



Index

- Access
 - file, 22
 - hardware. *See* Hardware rings for, 50–52
- Acrostic messages, 244
- Active offenses, 18
- Active process contexts, 60
- Address resolution protocol (ARP), 286–288
- Addresses
 - creating, 248–252
 - in detour patching, 121–125
 - endpoint associations with, 254–256
 - hardware, 217–218
 - kernel module, 299–301
 - MAC, 285–287
 - in paging, 55–56
 - for processes, 106–110
 - structures for, 246–248
 - virtual, 56, 58, 108
- AdjustTokenGroups function, 194
- AdjustTokenPrivileges function, 194
- Advisories, security, 14
- Alignment, instruction, 115
- AllCPURaised function, 190
- AMD processors, microcode updates, 236
- AppInit_DLLs key, 77
- Area-of-effect restrictions, 12
- ARP *See* Address Resolution Protocol
- ASCII payloads, steganography on, 244–245
- Attacker motives, 2–3
- Attacks, bounce, 266
- Authentication functions, patching, 133
- Authentication IDs (AUTH_IDS), 208–210
- Automated code-scanning tools, 16
- Back doors, 2
 - drawbacks of, 240–241
 - to op-codes, 236
 - software for, 215
- Back Orifice program, 2, 240
- \$(BASELIB) variable, 29
- Behavior detection, 308–312
- BHOs (Browser Helper Objects), 297
- Binary code, patching, 8
- bind function, 263
- Binding
 - in IAT hooking, 74
 - in inline function hooking, 75
 - to interfaces, 263–264
- BIOS, accessing, 221
- BLINK member
 - changing value of, 183–185, 187, 189
 - as process pointer, 180
- Bootloaders, modifying, 47
- Bootstrap code, activation of, 216
- Bouncing packets, 266–267
- Browser Helper Objects (BHOs), 297
- Buffer-overflow exploits, 11, 15–16
- Buffer pools, 278
- Buffers in NDIS, 274, 278
- Bug fixes by Microsoft, 13–14
- Build environments, 27
- Build utility, 29
- Bus
 - data, 217–218
 - I/O, 219–221
 - PCI, 221
- Bypassing
 - firewalls, 18
 - forensic tools, 18–19
- Call gates, 62
- Callbacks, protocol driver, 273–278
- Calls
 - DPCs, 190–191
 - far, 62
 - in rootkit detection, 299
- Cavern infection, 133
- Chains, driver, 138
- Channels, covert. *See* Covert channels
- Checked build environments, 27
- CheckFunctionBytesNt
 - DeviceIoControlFile function, 117–118
- CheckFunctionBytesSe
 - AccessCheck function, 118
- CheckNtoskrnlForOutsideJump function, 304–306
- CIH virus, 221
- Cleanup routines, 26
- cli instruction, 51
- Code
 - introducing into kernel, 25–26
 - patching. *See* Patching

- Code-byte patching method, 114
- Code-scanning tools, automated, 16
- Code segment (CS) registers, 62
- COMMAND BYTE for keyboard ports, 224
- Compiler libraries, 25
- Completion routines for IRPs, 102–106
- Connecting to remote servers, 257–259
- CONNECTION_CONTEXT pointer, 253
- CONNINFO101 structure, 103
- CONNINFO102 structure, 103
- CONNINFO110 structure, 103–104
- CONTAINING_RECORD macro, 150
- Contexts
 - active process, 60
 - for endpoints, 252–254
- Control flow, rerouting, 115–117
- Control Register Zero (CR0), 59, 66–67
- Control registers, 66–68
- Controllers, keyboard. *See* Keyboard controller access
- ConvertScanCodeToKeyCode function, 150
- Covert channels, 239–240
 - disguised TCP/IP protocols, 241–245
 - host emulation. *See* Host emulation
 - NDIS in. *See* NDIS interface
 - raw network manipulation, 262–267
 - remote command, control, and exfiltration of data, 240–241
 - TDI in. *See* TDI (Transport Data Interface) specification
- CPLs *See* Current Privilege Levels
- CPUs
 - interrupts for, 69
 - for ring enforcement, 50
 - tables for, 52
- CR0 (Control Register Zero), 59, 66–67
- CR1 register, 67
- CR2 register, 67
- CR3 register, 56, 60, 67
- CR4 register, 67
- CreateRemoteThread function, 79–80
- CS (code segment) registers, 62
- CSDVersion key, 174
- CSRSS.EXE file, 310
- ctrl2cap driver, 140
- Current privilege levels (CPLs), 54
- CurrentBuildNumber key, 174
- CurrentVersion key, 174
- Cyberwarfare, 6
- Data bus, 217–218
- DATA BYTE for keyboard ports, 224
- DbgPrint statement, 223–225
- DDK (Driver Development Kit), 27–28
- \$(DDK_LIB_PATH) variable, 29
- Debug statements, logging, 31–32
- Debug View tool, 31–32
- Decompressing .sys files, 43–46
- Deferred Procedure Calls (DPCs), 190–191
