

INDEX

A

- access violation handlers, 60
- AccessViolationHook, 72
- accumulator register. *See* EAX register
- AddBpt() function, 158
- AllExceptHook, 71
- analysis, automated static, 122
- anti-debugging routines in malware, 81
- appliances, VMware, 2
- associating processes, debuggers, 25–33
- attaching processes, 26
- attacks, format string, 114
- automated static analysis, 122

B

- base pointer, EBP register, 15
- binary data, Sulley primitives, 126
- black-box debuggers, vs. white-box, 13
- blocks, Sulley primitives, 127
- BpHook, 71
- breakpoints, 18–24, 43–55
 - handlers, 58
 - hardware, 21, 47–52
 - memory, 23, 52–55
 - soft, 19, 43
- buffer overflows, 112
- bypassing, DEP on Windows, 77

C

- calling conventions, 7
- C datatypes, constructing, 8
- cdecl convention, 7
- characters, filtering exploit strings, 75
- chunks() function, 156
- classes
 - PyCPU, 164
 - PyEmu, 165
 - PyMemory, 165
- code injection, 101
- CodeRefsFrom() function, 157
- CodeRefsTo() function, 157
- codes
 - debug events, 39
 - IOCTL dispatch routine, 145
- compiling, with *py2exe* library, 108
- compressors. *See* executable packers, *IdaPyEmu*
- constructing, C datatypes, 8
- conventions, calling, 7
- count register. *See* registers
- CPU registers, state, 33–38
- crash handlers, creating, 62
- CRC (cyclic redundancy check), 21
- CreateFileW call, 138
- CreateProcessA() function, 26
- CreateProcessHook, 72
- CreateRemoteThread() function, 98
- CreateThreadHook, 72

CreateToolhelp32Snapshot()
function, 34
cross-references, 156
C runtime, resolving printf()
function, 6
ctypes library, 5
CUP registers, 14
cyclic redundancy check (CRC), 21

D

Data Execution Prevention (DEP),
bypassing on Windows, 77
data generation, block-based
fuzzing, 123
data register. *See* EDX register
DataRefsFrom() function, 157
datatypes, C, 8
DataRefsTo() function, 157
debug events, 18
handlers, 39–42
registers, breakpoint styles, 21
debuggers, 13–24. *See also* Windows
debuggers
breakpoints, 18–24
debug events, 18
general-purpose CPU
registers, 14
hooks, 157
stacks, 16
defining, structures and unions, 9
delimiters, Sulley primitives, 125
DEP (Data Execution Prevention),
bypassing on Windows, 77
destination index, EDI register, 15
development environment, 1–11
obtaining and installing
Python 2.5, 2
operating system requirements, 2
setting up Eclipse and PyDev,
4–11
devices, discovering names, 143
DeviceToControl API, 139
disabling, DEP, 77
discovering device names, 143

DLL (dynamically linked libraries)
defined, 6
injection, 97–110
remote thread creation,
97–103
sample application, 104–110
loading, 99
driverlib library, 142–147
drivers, fuzzing Windows, 137–151
dwCreationFlags parameter
(CreateRemoteThread()
function), 98
dwDebugEventCode, 39
dwDesiredAccess parameter
(OpenProcess()
function), 29
dwFlags parameter
(CreateToolhelp32Snapshot()
function), 34
dwIoControlCode parameter
(DeviceToControl API), 139
dwStackSize parameter
(CreateRemoteThread()
function), 98
dynamically linked libraries. *See* DLL
(dynamically linked
libraries)

E

EAX register, 15
EBX register, 16
Eclipse
running scripts with, 5
setting up, 4–11
ECX register, 15
EDI register, 15
EDX register, 15
emulation, of functions in
IdaPyEmu, 172
encrypted traffic, sniffing, 86
endian keyword, 127
environment. *See* development
environment
ESI register, 15

- events
 - debug, 18
 - exception, 41
 - handlers, 39–42
- exception events, 41
- exception handlers, 167
- executable memory, CLC, 21
- executable packers, *IdaPyEmu*, 176
- execution
 - PyEmu*, 165
 - transferring to shellcode, 73
- ExitProcessHook*, 72
- ExitThreadHook*, 72
- extending, breakpoint handlers in *PyDbg*, 58

