

Contents

Preface	xxi
---------------	-----

Part I—Foundations

Chapter 1	Empty Cup Mind	3
1.1	An Uninvited Guest.....	3
1.2	Distilling a More Precise Definition.....	4
	The Attack Cycle	5
	The Role of Rootkits in the Attack Cycle.....	7
	Single-Stage Versus Multistage Droppers	8
	Other Means of Deployment	9
	A Truly Pedantic Definition.....	10
	Don't Confuse Design Goals with Implementation	12
	Rootkit Technology as a Force Multiplier.....	13
	The Kim Philby Metaphor: Subversion Versus Destruction	13
	Why Use Stealth Technology? Aren't Rootkits Detectable?.....	14
1.3	Rootkits != Malware.....	15
	Infectious Agents	15
	Adware and Spyware.....	16
	Rise of the Botnets.....	17
	Enter: Conficker.....	18
	Malware Versus Rootkits.....	18
1.4	Who Is Building and Using Rootkits?.....	19
	Marketing	19
	Digital Rights Management.....	20
	It's Not a Rootkit, It's a Feature	20
	Law Enforcement	21
	Industrial Espionage	22
	Political Espionage	23
	Cybercrime	24
	Who Builds State-of-the-Art Rootkits?.....	26
	The Moral Nature of a Rootkit	26
1.5	Tales from the Crypt: Battlefield Triage.....	27
1.6	Conclusions	32

2	Overview of Anti-Forensics.....	35
	Everyone Has a Budget: Buy Time	36
2.1	Incident Response.....	36
	Intrusion Detection System (and Intrusion Prevention System)	36
	Odd Behavior.....	37
	Something Breaks.....	37
2.2	Computer Forensics.....	38
	Aren't Rootkits Supposed to Be Stealthy? Why AF?.....	38
	Assuming the Worst-Case Scenario.....	39
	Classifying Forensic Techniques: First Method	40
	Classifying Forensic Techniques: Second Method.....	41
	Live Response	41
	When Powering Down Isn't an Option	43
	The Debate over Pulling the Plug.....	43
	To Crash Dump or Not to Crash Dump.....	44
	Postmortem Analysis	44
	Non-Local Data	45
2.3	AF Strategies	45
	Data Destruction.....	46
	Data Concealment	47
	Data Transformation.....	47
	Data Fabrication	48
	Data Source Elimination.....	48
2.4	General Advice for AF Techniques	48
	Use Custom Tools.....	48
	Low and Slow Versus Scorched Earth.....	49
	Shun Instance-Specific Attacks	49
	Use a Layered Defense.....	50
2.5	John Doe Has the Upper Hand.....	50
	Attackers Can Focus on Attacking	50
	Defenders Face Institutional Challenges	51
	Security Is a Process (and a Boring One at That).....	51
	Ever-Increasing Complexity.....	51
2.6	Conclusions	53
3	Hardware Briefing.....	55
3.1	Physical Memory.....	55
3.2	IA-32 Memory Models.....	58

	Flat Memory Model.....	58
	Segmented Memory Model.....	59
	Modes of Operation.....	59
3.3	Real Mode.....	60
	Case Study: MS-DOS.....	62
	Isn't This a Waste of Time? Why Study Real Mode?.....	64
	The Real-Mode Execution Environment.....	65
	Real-Mode Interrupts.....	67
	Segmentation and Program Control.....	70
	Case Study: Dumping the IVT.....	72
	Case Study: Logging Keystrokes with a TSR.....	73
	Case Study: Hiding the TSR.....	78
	Case Study: Patching the TREE.COM Command.....	82
	Synopsis.....	86
3.4	Protected Mode.....	87
	The Protected-Mode Execution Environment.....	87
	Protected-Mode Segmentation.....	90
	Protected-Mode Paging.....	94
	Paging with Address Extension.....	96
	A Closer Look at the Tables.....	98
	A Closer Look at the Control Registers.....	100
3.5	Implementing Memory Protection.....	102
	Protection Through Segmentation.....	102
	Limit Checks.....	103
	Type Checks.....	103
	Privilege Checks.....	103
	Restricted Instruction Checks.....	105
	Gate Descriptors.....	106
	The Protected-Mode Interrupt Table.....	109
	Protection Through Paging.....	110
	Summary.....	112
Chapter 4	System Briefing.....	115
4.1	Physical Memory under Windows.....	116
	Land of the Lost (Memory).....	118
	How Windows Uses Physical Address Extension.....	118
	Pages, Page Frames, and Page Frame Numbers.....	120
4.2	Segmentation and Paging under Windows.....	120

