 135, 151–152
- Dacls backdoor malware, 133–134, 159–160
- JMTTrader.app, 8–9

LC_LOAD_DYLIB load command, Mach-O binary file format, 105–106
LC_MAIN load command, Mach-O binary file format, 104–105
LC_SEGMENT_64 load command, Mach-O binary file format, 104
lfsc_dirlist function, EvilQuest malware, 272
__LINKEDIT segment, Mach-O binary file format, 104
lipo tool, Mach-O header, 102–103
Little Snitch firewall, 49, 209
LLDB debugger
    breakpoints, 170–174
        adding commands to, 173–174
        conditionally triggering, 172–173
        defined, 170
        managing, 174
        overview, 170–171
        setting on method names, 172
    controlling execution, 169–170
    examining runtime information, 174–176
    modifying process state, 176–178
    overview, 167–168
    starting session, 168–169
load commands, Mach-O binary file format
    LC_LOAD_DYLIB, 105–106
    LC_MAIN, 104–105
    LC_SEGMENT_64, 104
    overview, 103–104
local viral infection logic
    checking which files to infect, 255–257
    EvilQuest malware analysis, 253–263
    executing original code of infected file, 262–263
    executing and repersisting from infected files, 260–262
    infecting target files, 257–260
    listing candidate files for infection, 254
    overview, 253–254
login items, 24–26
login/logout hooks, 34
LoudMiner malware, 10
lsof utility, 159

M

MacDownloader malware, 49, 60
Mach-O binary file format, 76, 95
    classifying, 107–114
        code-signing information, 109–111
        hashes, 107–109
        Objective-C class information, 113–114
        strings, 112–113
    header, 100–103
        cputype member, 101
        fat_header, 102
        filetype member, 101
        file utility and, 103
        lipo tool and, 102–103
        magic member, 100–101
        offset member, 102
        otool utility and, 101
    load commands, 103–106
        LC_LOAD_DYLIB, 105–106
        LC_MAIN, 104–105
        LC_SEGMENT_64, 104
        overview, 103–104
    overview, 99–100
    segments, 106–107
        __DATA segment, 107
        __TEXT segment, 107
        __LINKEDIT segment, 104
MachOView utility, 101
mach port, 136
macOS Catalina (10.15), 4–5
MacRansom, 205–206
macros
    extracting, 89–90
    macro-based attacks, 14–15
MacUpdate website, 56, 117
magic member, Mach-O header, 100–101
malicious emails, 5–6
Malwarebytes, 54
Mami malware, 166
MD5 hash, 107–108
memory execution, 58–59
memory write command, LLDB debugger, 177–178

MH_BUNDLE (0x8) value, Mach-O header, 101
MH_DYLIB (0x6) value, Mach-O header, 101
MH_EXECUTE (0x2) value, Mach-O header, 101
MinerGate, 118
mnemonics, assembly instructions, 127
Mokes malware, 64, 159
Mughthesec malware, 54, 207, 213

N

name mangling, 133, 135
Netcat utility, 150
Netiquette utility, 60, 159–160
netstat utility, 158
nettop utility, 158
NetWire malware, 191–192
network monitoring
  Netiquette utility, 159–160
  network status monitors, 158–159
  network traffic monitors, 160–163
  overview, 157–158
nexti command, LLDB debugger, 169
nm utility, EvilQuest malware, 229
nonbinary analysis
  analyzing scripts, 76–89
    AppleScript, 82–88
    bash shell scripts, 76–78
    Perl scripts, 88–89
    Python scripts, 78–82
  applications, 91–95
  extracting malicious files from distribution packaging, 72–76
    Apple Disk Images, 72
    packages, 73–76
  identifying file types, 70–72
  Office documents, 89–91
  overview, 69–70
non-interactive shells, 57
nonoperations (NOPs), 200
NSData class method, Objective-C disassembly, 131–132
NSData object, 131–132
NSTask launch method, 134
Nygard, Steve, 113