- Descriptor checks, 53–54
- Descriptor privilege levels (DPLs), 53–54
- DetermineOSVersion function, 172–173
- Detour patching, 76, 114–115
 - function byte checking in, 117–118
 - NonPagedPool memory for, 121–122
 - overwritten instruction tracking, 118–120
 - rerouting control flow, 115–117
 - runtime address fixups in, 121–125
- DetourFunctionNtDeviceIoControlFile function, 123
- DetourFunctionSeAccessCheck function, 121
- Device drivers. *See* Drivers
- DEVICE_EXTENSION structure, 141–142
- Device IRQLs (DIRQLs), 189–190
- DeviceIoControl function, 176–177, 201
- DeviceTree utility, 137, 142
- DPCs *See* Deferred Procedure Calls
- Direct code-byte patching method, 114
- Direct Kernel Object Manipulation (DKOM), 169
 - benefits and drawbacks, 169–171
 - device driver communications, 175–179
 - hiding with, 179–193
 - device drivers, 185–189
 - processes, 180–185
 - synchronization issues, 189–193
 - operating system version determination, 171–175
 - process token privilege and group elevation with, 193–194
 - adding SIDs to tokens, 204–208
 - finding tokens, 194–195
 - log events in, 208–210
 - modifying tokens, 195–204
- DIRQLs (Device IRQLs), 189–190
- Disguised TCP/IP protocols, 241–242
 - ASCII payloads in, 244–245
 - DNS requests in, 243–244
 - encryption in, 242–243
 - timing in, 243
 - traffic patterns in, 242
- DISPATCH_LEVEL, 189–190
- DispatchPassDown function, 141
- DispatchRead function, 141, 146
- DKOM. *See* Direct Kernel Object Manipulation (DKOM)
- DLLs
 - forwarding, 74
 - injecting into processes, 76–81
 - listing, 306–307
- DNS (Domain Name Service), 241–244

- DPLs *See* Descriptor Privilege Levels
- DrainOutputBuffer function, 225–226, 232
- Driver Development Kit (DDK), 27–28
- DRIVER_OBJECT structure, 186–187
- Driver tables for IRPs, 98–99
- DRIVER type, 28
- DriverEntry function
 - detour patches, 124–125
 - device driver communication, 177–179
 - file filter drivers, 153–154
 - file handles, 37
 - I/O request packets, 34
 - IDTs, 64–65
 - jump templates, 128–130
 - keyboard LEDs, 228–229
 - keystroke monitors, 234–235
 - kernel hooks, 95
 - processes, 182–183
 - protocol registering, 268
 - runtime patching, 124–125
 - scancode mapping, 141, 144
 - SSDT hooks, 302–303
 - symbolic links, 38
 - threads in, 259, 261
 - Windows device drivers, 26, 30
- Drivers
 - communicating with, 175–179
 - hiding, 185–189
 - for introducing code into kernel, 25–26
 - layered, 135–136
 - file filter, 153–167
 - keyboard sniffers, 136–140
 - KLOG rootkit for, 140–153
 - loading, 40–43
 - for network operations, 24
 - Windows. *See* Windows device drivers
- DriverUnload function, 146
- EAs *See* Extended Attributes
- Easter Eggs, 9
- Eavesdropping, 5
- EFlags register, 67–68
- 8259 keyboard controller, 222
- Elevation, group, 193–194
- adding SIDs to tokens, 204–208
- finding tokens, 194–195
- log events in, 208–210
- modifying tokens, 195–204
- Embedded systems, 214
- Emulation, host, 285–286
 - ARP in, 286–288
 - IP gateways in, 289
 - MAC addresses in, 286
 - packet transmissions in, 289–293
- Encase tool, 19
- Encryption, 19
 - for covert channels, 242–243
 - with steganography, 244
- Endpoints
 - for address objects, 248
 - contexts for, 252–254
 - local address associations with, 254–256
- Entercept program, 17–18
- EnumProcesses function, 307
- EnumProcessModules function, 306–307
- EPROCESS structure
 - listing members of, 170, 189
 - locating, 202
 - in process hiding, 180–185
 - for tokens, 194
- ether_arp structure, 287
- ether_header structure, 287
- Ethernet addresses, switches for, 286
- ETHREAD structure, 180, 185
- Event Log, 208
- Event Viewer, faking out, 208–210
- EX_FAST_REF structure, 194
- Executable code, patching, 8
- Execution, tracing, 308
- Exfiltration of data, 2, 240–241
- Exploits
 - vs. rootkits, 11
 - software, 13–16
 - zero-day, 12
- Extended attributes (EAs), 247
- Far calls, 62
- Far jumps, 62, 118
- Far returns, 62
- Fast call methods, 94
- FastIo function, 165–166
- FASTIOPRESENT macro, 166
- File filter drivers, 153–167
- FILE_FULL_EA_INFORMATION structure, 247–249, 252–253
- File-hiding code, 24