	Segmentation	121
	Paging	123
	Linear to Physical Address Translation	127
	A Quicker Approach	128
	Comments on EPROCESS and KPROCESS	128
4.3	User Space and Kernel Space	130
	4-Gigabyte Tuning (4GT)	130
	To Each His Own	131
	Jumping the Fence	133
	User-Space Topography	133
	Kernel-Space Dynamic Allocation	135
	Address Windowing Extension	136
	PAE Versus 4GT Versus AWE	137
4.4	User Mode and Kernel Mode	137
	How Versus Where	137
	Kernel-Mode Components	139
	User-Mode Components	141
4.5	Other Memory Protection Features	144
	Data Execution Prevention	144
	Address Space Layout Randomization	148
	/GS Compiler Option	151
	/SAFESEH Linker Option	155
4.6	The Native API	155
	The IVT Grows Up	156
	A Closer Look at the IDT	157
	System Calls via Interrupt	159
	The SYSENTER Instruction	159
	The System Service Dispatch Tables	160
	Enumerating the Native API	163
	Nt*() Versus Zw*() System Calls	164
	The Life Cycle of a System Call	166
	Other Kernel-Mode Routines	168
	Kernel-Mode API Documentation	172
4.7	The BOOT Process	174
	Startup for BIOS Firmware	175
	Startup for EFI Firmware	177
	The Windows Boot Manager	177

	The Windows Boot Loader.....	178
	Initializing the Executive.....	181
	The Session Manager	182
	Wininit.exe	184
	Winlogon.exe.....	184
	Boot Process Recap	185
4.8	Design Decisions.....	186
	Hiding in a Crowd: Type 0	188
	Active Concealment: Type I and Type II.....	188
	Jumping Out of Bounds: Type III.....	190
	The Road Ahead	191
Chapter 5	Tools of the Trade.....	193
5.1	Development Tools.....	193
	Diagnostic Tools	194
	Disk-Imaging Tools	195
	For Faster Relief: Virtual Machines	196
	Tool Roundup	197
5.2	Debuggers.....	198
	Configuring CDB.exe	201
	Symbol Files.....	201
	Windows Symbols.....	202
	Invoking CDB.exe.....	203
	Controlling CDB.exe.....	204
	Useful Debugger Commands	205
	Examine Symbols Command (x).....	206
	List Loaded Modules (!m and !lmi)	207
	Display Type Command (dt)	209
	Unassemble Command (u).....	209
	Display Commands (d*)	210
	Registers Command (r)	212
5.3	The KD.exe Kernel Debugger.....	212
	Different Ways to Use a Kernel Debugger.....	212
	Physical Host-Target Configuration.....	215
	Preparing the Hardware.....	215
	Preparing the Software	218
	Launching a Kernel-Debugging Session	219

	Controlling the Target.....	221
	Virtual Host–Target Configuration	222
	Useful Kernel-Mode Debugger Commands	224
	List Loaded Modules Command (lm)	224
	!process.....	225
	Registers Command (r)	227
	Working with Crash Dumps	227
	Method No. 1: PS/2 Keyboard Trick.....	228
	Method No. 2: KD.exe Command.....	230
	Method No. 3: NotMyFault.exe.....	230
	Crash Dump Analysis	231
Chapter 6	Life in Kernel Space	233
6.1	A KMD Template	234
	Kernel-Mode Drivers: The Big Picture	234
	WDK Frameworks.....	236
	A Truly Minimal KMD.....	236
	Handling IRPs	240
	Communicating with User-Mode Code.....	245
	Sending Commands from User Mode	249
6.2	Loading a KMD.....	252
6.3	The Service Control Manager	253
	Using sc.exe at the Command Line.....	253
	Using the SCM Programmatically.....	255
	Registry Footprint.....	257
6.4	Using an Export Driver	258
6.5	Leveraging an Exploit in the Kernel	262
6.6	Windows Kernel-Mode Security	263
	Kernel-Mode Code Signing (KMCS).....	263
	KMCS Countermeasures	265
	Kernel Patch Protection (KPP).....	267
	KPP Countermeasures	268
6.7	Synchronization.....	269
	Interrupt Request Levels.....	269
	Deferred Procedure Calls	273
	Implementation.....	274
6.8	Conclusions	280