O

obfuscated scripts, 199–204
obfuscated strings
  EvilQuest malware, 197–199, 238–242
  locating, 191–192
__objc_* section, Mach-O binary __DATA segment, 106
objc_msgSend function, 172, 178–179, 182–183, 193
  assembly, 128–130
  Swift disassembly, 133
Objective-C disassembly, 130–133
Office macros
  extracting, 89–90
  macro-based attacks, 14–15
offset member, Mach-O header, 102
oletools toolset, 89–90
olevba utility, 89–90
operands, 127
organizationally unique identifier (OUI), 207, 213
osadecompile command, 82
OSAMiner malware, 56, 85–88
OS X Leopard, 4
OS X Mountain Lion, 4
otool utility, 101–102, 105
OUI (organizationally unique identifier), 207, 213

P

packages (*.pkg*), 73–76
packers, 201–202
patch binary
  disassembling, 231
  extracting embedded information from, 229–231
periodic scripts, 33–34
Perl scripts, 88–89
persist_executable_frombundle function, EvilQuest malware, 261
persist_executable function, EvilQuest malware, 246
persistence
  application and binary modifications, 42–43
  defined, 1, 23

dynamic libraries, 34–39
DYLD_* environment variables, 35–36
dylib hijacking, 37–39
dylib proxying, 36–37
overview, 34–35
event monitor rules, 41
EvilQuest malware analysis, 243–252
copies as launch items, 247–249
copy operation, 246–247
killing unwanted processes, 244–246
overview, 243–244
starting launch items, 249–252
KnockKnock open source utility, 44–45
launch agents, 26–32
launch daemons, 26–32
login items, 24–26
login/logout hooks, 34
overview, 23–24
plug-ins, 39–40
relaunch applications, 41–42
scheduling mechanisms, 32–34
cron jobs, 32–33
at jobs, 33
periodic scripts, 33–34
scripts, 41
physical attacks, 19
Pirate Bay website, 10
pirated applications, 9–10
Pirrit malware, 102, 200–201, 208
Pitchofcase adware, 40
.pkg (packages), 73–76
pkgutil utility, EvilQuest malware, 225
Platypus tool, 76–77, 115, 117–118
plug-ins, 39–40
prevent_trace function, EvilQuest malware, 237–238
print command, LLDB debugger, 175
printing (dumping), debugging process, 166–167, 174
privilege escalation
root privileges, 52–54
sandboxes, 50–51
process monitoring
overview, 150–151
ProcessMonitor utility, 151–153
Proc view, Hopper, 142
Proton malware, 16, 49, 208–209
pseudocode mode, Hopper, 146
ptrace system call, preventing debugging with, 210–211
PyInstaller Extractor tool, 119
PyInstaller tool, 115, 119
Python scripts, 78–82

R

ransomware. *See also* EvilQuest malware
defined, 23
KeRanger ransomware, 61, 213–215
RBP register, 127
RCX register, 131–132
RDX register, 131–132, 231
react_exec function, EvilQuest malware, 266–267
react_host function, EvilQuest malware, 270
react_keys function, EvilQuest malware, 269
react_ping function, EvilQuest malware, 270, 280
react_sav function, EvilQuest malware, 268
react_scmd function, EvilQuest malware, 270–271
react_start function, EvilQuest malware, 268
react_updatesettings function, EvilQuest malware, 281
Reaves, Jason, 277
reconnaissance logic, 48–49
Reed, Thomas, 43, 54, 207, 222
registers
assembly, 126–127
R8 register, 269
R9 register, 183
RAX register, 31, 126–128, 130–131, 136–137, 175–177, 190, 200, 211, 216, 236, 248, 239
RBP register, 127
RCX register, 131–132
RDX register, 131–132, 231
RSI register, 131–132, 129, 136–137, 179–180, 193
RSP register, 127