- Files
 - hidden, detecting, 308–309
 - kernel access to, 22
 - for Windows device drivers, 27–29
- FilterFastIoQueryStandardInfo function, 166–167
- FindFirstFile function, 72–73
- Finding
 - hooks
 - address ranges in, 299–301
 - IAT, 305–307
 - inline, 303–305
 - IRP handler, 305
 - SSDT, 302–303
 - tokens, 194–195
- FindNextFile function, 72–73
- FindProcessEPROC function, 183, 194, 202
- FindProcessToken function, 195
- FindPsLoadedModuleList function, 187
- FindResource function, 44
- Firewalls
 - bypassing, 18
 - source port control by, 266
- Firmware, 19, 215–217
- Flashable BIOS chips, 221
- FLINK member
 - changing value of, 183–185, 187, 189
 - offsets to, 181
 - as process pointer, 180
- Forensic tools, bypassing, 18–19
- Forging source ports, 266
- Free build environments, 27
- Function bytes, checking for, 117–118
- Fusion rootkits, 32–33
 - file handles for, 37–38
 - I/O Request Packets with, 33–36
 - symbolic links for, 38–39
- GainExclusivity function, 191–193
- Gates
 - cell, 62
 - interrupt, 64
 - task and trap, 65–66

- GDTs *See* Global Descriptor Tables
- GetDrivesToHook function, 154
- GetListOfModules function, 300–301
- GetLocationOfProcessName function, 182–183
- GetModuleFileNameEx function, 306–307
- GetModuleInformation function, 306–307
- GetProcAddress function, 74, 107–108
- GetVersionEx function, 172–173
- Global Descriptor Tables (GDTs), 52
 - dump of, 54–55
 - tricks using, 61
- GORINGZERO instruction, 236
- Group elevation with DKOM, 193–194
 - adding SIDs to tokens, 204–208
 - finding tokens, 194–195
 - log events in, 208–210
 - modifying tokens, 195–204
- Guarding-the-doors approach, 296–298
- HANDLE_TABLE structure, 311
- Hard reboots from keyboard controllers, 229
- Hardware, 49–50
 - access, 217–221
 - addresses in, 217–218
 - BIOS, 221
 - I/O bus, 219–220
 - keyboard controller. *See* Keyboard controller access
 - latching in, 218
 - PCI and PCMCIA devices, 221
 - timing in, 218–219
 - control registers, 66–68
 - firmware modifications, 216–217
- Interrupt Descriptor Tables, 62–66
 - manipulating, 213–215
 - microcode updates, 236–237
- memory descriptor tables, 61–62
- memory pages. *See* Memory pages
- multiprocessor systems, 68–69
- Ring Zero, 50–52
- system service descriptor tables, 66
- tables for, 52
- Hardware reordering of instructions, 69
- Hashing, 19
- Hidden items, detecting files, 308–309
- processes, 309–312
- Hiding
 - with DKOM, 179–180
 - device drivers, 185–189
 - processes, 180–185
 - synchronization issues, 189–193
 - processes, 24, 87–91, 180–185
- HIPS technology, 17–18
- Hlade’s Law, 135
- HOOK_SYSCALL macro, 86
- HookDrive function, 157
- HookDriveSet function, 154, 156–157
- HookedDeviceControl function, 99–102
- HookImportsOfImage function, 107–110
- HookInterrupts function, 93
- HookKeyboard function, 141–143
- Hooks, 71–73
 - finding, 298–299
 - address ranges in, 299–301
 - IAT, 305–307
 - inline, 303–305
 - IRP handler, 305
 - SSDT, 302–303
 - hybrid approach, 106–112
 - IAT, 72–74, 106–107
 - injecting DLLs into processes, 76–81
 - kernel, 81–82
 - IDTs, 91–95
 - IRPs, 96–106
 - SSDTs, 82–91
 - memory space for, 111–112
 - looking for, 298–308
- Host emulation, 285–286
- ARP in, 286–288
- IP gateways in, 289
- MAC addresses in, 286
- packet transmissions in, 289–293
- HTONL macro, 288
- HTONS macro, 288
- Hybrid hooking approach, 106–112
- HybridHook example, 106–110
- Hyper-threaded systems, 68–69
- IAT (Import Address Table)
 - finding hooks, 305–307
 - hooking, 72–74, 106–107
 - in rootkit detection, 299
- ICH (I/O Controller Hub) chips, 220–221
- ICMP packets, 245
- IdentifySSDTHooks function, 303
- Idle process, 310
- IDS software, bypassing, 18
- IDTENTRY structure, 92
- IDTINFO structure, 92
- IDTRs (interrupt descriptor table registers), 62–63
- IDTs (Interrupt Descriptor Tables), 52
 - hooking, 91–95
 - in rootkit detection, 299
 - working with, 62–66
- IMAGE_DIRECTORY_ENTRY_IMPORT structure, 108
- IMAGE_IMPORT_BY_NAME structure, 73
- IMAGE_IMPORT_DESCRIPTOR structure, 73, 108
- IMAGE_INFO structure, 107
- Import Address Table (IAT)