Part II: Postmortem

Chapter 7	Defeating Disk Analysis.....	283
7.1	Postmortem Investigation: An Overview	283
7.2	Forensic Duplication	285
	Countermeasures: Reserved Disk Regions.....	288
7.3	Volume Analysis.....	289
	Storage Volumes under Windows.....	289
	Manual Volume Analysis.....	291
	Countermeasures: Partition Table Destruction	293
	Raw Disk Access under Windows.....	293
	Raw Disk Access: Exceptions to the Rule.....	295
7.4	File System Analysis	298
	Recovering Deleted Files	298
	Recovering Deleted Files: Countermeasures.....	299
	Enumerating ADSs	301
	Enumerating ADSs: Countermeasures	302
	Recovering File System Objects	303
	Recovering File System Objects: Countermeasures.....	303
	Out-of-Band Concealment.....	304
	In-Band Concealment.....	310
	Enter: FragFS.....	321
	Application-Level Concealment.....	322
	Acquiring Metadata	323
	Acquiring Metadata: Countermeasures	327
	Altering Time Stamps.....	327
	Altering Checksums	330
	Identifying Known Files.....	330
	Cross-Time Versus Cross-View Diffs.....	332
	Identifying Known Files: Countermeasures	332
7.5	File Signature Analysis.....	334
	File Signature Analysis: Countermeasures	335
7.6	Conclusions	336
Chapter 8	Defeating Executable Analysis.....	337
8.1	Static Analysis	337
	Scan for Related Artifacts.....	338
	Verify Digital Signatures	338

	Dump String Data.....	339
	Inspect File Headers	340
	Disassembly and Decompilation	341
8.2	Subverting Static Analysis	343
	Data Transformation: Armoring	344
	Armoring: Cryptors	344
	Key Management.....	352
	Armoring: Packers.....	353
	Armoring: Metamorphic Code	355
	The Need for Custom Tools.....	359
	The Argument Against Armoring	360
	Data Fabrication	360
	False-Flag Attacks	363
	Data Source Elimination: Multistage Loaders	364
	Defense In-depth	365
8.3	Runtime Analysis	366
	The Working Environment	366
	Manual Versus Automated Runtime Analysis	369
	Manual Analysis: Basic Outline	370
	Manual Analysis: Tracing.....	371
	Manual Analysis: Memory Dumping	373
	Manual Analysis: Capturing Network Activity	375
	Automated Analysis.....	376
	Composition Analysis at Runtime	377
8.4	Subverting Runtime Analysis.....	378
	Tracing Countermeasures	379
	API Tracing: Evading Detour Patches.....	380
	API Tracing: Multistage Loaders	386
	Instruction-Level Tracing: Attacking the Debugger.....	386
	Break Points.....	386
	Detecting a User-Mode Debugger.....	387
	Detecting a Kernel-Mode Debugger	390
	Detecting a User-Mode or a Kernel-Mode Debugger	391
	Detecting Debuggers via Code Checksums	392
	The Argument Against Anti-Debugger Techniques.....	392
	Instruction-Level Tracing: Obfuscation	393
	Obfuscating Application Data	394

	Obfuscating Application Code	395
	Hindering Automation	398
	Countering Runtime Composition Analysis	400
8.5	Conclusions	400