F

- FastLogHook*, 72
- file fuzzer, 115–121
- file hiding, DLL injection, 104
- File Transfer Protocol (FTP), *Sulley*, 129
- filtering, exploit strings, 75
- FirstSeg()* function, 155
- format string attacks, 114
- FTP (File Transfer Protocol), *Sulley*, 129
- function emulation, *IdaPyEmu*, 172
- functions. *See also individual function names*
 - finding function cross-references, 158
 - function code coverage, 160
 - IDAPython*, 155–158
 - locating dangerous function calls, 65
- functions()* function, 156
- fuzzing, 111–122
 - automated static analysis, 122
 - bug classes, 112–115
 - code coverage, 122
 - files, 115–121
 - Sulley* web interface, 133–136
 - Windows drivers, 137–151

G

- generation fuzzers, 111
- GetBptQty()* function, 158
- GetFuncOffset()* function, 156
- GetFunctionName()* function, 156
- GetInputFileMD5()* function, 155
- get_memory()* function, 166
- GetRegValue()* function, 158
- get_stack_argument()* function, 166
- get_stack_variable()* function, 166
- GetThreadContext()* function, 35
- global flags (GFlags), 113
- groups, *Sulley* primitives, 127
- guard page permissions, 23

H

- handlers
 - access violation, 60
 - breakpoint, 58
 - crash, 62
 - event, 39–42
 - LoadLibrary*, 180
 - PyEmu*, 166–170
- handles, *ret* instruction handler, 175
- handling soft breakpoints, 43
- hard hooking, *Immunity Debugger*, 90–95
- hardware breakpoints, 21, 47–52
- heap overflows, 113
- hippie *PyCommand*, 91
- hooking, 85–95
 - hard hooking with *Immunity Debugger*, 90–95
 - soft hooking with *PyDbg*, 86–90
- hook types, 71
 - debugger, 157
- hProcess* parameter
 - (*CreateRemoteThread()* function), 98
- hSnapshot* parameter
 - (*CreateToolhelp32Snapshot()* function), 34
- HTTP fuzzing, example, 128

I

- IdaPyEmu, PyEmu, 171–181
- IDAPython, 153–162
 - example scripts, 158–162
 - functions, 155–158
 - installing, 154
- Immunity Debugger, 69–83
 - anti-debugging routines in
 - malware, 81
 - driver fuzzing, 139–142
 - exploit development, 73–81
 - hard hooking, 90–95
 - installing, 2, 70
- impacket library, 124
- indexes, source and destination
 - indexes, 15
- injection. *See* code injection; DLL, injection
- input/output controls (IOCTL),
 - fuzzing Windows
 - drivers, 138
 - dispatch routine, 144
- installing
 - IDAPython, 154
 - Immunity Debugger, 2, 70
 - impacket library, 124
 - PyEmu, 163–181
 - Python 2.5, 2
 - Sulley, 124
 - UPX, 176
 - WinPcap library, 124
- instruction handlers, 168
- integer overflows, 113
- integers, Sulley primitives, 126
- intelligent debugging, 14
- IOCTL (input/output controls),
 - fuzzing windows
 - drivers, 138
 - dispatch routine, 144
- IsDebuggerPresent function, 81

K

- kernel mode, black-box
 - debuggers, 14
- keywords, integer
 - representations, 127

L

- libraries
 - ctypes, 5
 - DLLs, 6
 - Driverlib, 142–147
 - handlers, 167
 - py2exe*, 108
 - WinPcap, 124
- Linux
 - installing Python in, 3
 - using Python in, 2
- LoadDLLHook, 72
- loading, DLLs, 6, 99
- LoadLibrary() function, 99
- LoadLibrary handler, 180
- LocByName() function, 156
- LogBpHook, 71
- lpBytesReturned parameter
 - (DeviceToControl API), 139
- lpFileName parameter (LoadLibrary() function), 99
- lpInBuffer parameter
 - (DeviceToControl API), 139
- lpParameter parameter
 - (CreateRemoteThread() function), 98
- lpStartAddress parameter
 - (CreateRemoteThread() function), 98
- lpThreadAttributes parameter
 - (CreateRemoteThread() function), 98
- lpThreadId parameter
 - (CreateRemoteThread() function), 98

M

- malware, 81
- memory
 - breakpoints, 23, 52–55
 - handlers, 169
 - PyEmu, 165
- metrics, code coverage, 122
- Microsoft Windows. *See* Windows
- modifiers, register, 165
- monitoring, networks and processes with Sulley, 132
- mutation fuzzers, 111
- mutators, example of, 119
- my_debugger_defines.py* file, 30

N

- names, discovering for devices, 143
- networks, monitoring with
 - Sulley, 132
- NextSeg() function, 155
- nInBufferSize parameter
 - (DeviceToControl API), 139
- NtSetInformationProcess() function, 77