 - finding hooks, 305–307
 - hooking, 72–74, 106–107
 - in rootkit detection, 299
- in_addr structure, 263
- in instruction, 51, 217
- Include files, 25
- INCLUDES variable, 28
- INETADDR macro, 288
- Infected files for reboot survival, 46
- .ini files for surviving reboot, 46
- InitThreadKeyLogger function, 143–144
- Injecting DLLs into processes, 76–81

- Inline functions
 - finding hooks, 303–305
 - hooking, 74–76
- InstallTCPDriverHook
 - function, 98–99
- InstDrv tool, 30–31
- Instructions, alignment, 115
- INT 2E instruction, 83
- Integrity Protection Driver (IPD), 17, 296–297
- Intel processors, microcode updates, 236
- Interfaces, binding to, 263–264
- Interlocked functions, 190
- InterlockedExchange function, 98
- Interrupt descriptor table registers (IDTRs), 62–63
- Interrupt Descriptor Tables (IDTs), 52
 - hooking, 91–95
 - in rootkit detection, 299
 - working with, 62–66
- Interrupt flags, 68
- Interrupt gates, 64
- Interrupt service routines (ISRs), 63, 125–126
- Interrupt tables
 - for CPUs, 69
 - with jump templates, 126–132
- Interrupts for keystrokes, 229–235
- I/O bus, 219–221
- I/O Control Codes (IOCTLs), 33–34, 38, 175–176
- I/O Controller Hub (ICH) chips, 220–221
- I/O Request Packets. *See* IRPs (I/O Request Packets)
- IO_STACK_LOCATION, 138–139
- IoAttachDevice function, 143
- IoCallDriver function, 138–139, 146–147, 255
- IoCompletionRoutine function, 100, 102–106
- IoCopyCurrentIrpStackLocationToNext function, 146
- IoCreateDevice function, 141
- IoCreateSymbolicLink function, 38
- IOCTL_DRV_INIT IOCTL, 176
- IOCTL_DRV_VER IOCTL, 176
- IOCTLs (I/O Control Codes), 33–34, 38, 175–176
- IoDetachDevice function, 151
- IoGetCurrentIrpStackLocation function, 146
- IoGetCurrentProcess function, 180
- IoGetDeviceObjectPointer function, 98
- IoGetNextIrpStackLocation function, 146
- IoSetCompletionRoutine function, 147, 161
- IoSkipCurrentIrpStackLocation function, 140
- IoSkipCurrentStackLocation function, 139–140
- IPD (Integrity Protection Driver), 17, 296–297
- IRP_ values, 35
- IRP_MJ_DEVICE_CONTROL, 177–178
- IRPs (I/O Request Packets), 33–36
 - completion routines for, 102–106
 - driver tables for, 98–99
 - finding hooks, 305
 - hooking, 96–106
 - for keyboards, 136–137
 - in rootkit detection, 299
 - and stack locations, 137–140
 - working with, 33–36
- ISRs. *See* Interrupt Service Routines
- jmp instructions, 299
- Jump templates, 125–132
- Jumps, far, 62, 118
- KeCurrentProcessorNumber function, 190
- KeGetCurrentIrql function, 190
- KeInitializeDpc function, 191
- KeInitializeEvent function, 257, 259
- KeInsertQueueDpc function, 191
- KeNumberProcesses function, 190
- KeRaiseIrql function, 190
- KeReleaseSemaphore function, 152
- kernel, 21–22
 - components of, 22–23
 - decompressing .sys files, 43–46
 - fusion rootkits, 32–33
 - file handles for, 37–38
 - IRPs with, 33–36
 - symbolic links for, 38–39
 - hooking, 81–82
 - IDTs, 91–95
 - IRPs, 96–106
 - SSDTs, 82–91
 - introducing code into, 25–26
 - loading rootkits into, 39–43
 - logging debug statements, 31–32
 - NDIS TCP/IP support. *See* NDIS interface
 - rootkit design for, 23–25
 - surviving reboots, 46–47
 - TDI TCP/IP support. *See* TDI (Transport Data Interface) specification
 - Windows device drivers for, 26–30
- Kernel.dll file, 72
- kernel mode, 54
 - for networking code, 246–261
 - self-determination, 173–174
- Kernel modules, address ranges of, 299–301
- Kernel32.dll file, 306
- Kernel's Processor Control Blocks (KPRCBs), 180–181, 186
- KeServiceDescriptorTable tables, 82–83, 85
- KeSetTargetProcessorDPC function, 191
- KeSetTimerEx command, 228
- KeStallExecutionProcessor function, 219, 225
- KeWaitForSingleObject function, 149–150, 152–153
- Keyboard controller access, 222–235
 - controller addressing, 222
 - for keystroke monitoring, 229–235
 - for LED indicators, 222–229
 - for hard reboots, 229
- KEYBOARD_INPUT_DATA structures, 147–148

- Keyboard sniffers, 136–137
 - IRPs for, 137–140
 - KLOG, 140–153
- Keypress events, 148
- Keyrelease events, 148
- KiSystemService dispatcher, 82, 93
- KLOG rootkit, 140–153
- KPRCBs *See* Kernel's Processor Control Blocks