Part III: Live Response

Chapter 9	Defeating Live Response	405
	Autonomy: The Coin of the Realm	406
	Learning the Hard Way: DDefy	407
	The Vendors Wise Up: Memoryze	411
9.1	Live Incident Response: The Basic Process	412
9.2	User-Mode Loaders (UMLs)	417
	UMLs That Subvert the Existing APIs	417
	The Argument Against Loader API Mods	418
	The Windows PE File Format at 10,000 Feet	419
	Relative Virtual Addresses	420
	PE File Headers	421
	The Import Data Section (.idata)	424
	The Base Relocation Section (.reloc)	427
	Implementing a Stand-Alone UML	429
9.3	Minimizing Loader Footprint	434
	Data Contraception: Ode to The Grugq	434
	The Next Step: Loading via Exploit	435
9.4	The Argument Against Stand-Alone PE Loaders	435
Chapter 10	Building Shellcode in C	437
	Why Shellcode Rootkits?	438
	Does Size Matter?	439
10.1	User-Mode Shellcode	440
	Visual Studio Project Settings	441
	Using Relative Addresses	443
	Finding Kernel32.dll: Journey into the TEB and PEB	446
	Augmenting the Address Table	452
	Parsing the kernel32.dll Export Table	453
	Extracting the Shellcode	456
	The Danger Room	460

	Build Automation	462
10.2	Kernel-Mode Shellcode.....	462
	Project Settings: \$(NTMAKEENV)\makefile.new	463
	Project Settings: SOURCES.....	464
	Address Resolution.....	465
	Loading Kernel-Mode Shellcode	468
10.3	Special Weapons and Tactics.....	471
10.4	Looking Ahead	473
Chapter 11	Modifying Call Tables	475
11.1	Hooking in User Space: The IAT	478
	DLL Basics	478
	Accessing Exported Routines.....	480
	Injecting a DLL.....	482
	Walking an IAT from a PE File on Disk.....	487
	Hooking the IAT	492
11.2	Call Tables in Kernel Space	496
11.3	Hooking the IDT	497
	Handling Multiple Processors: Solution #1	499
	Naked Routines	503
	Issues with Hooking the IDT.....	506
11.4	Hooking Processor MSRs	507
	Handling Multiple Processors: Solution #2.....	509
11.5	Hooking the SSDT	514
	Disabling the WP Bit: Technique #1	515
	Disabling the WP Bit: Technique #2	517
	Hooking SSDT Entries	519
	SSDT Example: Tracing System Calls.....	520
	SSDT Example: Hiding a Process.....	523
	SSDT Example: Hiding a Network Connection.....	529
11.6	Hooking IRP Handlers	530
11.7	Hooking the GDT: Installing a Call Gate.....	533
	Ode to Dreg	542
11.8	Hooking Countermeasures	542
	Checking for Kernel-Mode Hooks	543
	Checking IA32_SYSENTER_EIP.....	546
	Checking INT 0x2E	548

	Checking the SSDT	549
	Checking IRP Handlers	550
	Checking for User-Mode Hooks.....	552
	Parsing the PEB: Part I.....	555
	Parsing the PEB: Part II.....	558
11.9	Counter-Countermeasures	558
	Assuming the Worst Case.....	559
	Worst-Case Countermeasure #1	559
	Worst-Case Countermeasure #2	559
Chapter 12	Modifying Code.....	561
	Types of Patching	562
	In-Place Patching.....	562
	Detour Patching	563
	Prologue and Epilogue Detours.....	565
	Detour Jumps.....	566
12.1	Tracing Calls	567
	Detour Implementation.....	572
	Acquire the Address of the NtSetValueKey()	575
	Initialize the Patch Metadata Structure.....	576
	Verify the Original Machine Code Against a Known Signature	577
	Save the Original Prologue and Epilogue Code	578
	Update the Patch Metadata Structure	578
	Lock Access and Disable Write-Protection	579
	Inject the Detours	579
	The Prologue Detour	580
	The Epilogue Detour	582
	Postgame Wrap-Up.....	586
12.2	Subverting Group Policy	586
	Detour Implementation.....	588
	Initializing the Patch Metadata Structure	588
	The Epilogue Detour	589
	Mapping Registry Values to Group Policies.....	593
12.3	Bypassing Kernel-Mode API Loggers	595
	Fail-Safe Evasion.....	596
	Kicking It Up a Notch	600
12.4	Instruction Patching Countermeasures	600