registers *(continued)*
    modifying register value, 216
    "scratch," 136
register write command, LLDB
        debugger, 176–177
relaunch applications, 41–42
remote communications logic,
        EvilQuest malware analysis
    get_mediator function, 263–265
    remote tasking logic, 265–271
remote download/upload, 59–61
remote process, 58
remote services, compromising, 17–18
remote shells, 57–58
remote tasking logic, EvilQuest
        malware, 265–271
    overview, 265–266
    react_exec function, 266–267
    react_host function, 270
    react_keys function, 269
    react_ping function, 270
    react_sav function, 268
    react_scmd function, 270–271
    react_start function, 268
repersistence logic, EvilQuest malware
        analysis, 252–253
reverse engineering
    creating binary to analyze, 140–141
    display modes, 145–146
    interface, 141–143
    loading binary, 141
    viewing disassembly, 143–145
Riordan, James, 217
root privileges, 52–54
RSI register, 131–132, 129, 136–137,
        179–180, 193
RSP register, 127
run_audio and run_image function,
        EvilQuest malware, 281
run_daemon function, EvilQuest
        malware, 250–251
run_target function, EvilQuest
        malware, 262–263
run command, LLDB debugger, 169
run-only AppleScript files, 84–87
runtime information, displaying, 174–176

S

s_is_high_time function, EvilQuest
        malware, 275
Safe Finder, 54–55
sandboxes, 50–51
scheduling mechanisms
    cron jobs, 32–33
    at jobs, 33
    periodic scripts, 33–34
Schneier, Bruce, 217
"scratch" registers, 136
Script Editor, 82, 84
scripts, 41
    AppleScript, 82–88
    bash shell, 76–78
    Perl, 88–89
    Python, 78–82
scutil command, 89
SDK files, 100–101
segments, Mach-O binary file format,
        106–107
    __DATA segment, 107
    __TEXT segment, 107
    __LINKEDIT segment, 104
set_important_files function,
        EvilQuest malware, 252–253
setLaunchPath: method, Swift
        disassembly, 133
SHA-1 hash, 107–108
Shlayer malware, 4–5, 116
Siggen malware, 76–80, 94–95
signing certificate, 109
--silent command line parameter,
        EvilQuest malware, 231–232
SIP (System Integrity Protection)
        status, 169, 208
16-bit registers, 127
64-bit registers, 127
spurious (garbage) instructions, 200
spyware, 64–65
stack, defined, 127
static analysis
    assembly, 126–130
    binary analysis, 99–122
        extracting nonbinary
            components, 116–122

Mach-O binary file format, 99–114
tools used to build binaries, 115–116
decompilation, 139–140
defined, 67
disassembly, 130–139
nonbinary analysis, 69–95
analyzing scripts, 76–89
applications, 91–95
extracting malicious files from distribution packaging, 72–76
identifying file types, 70–72
Office documents, 89–91
overview, 69–70
reverse engineering with Hopper, 140–146
stealth, 62–64
`stepi` command, LLDB debugger, 169
stepping through, debugging process, 166
Stokes, Phil, 80, 85, 87, 277
string-based obfuscation
defined, 188
encrypted strings, 189–191
finding deobfuscation code, 193–194
forcing malware to execute decryption routine, 197–199
locating obfuscated strings, 191–192
sensitive strings disguised as constants, 188–189
via Hopper script, 194–196
strings
extracting embedded strings, 112–113
obfuscated strings
EvilQuest malware, 197–199, 238–242
locating, 191–192
`strings` utility
EvilQuest malware, 229–230
WindTail malware, 112–113
Str view, Hopper, 142
sudoers file, 54
supply chain attacks, 16
surveys, 48–50
Suspicious Package utility, 73–75, 225–228
Swift disassembly, 133–135
System Integrity Protection (SIP) status, 169, 208
Szor, Peter, 253