- KPROCESS structure, 61
- KTHREAD structure, 180
- KUSER_SHARED_DATA
 - memory area, 111
- Languages, type-safe, 15–16
- Latching, 218
- Late-demand binding, 74
- Layered drivers, 135–136
 - file filter, 153–167
 - keyboard sniffers, 136–140
 - KLOG rootkit for, 140–153
- LDTs *See* Local Descriptor Tables
- LED keyboard indicators, 222–229
- LGDT instruction, 61
- Libraries
 - compiler, 25
 - linking, 29
- LIDS *See* Linux Intrusion Detection System
- LIDT *See* Load Interrupt Descriptor Table instruction
- Linkage key, 268
- Linkages, 268–269
- Linking libraries, 29
- Links, symbolic
 - for fusion rootkits, 38–39
 - in rootkit detection, 298
- Linux, 9
- Linux Intrusion Detection System (LIDS), 17
- LIST_ENTRY structure, 180, 186, 311
- ListProcessesByHandleTable
 - function, 311–312
- Load Interrupt Descriptor Table (LIDT)
 - instruction, 63
- Loading
 - drivers, 30–31, 40–43
 - rootkits, 39–43
- LoadLibrary function, 74, 80, 111
- LoadResource function, 44–45
- Local addresses
 - creating, 248–252
 - endpoint associations with, 254–256
- Local Descriptor Tables (LDTs)
 - purpose of, 52
 - table-indicator bit in, 62
- Locally Unique Identifiers (LUIDs), 200–204
- Logging
 - debug statements, 31–32
 - processes, 208–210
- Loki tool, 245
- Look-ahead buffers in NDIS, 274, 278
- LookupPrivilegeValue
 - function, 200
- Lookups, page-table, 56–58
- LUID_AND_ATTRIBUTES
 - structure, 200–204
- LUIDs *See* Locally Unique Identifiers
- MAC addresses, 285–287
- MAC interface, 267
- Machine status word, 66
- Major function pointers, 34–36
- MAKEFILE file, 29
- MAKELONG macro, 92, 230–231
- Malicious code, 9
- Malicious modifications, 6
- MappedSystemCallTable
 - function, 85
- Mapping scancodes, 140–141
- MDL *See* Memory Descriptor Lists
- Memory Descriptor Lists (MDLs), 84–85
- Memory descriptor tables, 61–62
- Memory management
 - access restrictions, 50–52, 69
 - by kernel, 22
 - for SSDTs, 84–86
- Memory pages, 53
 - address translation for, 55–56
 - checks for, 53–55, 59–60
 - multiple processes, 59–60
 - page directories
 - checks, 53–54
 - entries, 56, 58
 - multiple, 59–60
- page tables
 - directories, 56
 - entries, 58–59
 - lookups, 56–58
 - processes and threads in, 60–61
 - read-only access to, 59
- Memory space for hooking, 111–112
- METHOD_BUFFERED
 - function, 176
- METHOD_NEITHER
 - function, 100
- Microcode updates, 236–237
- Microsoft, bug fixes by, 13–14
- Migbot rootkit
 - loading drivers with, 40–41
 - rerouting control flow using, 115–117
- Minimal footprints, 241
- Model-Specific Registers (MSRs), 94–95
- Modifications
 - firmware, 216–217
 - software, 6, 10
 - source-code, 9–10
 - tokens, 195–196
- MODULE_ENTRY structure, 186, 189
- MODULE_INFO structure, 300–301
- MODULE_INFORMATION
 - structure, 307
- MODULEINFO structure, 307
- Monitors, keystroke, 229–235
- Morris Worm, 7
- Motives of attackers, 2–3
- Moving whole packets, 278–285
- MSRs (Model-Specific Registers), 94–95
- Multiple page directories and processes in memory
 - pages, 59–60
- Multiprocessor systems, 68–69
- my_function_detour_
 - ntdeviceiocontrolfile
 - function, 120, 123–124
 - my_function_detour_
 - seaccesscheck function, 119–120
- MyImageLoadNotify function, 107–108
- MyKiFastCallEntry function, 95
- MyPassThru function, 139

- Naked functions, 118
- NDIS_BUFFER structure, 289, 292
- NDIS_PACKET structure, 279, 289–292
- NDIS interface, 267
 - for host emulation, 285–293
 - packet moving in, 278–285
 - protocol driver callbacks in, 273–278
 - protocol registration in, 267–273
- NdisAllocateBuffer function, 290
- NdisAllocateBufferPool function, 278
- NdisAllocateMemory function, 290
- NdisAllocatePacketPool function, 278
- NdisOpenAdapter function, 271
- NdisQueryBuffer function, 292
- NdisRegisterProtocol function, 270–271
- NdisRequest function, 273
- NdisSend function, 289–291
- NdisTransferData function, 280
- NdisTransportData function, 278
- NdisUnchainBufferAtFront function, 292
- NetBus program, 2
- Network-based IDS (NIDS), 17–18
- NetworkCards key, 268
- Networks
 - drivers for, 24
 - raw manipulation, 262–267