Chapter 13	Modifying Kernel Objects	603
13.1	The Cost of Invisibility	603
	Issue #1: The Steep Learning Curve	604
	Issue #2: Concurrency	604
	Issue #3: Portability and Pointer Arithmetic	605
	Branding the Technique: DKOM	607
	Objects?	607
13.2	Revisiting the EPROCESS Object	608
	Acquiring an EPROCESS Pointer	608
	Relevant Fields in EPROCESS	611
	UniqueProcessId	611
	ActiveProcessLinks	611
	Token	613
	ImageFileName	613
13.3	The DRIVER_SECTION Object	613
13.4	The Token Object	615
	Authorization on Windows	616
	Locating the Token Object	619
	Relevant Fields in the Token Object	621
13.5	Hiding a Process	625
13.6	Hiding a Driver	630
13.7	Manipulating the Access Token	634
13.8	Using No-FU	637
13.9	Kernel-Mode Callbacks	640
13.10	Countermeasures	643
	Cross-View Detection	643
	High-Level Enumeration: CreateToolhelp32Snapshot()	644
	High-Level Enumeration: PID Bruteforce	646
	Low-Level Enumeration: Processes	649
	Low-Level Enumeration: Threads	651
	Related Software	658
	Field Checksums	659
13.11	Counter-Countermeasures	659
	The Best Defense: Starve the Opposition	660
	Commentary: Transcending the Two-Ring Model	661
	The Last Line of Defense	662

Chapter 14	Covert Channels	663
14.1	Common Malware Channels	663
	Internet Relay Chat	664
	Peer-to-Peer Communication	664
	HTTP	665
14.2	Worst-Case Scenario: Full Content Data Capture	668
	Protocol Tunneling	669
	DNS	670
	ICMP	670
	Peripheral Issues	672
14.3	The Windows TCP/IP Stack	673
	Windows Sockets 2	674
	Raw Sockets	675
	Winsock Kernel API	676
	NDIS	677
	Different Tools for Different Jobs	680
14.4	DNS Tunneling	680
	DNS Query	680
	DNS Response	683
14.5	DNS Tunneling: User Mode	685
14.6	DNS Tunneling: WSK Implementation	689
	Initialize the Application's Context	696
	Create a Kernel-Mode Socket	697
	Determine a Local Transport Address	698
	Bind the Socket to the Transport Address	699
	Set the Remote Address (the C2 Client)	700
	Send the DNS Query	702
	Receive the DNS Response	703
14.7	NDIS Protocol Drivers	705
	Building and Running the NDISProt 6.0 Example	707
	An Outline of the Client Code	710
	An Outline of the Driver Code	713
	The Protocol*() Routines	716
	Missing Features	721
14.8	Passive Covert Channels	722

Chapter 15	Going Out-of-Band	725
	Ways to Jump Out-of-Band	726
15.1	Additional Processor Modes	726
	System Management Mode	727
	Rogue Hypervisors	732
	White Hat Countermeasures	736
	Rogue Hypervisors Versus SMM Rootkits	737
15.2	Firmware	738
	Mobo BIOS	738
	ACPI Components	741
	Expansion ROM	742
	UEFI Firmware	744
15.3	Lights-Out Management Facilities	745
15.4	Less Obvious Alternatives	745
	Onboard Flash Storage	746
	Circuit-Level Tomfoolery	746
15.5	Conclusions	748

Part IV: Summation

Chapter 16	The Tao of Rootkits	753
	The Dancing Wu Li Masters	753
	When a Postmortem Isn't Enough	755
	The Battlefield Shifts Again	757
16.1	Core Stratagems	757
	Respect Your Opponent	758
	Five Point Palm Exploding Heart Technique	758
	Resist the Urge to Smash and Grab	759
	Study Your Target	760
16.2	Identifying Hidden Doors	760
	On Dealing with Proprietary Systems	761
	Staking Out the Kernel	761
	Kingpin: Hardware Is the New Software	762
	Leverage Existing Research	762
16.3	Architectural Precepts	763
	Load First, Load Deep	763
	Strive for Autonomy	764
	Butler Lampson: Separate Mechanism from Policy	764

16.4	Engineering a Rootkit.....	764
	Stealth Versus Development Effort	765
	Use Custom Tools.....	765
	Stability Counts: Invest in Best Practices.....	766
	Gradual Enhancement	766
	Failover: The Self-Healing Rootkit	768
16.5	Dealing with an Infestation	768
	Index	771
	Photo Credits.....	783