

Table of Contents

ABOUT THE AUTHOR **xxi**

PREFACE **xxii**

ACKNOWLEDGMENTS **xxv**

PART I Strategies of the Attacker 1

1 INTRODUCTION TO THE GAMES OF NATURE 3

1.1 Early Models of Self-Replicating Structures 4

1.1.1 John von Neumann: Theory of Self-Reproducing Automata 5

1.1.2 Fredkin: Reproducing Structures 7

1.1.3 Conway: Game of Life 8

1.1.4 Core War: The Fighting Programs 12

1.2 Genesis of Computer Viruses 17

1.3 Automated Replicating Code: The Theory and Definition of
Computer Viruses 18

References 20

2 THE FASCINATION OF MALICIOUS CODE ANALYSIS 23

2.1 Common Patterns of Virus Research 26

2.2 Antivirus Defense Development 27

2.3 Terminology of Malicious Programs 28

2.3.1 Viruses 28

2.3.2 Worms 29

2.3.3 Logic Bombs 30

2.3.4 Trojan Horses	31
2.3.5 Germs	32
2.3.6 Exploits	33
2.3.7 Downloaders	33
2.3.8 Dialers	33
2.3.9 Droppers	33
2.3.10 Injectors	34
2.3.11 Auto-Rooters	34
2.3.12 Kits (Virus Generators)	34
2.3.13 Spammer Programs	35
2.3.14 Flooders	35
2.3.15 Keyloggers	36
2.3.16 Rootkits	36
2.4 Other Categories	36
2.4.1 Joke Programs	37
2.4.2 Hoaxes: Chain Letters	37
2.4.3 Other Pests: Adware and Spyware	38
2.5 Computer Malware Naming Scheme	38
2.5.1 <family_name>	40
2.5.2 <malware_type>://	40
2.5.3 <platform>/	40
2.5.4 .<group_name>	41
2.5.5 <infective_length>	41
2.5.6 <variant>	41
2.5.7 [<devolution>]	41
2.5.8 <modifiers>	41
2.5.9 :<locale_specifier>	42
2.5.10 #<packer>	42
2.5.11 @m or @mm	42
2.5.12 !<vendor-specific_comment>	42
2.6 Annotated List of Officially Recognized Platform Names	42
References	46

3 MALICIOUS CODE ENVIRONMENTS 49

3.1 Computer Architecture Dependency	52
3.2 CPU Dependency	53
3.3 Operating System Dependency	55
3.4 Operating System Version Dependency	55

- 3.5 File System Dependency 56
 - 3.5.1 Cluster Viruses 56
 - 3.5.2 NTFS Stream Viruses 58
 - 3.5.3 NTFS Compression Viruses 59
 - 3.5.4 ISO Image Infection 59
- 3.6 File Format Dependency 59
 - 3.6.1 COM Viruses on DOS 59
 - 3.6.2 EXE Viruses on DOS 60
 - 3.6.3 NE (New Executable) Viruses on 16-bit Windows and OS/2 60
 - 3.6.4 LX Viruses on OS/2 60
 - 3.6.5 PE (Portable Executable) Viruses on 32-bit Windows 61
 - 3.6.6 ELF (Executable and Linking Format) Viruses on UNIX 64
 - 3.6.7 Device Driver Viruses 65
 - 3.6.8 Object Code and LIB Viruses 66
- 3.7 Interpreted Environment Dependency 66
 - 3.7.1 Macro Viruses in Microsoft Products 66
 - 3.7.2 REXX Viruses on IBM Systems 78
 - 3.7.3 DCL (DEC Command Language) Viruses on DEC/VMS 79
 - 3.7.4 Shell Scripts on UNIX (csh, ksh, and bash) 80
 - 3.7.5 VBScript (Visual Basic Script) Viruses on Windows Systems 81
 - 3.7.6 BATCH Viruses 82
 - 3.7.7 Instant Messaging Viruses in mIRC, PIRCH scripts 83
 - 3.7.8 SuperLogo Viruses 83
 - 3.7.9 JScript Viruses 85
 - 3.7.10 Perl Viruses 86
 - 3.7.11 WebTV Worms in JellyScript Embedded in HTML Mail 86
 - 3.7.12 Python Viruses 87
 - 3.7.13 VIM Viruses 87
 - 3.7.14 EMACS Viruses 87
 - 3.7.15 TCL Viruses 87
 - 3.7.16 PHP Viruses 88
 - 3.7.17 MapInfo Viruses 88
 - 3.7.18 ABAP Viruses on SAP 89
 - 3.7.19 Help File Viruses on Windows—When You Press F1... 89
 - 3.7.20 JScript Threats in Adobe PDF 90
 - 3.7.21 AppleScript Dependency 90
 - 3.7.22 ANSI Dependency 90
 - 3.7.23 Macromedia Flash ActionScript Threats 91