T

`tcpdump` utility, 160
team identifier, 109
`__TEXT` segment, Mach-O binary file format, 104, 106–107
ThiefQuest malware, 278. *See also* EvilQuest malware
32-bit registers, 127
Thomas, Adam, 54
Tor utility, 150
Trend Micro, 15, 278–279, 281
trojanized applications, 8–9
typosquatting, 6

U

universal binaries, 102
UPX packer, 201, 213
user-assisted infections. *See also* infection vectors
anti-infection protection mechanisms, 4–5
malicious emails, 5–6
`/usr/bin/osascript` command, AppleScript, 82

V

VBA (Visual Basic for Applications), 14
`-v` flag, Mach-O header, 101
Vilaça, Pedro, 202
virtual machine detection
counting logical and physical CPUs, 206
MAC address, checking, 207
overview, 204–205
SIP status, checking, 208
system model name, checking, 205–206
virtual machine-thwarting logic, 233–234
viruses
checking which files to infect, 255–257

viruses *(continued)*
  defined, 253
  infected files
    executing and repersisting from, 260–262
    executing original code of, 262–263
  infecting target files, 257–260
  listing candidate files for infection, 254
VirusTotal antivirus scanning portal, 108
Visual Basic for Applications (VBA), 14
VMware, 207
VST Crack website, 10

W

Wacaw tool, 150
WhatsAppService.app, 76–77
WhatsYourSign (WYS) tool, 71–73
`whoami` command, 89
WindTail malware, 10–13, 59–61, 70, 91–95
  Info.plist file, 93–95
  key/value pairs, 94
  `strings` utility, 112–113
Wireshark application, 161–163
`writeconfig.xpc` service, 53
`writeToFile:atomically:` method, Objective-C disassembly, 132
WYS (WhatsYourSign) tool, 71–73

X

XAgent malware, 114
`x/b` command, LLDB debugger, 175
Xcode projects, 15–16
XCSSET malware, 15–16
`x/i` command, LLDB debugger, 175
`x/s` command, LLDB debugger, 175
XSLCmd malware, 53

Y

Yort malware, 91

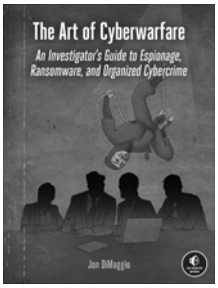
Z

zero-day exploits, 18–19
ZuRu malware, 106

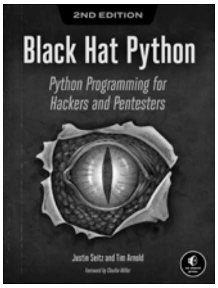
RESOURCES

Visit *https://nostarch.com/art-mac-malware/* for errata and more information.

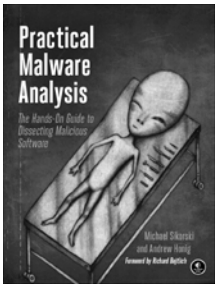
More no-nonsense books from NO STARCH PRESS



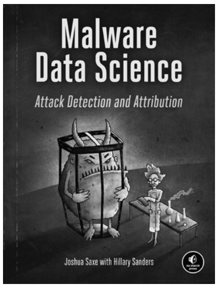
THE ART OF CYBERWARFARE
An Investigator's Guide to Espionage, Ransomware, and Organized Cybercrime
BY JON DIMAGGIO
241 PP., $39.99
ISBN 978-1-71850-214-7



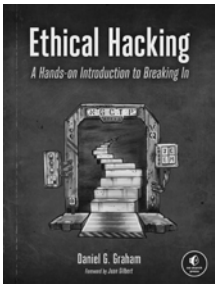
BLACK HAT PYTHON, 2ND EDITION
Python Programming for Hackers and Pentesters
BY JUSTIN SEITZ AND TIM ARNOLD
216 PP., $44.99
ISBN 978-1-71850-112-6