O

- one-shot breakpoints, 20
- opcode handlers, 168
- opening processes, 26
- OpenProcess() function, 29
- OpenThread() function, 33
- operating systems, requirements, 2
- overflows
 - buffers, 112
 - integers, 113

P

- page permissions, querying and manipulating, 52
- page size, calculating, 52

- parameters, passing by reference, 9
- PEPyEmu, 175
- permissions, page, 52
- persistent breakpoints, 20
- PostAnalysisHook, 72
- primitives, Sulley, 125–128
- printf() function, 6, 45, 114
- processes
 - associating to debuggers, 25–33
 - attaching, 26
 - disabling DEP, 77
 - inserting shellcode, 101
 - iteration, defeating, 82
 - monitoring, with Sulley, 132
 - opening, vs. attaching, 26
 - snapshots, obtaining, 63
- program counter handler, 170
- py2exe* library, compiling with, 108
- PyCommands, 71
- PyCPU, 164
- PyDbg, 57–68
 - access violation handlers, 60
 - breakpoint handlers, 58
 - process snapshots, 63
 - sample tool, 65–68
 - soft hooking, 86–90
- PyDev, setting up, 4–11
- PyEmu, 163–181
 - defined, 164–170
 - IdaPyEmu, 171–181
 - installing, 164
- PyHooks, 71
- PyMemory, 164
- Python, installing, 2

Q

- querying, page permissions, 52

R

- random primitives, Sulley, 126
- ReadProcessMemory() function, 43
- receive_ftp_banner() function, 131

- registers
 - CPU, 14
 - debug, 21
 - EAX, 15
 - EBX, 16
 - ECX, 15
 - EDI, 15
 - EDX, 15
 - ESI, 15
 - handlers, 167
 - modifiers, PyEmu, 165
- remote thread creation, DLL
 - injection, 98–110
- requirements, for operating systems, 2
- ret instruction handler, 175

S

- s_binary() directive, 126
- ScreenEA() function, 155
- scripted debuggers, advantages
 - of, 18
- scripting, IDAPython, 153–162
- scripts, running from Eclipse, 5
- SegByName() function, 156
- SegEnd() function, 156
- segments, IDAPython, 155
- Segments() function, 156
- SegName() function, 156
- SegStart() function, 156
- servers
 - FTP, 129
 - socket, 110
- sessions, Sulley, 131
- set_memory() function, 166
- SetRegValue() function, 158
- set_stack_argument() function, 166
- set_stack_variable() function, 166
- SetThreadContext() function, 35
- setting soft breakpoints, 43

- shellcode
 - inserting into processes, 101
 - transferring execution to, 73
- signed keyword, 127
- sniffing encrypted traffic, 86
- socket servers, example of, 110
- soft breakpoints, 19, 43
 - CRC, 21
 - PyDbg function for setting, 58
 - setting and handling, 43
- soft hooks
 - defined, 85
 - PyDbg, 86–90
- source indexes, ESI register, 15
- s_random() directive, 126
- stacks, 16
 - overflows, 112
 - pointers, ESP register, 15
 - size, calculating, 161
- state, CPU registers, 33–38
- static analysis, automated, 122
- static primitives, Sulley, 126
- stdcall convention, 7
- STDCALLFastLogHook, 72
- strings
 - format string attacks, 114
 - Sulley primitives, 125
- structures, defining, 9
- Sulley, 123–136
 - installing, 124
 - primitives, 125–128
 - WarFTPD, 129–136
- switch statement, 147

T

- testing
 - file fuzzers, 121
 - IDAPython installation, 154
- thread enumeration, 34
- threads, remote thread creation, 98–110
- thresholds, stack variables, 162

U

- unions, defining, 9
- UnloadDLLHook, 72
- unpacking, UPX, 177–181
- UPX Packer, IdaPyEmu, 176–181
- user mode, black-box debuggers, 14
- utility functions, IDAPython, 155

V

- verbs, 128
- verifying. *See* testing
- VirtualProtectEx() function, 53
- VMware, appliances, 2

W

- WarFTPD, Sulley, 129–136
- white-box debuggers, vs. black-box debuggers, 13
- Windows
 - debuggers, 25–55
 - associating processes, 25–33
 - breakpoints, 43–55
 - CPU register state, 33–38
 - debug event handlers, 39–42
 - fuzzing drivers in, 137–151
 - GFlags, 113
 - installing Python in, 2
 - using Python in, 2
- WinPcap library, 124
- WriteProcessMemory() function, 43

X

- x86 assembly, ESI and EDI registers, 15