3.7.24	HyperTalk Script Threats	91
3.7.25	AutoLisp Script Viruses	92
3.7.26	Registry Dependency	93
3.7.27	PIF and LNK Dependency	94
3.7.28	Lotus Word Pro Macro Viruses	94
3.7.29	AmiPro Document Viruses	94
3.7.30	Corel Script Viruses	95
3.7.31	Lotus 1-2-3 Macro Dependency	96
3.7.32	Windows Installation Script Dependency	96
3.7.33	AUTORUN.INF and Windows INI File Dependency	97
3.7.34	HTML (Hypertext Markup Language) Dependency	97
3.8	Vulnerability Dependency	98
3.9	Date and Time Dependency	98
3.10	JIT Dependency: Microsoft .NET Viruses	99
3.11	Archive Format Dependency	100
3.12	File Format Dependency Based on Extension	101
3.13	Network Protocol Dependency	102
3.14	Source Code Dependency	102
3.14.1	Source Code Trojans	104
3.15	Resource Dependency on Mac and Palm Platforms	104
3.16	Host Size Dependency	105
3.17	Debugger Dependency	106
3.17.1	Intended Threats that Rely on a Debugger	108
3.18	Compiler and Linker Dependency	108
3.19	Device Translator Layer Dependency	109
3.20	Embedded Object Insertion Dependency	112
3.21	Self-Contained Environment Dependency	113
3.22	Multipartite Viruses	115
3.23	Conclusion	116
	References	117

4 CLASSIFICATION OF INFECTION STRATEGIES 121

4.1	Boot Viruses	122
4.1.1	Master Boot Record (MBR) Infection Techniques	124
4.1.2	DOS BOOT Record (DBR) Infection Techniques	126

4.1.3	Boot Viruses That Work While Windows 95 Is Active	129
4.1.4	Possible Boot Image Attacks in Network Environments	129
4.2	File Infection Techniques	129
4.2.1	Overwriting Viruses	130
4.2.2	Random Overwriting Viruses	131
4.2.3	Appending Viruses	132
4.2.4	Prepending Viruses	133
4.2.5	Classic Parasitic Viruses	135
4.2.6	Cavity Viruses	136
4.2.7	Fractionated Cavity Viruses	137
4.2.8	Compressing Viruses	139
4.2.9	Amoeba Infection Technique	140
4.2.10	Embedded Decryptor Technique	141
4.2.11	Embedded Decryptor and Virus Body Technique	142
4.2.12	Obfuscated Tricky Jump Technique	143
4.2.13	Entry-Point Obscuring (EPO) Viruses	145
4.2.14	Possible Future Infection Techniques: Code Builders	155
4.3	An In-Depth Look at Win32 Viruses	157
4.3.1	The Win32 API and Platforms That Support It	158
4.3.2	Infection Techniques on 32-Bit Windows	160
4.3.3	Win32 and Win64 Viruses: Designed for Microsoft Windows?	181
4.4	Conclusion	183
	References	184

5 CLASSIFICATION OF IN-MEMORY STRATEGIES 185

5.1	Direct-Action Viruses	186
5.2	Memory-Resident Viruses	186
5.2.1	Interrupt Handling and Hooking	188
5.2.2	Hook Routines on INT 13h (Boot Viruses)	191
5.2.3	Hook Routines on INT 21h (File Viruses)	193
5.2.4	Common Memory Installation Techniques Under DOS	196
5.2.5	Stealth Viruses	199
5.2.6	Disk Cache and System Buffer Infection	209
5.3	Temporary Memory-Resident Viruses	210
5.4	Swapping Viruses	211
5.5	Viruses in Processes (in User Mode)	211
5.6	Viruses in Kernel Mode (Windows 9x/Me)	212