- NewZWQuerySystem
 - Information function, 89–91
- NIDS (network-based IDS), 17–18
- NonPagedPool memory, 121–122
- NOP instructions, 116
- NtDeviceIoControlFile function, 115
- Ntdll.dll file, 72, 92, 306
- NtLoadDriver function, 296
- NtOpenSection function, 296
- NTOSKRNL structure, 302
- NtQuerySystemInformation function, 87
- NumberOfRaisedCPU function, 190–191
- OBJ_KERNEL_HANDLE, 248
- Object attributes structure, 248–249
- Objects in DKOM, 169
- Offensive technologies, 17–19
- Okena Storm Watch program, 17–18
- OldIrpMjDeviceControl function, 101
- OnBindAdapter function, 276
- OnCloseAdapterDone function, 273–274
- OnOpenAdapterDone function, 271–274, 278–279
- OnPNPEvent function, 276–277
- OnProtocolUnload function, 277
- OnReadCompletion function, 147
- OnReceiveDoneStub function, 275
- OnReceivePacket function, 277
- OnReceiveStub function, 273–275, 278, 280–283
- OnRequestDone function, 276
- OnResetDone function, 276
- OnSendDone function, 274, 290–291
- OnSniffedPacket function, 283–285
- OnStatus function, 275
- OnStatusDone function, 276
- OnStubDispatch function, 33–34
- OnTransferDataDone function, 274, 280, 283–284
- OnUnbindAdapter function, 276
- OnUnload function
 - for IRPs, 34
 - for jump templates, 131–132
 - for keystroke monitors, 232
 - for protocol callbacks, 277–278
- OpenProcess function, 79
- OpenProcessToken function, 194
- Operating systems
 - kernel. *See* kernel version determination, 171–175
- OSVERSIONINFO structure, 171–172
- OSVERSIONINFOEX structure, 171–172
- OSVERSIONINFOEXW structure, 174
- OSVERSIONINFOW structure, 174
- OurFilterDispatch function, 161
- OurFilterHookDone function, 162
- out instruction, 51, 217
- Overwrite process in code patching, 116–117
- Overwritten instructions, tracking, 118–120
- Packet pools, 278
- Packets
 - bouncing, 266–267
 - moving, 278–285
 - sending
 - in host emulation, 289–293
 - with raw sockets, 265–266
- Page Directory table, 52
- Page frames, 57
- Pageable drivers, 40
- Paged in memory, 55
- Paged out memory, 55
- Pages, memory, 53
 - address translation for, 55–56
 - checks for, 53–55, 59–60
 - multiple processes, 59–60
 - page directories
 - checks, 53–54
 - entries, 56, 58
 - multiple, 59–60
 - page tables
 - directories, 56
 - entries, 58–59
 - lookups, 56–58
 - processes and threads in, 60–61
 - read-only access to, 59
- Patching
 - description, 8
 - runtime, 113–114
 - detour. *See* Detour patching

- Patching (*cont.*)
 - jump templates, 125–132
 - variations, 133
- PCI and PCMCIA device
 - access, 221
- PE *See* Portable Executable
- PEBs *See* Process Environment Blocks
- Pending status in NDIS, 271
- Peripheral buses, 219
- PIC *See* Programmable Interrupt Controller
- PIDs *See* Process Identifiers
- Portable Executable (PE)
 - format, 108
- Ports
 - forging sources, 266
 - for keyboard controller, 222
 - reading and writing, 217
- Preambles, 75–76
- Prefix method, 162
- Print_keystroke function, 233
- Privileges for tokens, 196–204
- Process Environment Blocks (PEBs), 307
- Process Explorer, 198
- Process Identifiers (PIDs)
 - in hybrid hooking, 107
 - in process detection, 180–182, 311
 - for remote threads, 79
- Process tokens
 - finding, 194–195
 - log events in, 208–210
 - modifying, 195–204
 - SIDs for, 204–208
- Processes
 - address space for, 106–110
 - hidden, detecting, 309–312
 - hiding, 24, 87–91, 180–185
 - injecting DLLs into, 76–81
 - kernel management by, 22
 - listing, sources of, 310–312
 - logging, 208–210
 - in memory pages, 59–61
 - scheduling, 185
 - vs. tasks, 66
- Processors
 - in embedded systems, 214
 - IDTs for, 91
- Programmable Interrupt Controller (PIC), 229
- Promiscuous sniffing, 264–265
- Protocol driver callbacks, 273–278
- ProtocolCharacteristics
 - structure, 269
- Protocols
 - disguised. *See* Disguised TCP/IP protocols
 - registering, 267–272
- PsCreateSystemThread
 - function, 144
- PsGetCurrentProcess function, 94, 180, 186
- PsGetVersion function, 173
- PsLoadedModuleResource
 - function, 189
- PspActiveProcessMutex
 - function, 189
- PspExitProcess function, 184–185
- PsSetImageLoadNotifyRoutine