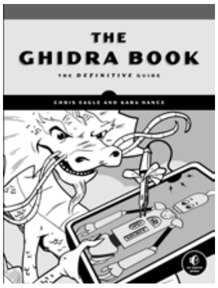
PRACTICAL MALWARE ANALYSIS
The Hands-On Guide to Dissecting Malicious Malware
BY MICHAEL SIKORSKI AND ANDREW HONIG
800 PP., $59.95
ISBN 978-1-59327-290-6



MALWARE DATA SCIENCE
Attack Detection and Attribution
BY JOSHUA SAXE AND HILLARY SANDERS
272 PP., $49.95
ISBN 978-1-59327-859-5



ETHICAL HACKING
A Hands-on Introduction to Breaking In
BY DANIEL G. GRAHAM
376 PP., $44.99
ISBN 978-1-71850-187-4



THE GHIDRA BOOK
The Definitive Guide
BY CHRIS EAGLE AND KARA NANCE
608 PP., $59.95
ISBN 978-1-71850-102-7

PHONE:
800.420.7240 OR
415.863.9900

EMAIL:
SALES@NOSTARCH.COM

WEB:
WWW.NOSTARCH.COM

Never before has the world relied so heavily on the Internet to stay connected and informed. That makes the Electronic Frontier Foundation's mission—to ensure that technology supports freedom, justice, and innovation for all people—more urgent than ever.

For over 30 years, EFF has fought for tech users through activism, in the courts, and by developing software to overcome obstacles to your privacy, security, and free expression. This dedication empowers all of us through darkness. With your help we can navigate toward a brighter digital future.

ELECTRONIC FRONTIER FOUNDATION EFF

LEARN MORE AND JOIN EFF AT EFF.ORG/NO-STARCH-PRESS

"An invaluable resource for anyone looking to level up their skills."

—Maria Markstedter (@Fox0x01), founder of Azeria Labs and *Forbes* Person of the Year in Cybersecurity

Defenders must fully understand how malicious software works if they hope to stay ahead of the increasingly sophisticated threats facing Apple products today. *The Art of Mac Malware* is a comprehensive handbook for cracking open these malicious programs and seeing what's inside.

Discover the secrets of nation-state backdoors, destructive ransomware, and subversive cryptocurrency miners as you uncover their infection methods, persistence strategies, and insidious capabilities. Then work with and extend foundational reverse-engineering tools to extract and decrypt embedded strings, unpack protected Mach-O malware, and even reconstruct binary code. Next, using a debugger, you'll execute the malware, instruction by instruction, to discover exactly how it operates. In the book's final section, you'll put these lessons into practice by analyzing a complex Mac malware specimen on your own.

You'll learn to:

- Recognize common infection vectors, persistence mechanisms, and payloads leveraged by Mac malware
- Triage unknown samples in order to quickly classify them as benign or malicious
- Work with static analysis tools, including disassemblers, in order to study malicious scripts and compiled binaries
- Leverage dynamic analysis tools, such as monitoring tools and debuggers, to gain further insight into sophisticated threats
- Quickly identify and bypass anti-analysis techniques aimed at thwarting your analysis attempts

A current leader in the field of macOS threat analysis, Patrick Wardle uses real-world examples pulled from his original research. *The Art of Mac Malware* is the definitive resource to battling these increasingly prevalent and insidious Apple-focused threats.

About the Author

PATRICK WARDLE is the founder of Objective-See, a nonprofit that creates open source macOS security tools. He spends his time finding Apple zero-days, analyzing macOS malware, speaking at global security conferences, and writing tools to protect Mac users around the world. He has also worked at NASA and the NSA, making him intimately familiar with aliens, spies, and talking nerdy.

no starch press

THE FINEST IN GEEK ENTERTAINMENT™

www.nostarch.com

$49.99 ($65.99 CDN)

ISBN 978-1-7185-0194-2

54999

9 781718 501942