5.7 Viruses in Kernel Mode (Windows NT/2000/XP) 213

5.8 In-Memory Injectors over Networks 215

References 216

6 BASIC SELF-PROTECTION STRATEGIES 217

6.1 Tunneling Viruses 218

6.1.1 Memory Scanning for Original Handler 218

6.1.2 Tracing with Debug Interfaces 219

6.1.3 Code Emulation-Based Tunneling 219

6.1.4 Accessing the Disk Using Port I/O 219

6.1.5 Using Undocumented Functions 219

6.2 Armored Viruses 220

6.2.1 Antidisassembly 220

6.2.2 Encrypted Data 221

6.2.3 Code Confusion to Avoid Analysis 222

6.2.4 Opcode Mixing-Based Code Confusion 223

6.2.5 Using Checksum 224

6.2.6 Compressed, Obfuscated Code 225

6.2.7 Antidebugging 226

6.2.8 Antiheuristics 234

6.2.9 Antiemulation Techniques 242

6.2.10 Antigoat Viruses 247

6.3 Aggressive Retroviruses 247

References 250

7 ADVANCED CODE EVOLUTION TECHNIQUES AND COMPUTER VIRUS GENERATOR KITS 251

7.1 Introduction 252

7.2 Evolution of Code 252

7.3 Encrypted Viruses 253

7.4 Oligomorphic Viruses 259

7.5 Polymorphic Viruses 261

7.5.1 The 1260 Virus 261

7.5.2 The Dark Avenger Mutation Engine (MtE) 262

7.5.3 32-Bit Polymorphic Viruses 264

7.6 Metamorphic Viruses 269

7.6.1 What Is a Metamorphic Virus? 269

7.6.2 Simple Metamorphic Viruses	270
7.6.3 More Complex Metamorphic Viruses and Permutation Techniques	273
7.6.4 Mutating Other Applications: The Ultimate Virus Generator?	276
7.6.5 Advanced Metamorphic Viruses: Zmist	277
7.6.6 {W32, Linux}/Simile: A Metamorphic Engine Across Systems	281
7.6.7 The Dark Future—MSIL Metamorphic Viruses	286
7.7 Virus Construction Kits	288
7.7.1 VCS (Virus Construction Set)	289
7.7.2 GenVir	289
7.7.3 VCL (Virus Creation Laboratory)	289
7.7.4 PS-MPC (Phalcon-Skism Mass-Produced Code Generator)	290
7.7.5 NGVCK (Next Generation Virus Creation Kit)	291
7.7.6 Other Kits and Mutators	291
7.7.7 How to Test a Virus Construction Tool?	293
References	293
8 CLASSIFICATION ACCORDING TO PAYLOAD	295
8.1 No-Payload	296
8.2 Accidentally Destructive Payload	297
8.3 Nondestructive Payload	297
8.4 Somewhat Destructive Payload	300
8.5 Highly Destructive Payload	301
8.5.1 Viruses That Overwrite Data	301
8.5.2 Data Diddlers	302
8.5.3 Viruses That Encrypt Data: The “Good,” the Bad, and the Ugly	303
8.5.4 Hardware Destroyers	305
8.6 DoS (Denial of Service) Attacks	306
8.7 Data Stealers: Making Money with Viruses	308
8.7.1 Phishing Attacks	308
8.7.2 Backdoor Features	309
8.8 Conclusion	312
References	312
9 STRATEGIES OF COMPUTER WORMS	313
9.1 Introduction	314
9.2 The Generic Structure of Computer Worms	315
9.2.1 Target Locator	315

9.2.2	Infection Propagator	315
9.2.3	Remote Control and Update Interface	316
9.2.4	Life-Cycle Manager	316
9.2.5	Payload	318
9.2.6	Self-Tracking	318
9.3	Target Locator	319
9.3.1	E-Mail Address Harvesting	319
9.3.2	Network Share Enumeration Attacks	324
9.3.3	Network Scanning and Target Fingerprinting	326
9.4	Infection Propagators	331
9.4.1	Attacking Backdoor-Compromised Systems	331
9.4.2	Peer-to-Peer Network Attacks	332
9.4.3	Instant Messaging Attacks	333
9.4.4	E-Mail Worm Attacks and Deception Techniques	333
9.4.5	E-Mail Attachment Inserters	334
9.4.6	SMTP Proxy-Based Attacks	334
9.4.7	SMTP Attacks	335
9.4.8	SMTP Propagation on Steroids Using MX Queries	338
9.4.9	NNTP (Network News Transfer Protocol) Attacks	338
9.5	Common Worm Code Transfer and Execution Techniques	338
9.5.1	Executable Code-Based Attacks	339
9.5.2	Links to Web Sites or Web Proxies	339
9.5.3	HTML-Based Mail	340
9.5.4	Remote Login-Based Attacks	341
9.5.5	Code Injection Attacks	341
9.5.6	Shell Code-Based Attacks	342
9.6	Update Strategies of Computer Worms	345
9.6.1	Authenticated Updates on the Web or Newsgroups	346
9.6.2	Backdoor-Based Updates	351
9.7	Remote Control via Signaling	351
9.7.1	Peer-to-Peer Network Control	352
9.8	Intentional and Accidental Interactions	354
9.8.1	Cooperation	354
9.8.2	Competition	357
9.8.3	The Future: A Simple Worm Communication Protocol?	359
9.9	Wireless Mobile Worms	359
	References	361