 - function, 106–107, 112
- RaiseCPUIrqlAndWait
 - function, 191–192
- Raw network manipulation, 262
 - binding to interfaces, 263–264
 - bouncing packets, 266–267
 - forging sources, 266
 - sending packets, 265–266
 - sniffing, 264–265
 - on Windows XP, 262–263
- Read-only table access, 59
- ReadFile function, 35, 38
- Reading ports, 217
- Reboots
 - from keyboard controllers, 229
 - surviving, 46–47
- recvfrom function, 264
- Registering
 - protocols, 267–272
 - for surviving reboot, 46–47
- Registers
 - control, 66–68
 - latching between, 218
- Registry
 - for injecting DLLs into processes, 77
 - key detection, 308–309
 - operating system version queries in, 174–175
- RegOpenKeyEx function, 309
- RegQueryValue function, 174
- RegQueryValueEx function, 174, 309
- Relative Virtual Addresses (RVAs), 108
- Remote command and control, 5, 240–241
- Remote servers
 - connecting to, 257–259
 - sending data to, 259–261
- Remote shells, 240
- Remote threads, 79–81
- Reordering of instructions, 69
- REQINFO structure, 104
- Rerouting control flow, 115–117
- ResponseToArp function, 287
- Restarting rootkits, 24–25
- Returns, far, 62
- Ring Zero, 50–52
- Rings, 50–52, 54–55
- RootkitDispatch function, 177–179
- RootkitRevealer tool, 308–309
- Rootkits
 - characteristics of, 4
 - detecting, 295–296
 - behavior detection, 308–312
 - guarding-the-doors approach, 296–298
 - looking for hooks, 298–308
 - scanning rooms, 298
 - vs. exploits, 11
 - history of, 7–8
 - for kernel, 23–25
 - legitimate uses of, 6–7
 - loading, 39–43
 - offensive technologies, 17–19
 - operation of, 8–10
 - purpose of, 4–5
 - restarting, 24–25
 - and software exploits, 13–16
 - vs. viruses, 11–13
- RtlCopyMemory function, 206–207
- RtlGetVersion function, 174
- Run key, 46
- Runtime address fixups, 121–125
- Runtime patching, 113–114
 - detour. *See* Detour patching
 - jump templates, 125–132
 - variations, 133
- RVAs *See* Relative Virtual Addresses
- Scancodes
 - in IRPs, 136–137
 - mapping, 140–141
- Scanning rooms, 298

- Scheduling processes, 185
- SCM. *See* Service Control Manager
- SeAccessCheck function, 115
- Segment checks, 53–54
- Segment descriptors, 53
- Sending
 - data to remote servers, 259–261
 - packets
 - in host emulation, 289–293
 - with raw sockets, 265–266
 - TCP handshakes, 257–259
- SendKeyboardCommand function, 226–227, 232
- SendRaw function, 288–289
- sendto function, 265–266
- Service Control Manager (SCM), 41–42, 183, 187
- ServiceDescriptorEntry structure, 85, 303
- Services key, 297
- SetLEDs function, 227
- SetPriv function, 199–202
- SetWindowsHookEx function, 78
- SGDT instruction, 61
- Siberian gas pipeline explosion, 6
- SID_AND_ATTRIBUTES structure, 204–207
- SIDs for tokens, 195–196, 204–208
- SIDT *See* Store Interrupt Descriptor Table
- Signatures, scanning for, 298
- SizeOfResource function, 45
- SMP *See* Symmetric Multi-Processing
- SMSS.EXE file, 310
- Sniffers, keyboard, 136–140
- Sniffing with raw sockets, 264–265
- Socket function, 263
- Sockets on Windows XP, 262–263
- Software eavesdropping, 5
- Software exploits, 11–16
- Software modifications, 6, 10
- Source-code modifications, 9–10
- Source port forging, 266
- SOURCES file, 27–29
- Spinlocks, 145
- Spyware modifications, 9
- SSDTs (System Service Dispatch Tables), 82–84
 - finding hooks, 302–303
 - hooking, 86–91
 - in rootkit detection, 299
 - memory protection for, 84–86
 - purpose of, 52, 66
- SSPTs (System Service Parameter Tables), 82
- Stack and IRPs, 137–140, 146
- STATUS BYTE for keyboard ports, 224
- Stealth, role of, 2–3
- Steganography, 19
 - on ASCII payloads, 244–245
 - for covert channels, 239, 243–245
- sti instruction, 51
- Store Interrupt Descriptor Table (SIDT) instruction, 63–65, 92
- Storm Watch program, 17–18
- Surviving reboots, 46–47
- SwapContext function, 309–310
- Switches for ARP, 286
- Symbolic links
 - for fusion rootkits, 38–39
 - in rootkit detection, 298
- Symmetric Multi-Processing (SMP) systems, 68–69
- SYN packets, 241, 266–267
- SYN-ACK packets, 266–267
- Synchronization issues, 189–193
- .sys files, decompressing, 43–46
- SYSCALL_INDEX macro, 86
- SYSENTER instruction
 - IDT hooks with, 94–95
 - for system calls, 66
 - for system service dispatcher, 83
- SYSTEM_LOAD_AND_CALL IMAGE method, 40–41
- System process, 310
- SYSTEM_PROCESSES structure, 88
- System Service Descriptor Tables (SSDTs)
 - purpose of, 52, 66
 - in rootkit detection, 299
- System Service Dispatch Tables (SSDTs), 82–84
 - finding hooks, 302–303
 - hooking, 86–91
 - memory protection for, 84–86
- System Service Parameter Tables (SSPTs), 82
- SYSTEM_THREADS structure, 88
- SystemModuleInformation function, 300
- SYSTEMSERVICE macro, 86
- TA_TRANSPORT_ADDRESS structure, 250
- Table-indicator bit, 62
- Tables
 - driver, 98–99
 - for hardware, 52
 - read-only access to, 59
- TARGETLIBS variable, 29
- TARGETNAME variable, 28
- TARGETPATH variable, 28
- TARGETTYPE variable, 28
- Task gates, 65–66
- Task Switch Segments (TSSs), 66
- TCP handshakes, 257–259
- TCP/IP protocols, disguised, 241–242
 - ASCII payloads in, 244–245
 - DNS requests in, 243–244
 - encryption in, 242–243
 - timing in, 243
 - traffic patterns in, 242
- TCPIP.SYS driver, 97–98
- TDI (Transport Data Interface) specification
 - for kernel mode networking code, 246
 - address structures for, 246–247
 - endpoint/address associations in, 254–256
 - endpoints with context in, 252–254
 - local address objects for, 248–252
 - remote server connections in, 257–259
 - remote server data transmissions in, 259–261
- vs. NDIS, 293

- TDI_ADDRESS_IP structure, 250
- TDI_IP_ADDRESS structure, 250
- TDIEntityID structure, 99–100
- TDIObjectID structure, 99–100
- Templates, jump, 125–132
- ThreadKeyLogger function, 144, 149
- Threads
 - in memory pages, 60–61
 - remote, 79–81
- Time factors
 - in communication, 243
 - in hardware access, 219
- TimerDPC function, 223, 227–228
- Timers
 - in keyboard sniffers, 152
 - land-mine, 12
- Tokens
 - finding, 194–195
 - modifying, 195–204
 - SIDs for, 195–196, 204–208
 - log events in, 208–210
- Tombstones, 31
- Tracing execution, 308
- Tracking overwritten instructions, 118–120
- Traffic patterns for covert channels, 242
- Trampolines, 76
- Translation, address, 55–56
- Transport Data Interface specification. *See* TDI (Transport Data Interface) specification
- Trap flags, 67–68
- Trap gates, 65–66
- Tripwire tool, 8, 19
- Trojan files, 46
- TSSs (Task Switch Segments), 66
- Type-safe languages, 15–16
- UDP packets, spoofing, 243
- UNHOOK_SYSCALL macro, 86
- UnhookDrive function, 157
- UNIX operating systems, 7–8
- Unload function
 - for keyboard sniffers, 151
 - for Windows device drivers, 30–31
- Updates, microcode, 236–237
- User mode
 - determining the OS version from, 171–173
 - file handles for, 37, 54
 - fusion rootkits for, 32–33
 - Ring Three programs, 50, 54
- User/supervisor bit, 54
- Userland
 - device driver communication from, 175–179
 - hooks, 71–73
 - IAT, 73–74
 - inline functions, 74–76
 - injecting DLLs into processes, 76–81
 - _util_decompress_sysfile function, 44
 - _util_load_sysfile function, 42–43
- Virtual addresses, 56, 58, 108
- VirtualAllocEx function, 80
- Viruses, 7
 - problems with, 12–13
 - vs. rootkits, 11–12
- WaitForKeyboard function, 224–226, 231–232
- WatchGuard ServerLock software, 17
- Whole packet moving, 278–285
- Windows device drivers, 26–27
 - build environments for, 27
 - build utility for, 29
 - DDK for, 27
 - files for, 27–29
 - loading and unloading, 30–31
- Windows Device Manager, 185
- Windows Event Viewer, faking out, 208–210
- Windows hooks, 77–79
- Windows XP, raw sockets on, 262–263
- Write protect (WP) bits, 66–67
- WriteFile function, 35, 38
- WriteProcessMemory function, 80, 106
- Writing to ports, 217
- WSAIoctl function, 265
- WSAStartup function, 262–263
- Zero-day exploits, 12
- ZwCreateFile function, 247–248, 251–252, 297
- ZwCreateKey function, 297
- ZwCreateLinkObject function, 297
- ZwOpenFile function, 297
- ZwOpenKey function, 297
- ZwOpenProcess function, 297
- ZwOpenSection function, 297
- ZwQuerySystemInformation function, 87–88, 185, 299–301
- ZwSetSystemInformation function, 296–297
- ZwSetValueKey function, 297
- ZwWriteFile function, 151