10 EXPLOITS, VULNERABILITIES, AND BUFFER OVERFLOW ATTACKS 365

10.1 Introduction 366

10.1.1 Definition of Blended Attack 366

10.1.2 The Threat 366

10.2 Background 367

10.3 Types of Vulnerabilities 368

10.3.1 Buffer Overflows 368

10.3.2 First-Generation Attacks 369

10.3.3 Second-Generation Attacks 371

10.3.4 Third-Generation Attacks 378

10.4 Current and Previous Threats 394

10.4.1 The Morris Internet Worm, 1988 (Stack Overflow to Run Shellcode) 395

10.4.2 Linux/ADM, 1998 (“Copycatting” the Morris Worm) 397

10.4.3 The CodeRed Outbreak, 2001 (The Code Injection Attack) 398

10.4.4 Linux/Slapper Worm, 2002 (A Heap Overflow Example) 401

10.4.5 W32/Slammer Worm, January 2003 (The Mini Worm) 407

10.4.6 Blaster Worm, August 2003 (Shellcode-Based Attack on Win32) 410

10.4.7 Generic Buffer Overflow Usage in Computer Viruses 413

10.4.8 Description of W32/Badtrans.B@mm 414

10.4.9 Exploits in W32/Nimda.A@mm 414

10.4.10 Description of W32/Bolzano 415

10.4.11 Description of VBS/Bubbleboy 417

10.4.12 Description of W32/Blebla 418

10.5 Summary 419

References 420

Part II STRATEGIES OF THE DEFENDER 423

11 ANTIVIRUS DEFENSE TECHNIQUES 425

11.1 First-Generation Scanners 428

11.1.1 String Scanning 428

11.1.2 Wildcards 430

11.1.3 Mismatches 432

11.1.4 Generic Detection 432

11.1.5 Hashing 432

11.1.6 Bookmarks 433

11.1.7 Top-and-Tail Scanning 435

- 11.1.8 Entry-Point and Fixed-Point Scanning 435
- 11.1.9 Hyperfast Disk Access 436
- 11.2 Second-Generation Scanners 437
 - 11.2.1 Smart Scanning 437
 - 11.2.2 Skeleton Detection 437
 - 11.2.3 Nearly Exact Identification 437
 - 11.2.4 Exact Identification 439
- 11.3 Algorithmic Scanning Methods 441
 - 11.3.1 Filtering 443
 - 11.3.2 Static Decryptor Detection 444
 - 11.3.3 The X-RAY Method 446
- 11.4 Code Emulation 451
 - 11.4.1 Encrypted and Polymorphic Virus Detection Using Emulation 455
 - 11.4.2 Dynamic Decryptor Detection 459
- 11.5 Metamorphic Virus Detection Examples 461
 - 11.5.1 Geometric Detection 461
 - 11.5.2 Disassembling Techniques 462
 - 11.5.3 Using Emulators for Tracing 463
- 11.6 Heuristic Analysis of 32-Bit Windows Viruses 467
 - 11.6.1 Code Execution Starts in the Last Section 468
 - 11.6.2 Suspicious Section Characteristics 468
 - 11.6.3 Virtual Size Is Incorrect in PE Header 468
 - 11.6.4 Possible “Gap” Between Sections 468
 - 11.6.5 Suspicious Code Redirection 469
 - 11.6.6 Suspicious Code Section Name 469
 - 11.6.7 Possible Header Infection 469
 - 11.6.8 Suspicious Imports from KERNEL32.DLL by Ordinal 469
 - 11.6.9 Import Address Table Is Patched 469
 - 11.6.10 Multiple PE Headers 469
 - 11.6.11 Multiple Windows Headers and Suspicious KERNEL32.DLL Imports 470
 - 11.6.12 Suspicious Relocations 470
 - 11.6.13 Kernel Look-Up 470
 - 11.6.14 Kernel Inconsistency 471
 - 11.6.15 Loading a Section into the VMM Address Space 471
 - 11.6.16 Incorrect Size of Code in Header 471
 - 11.6.17 Examples of Suspicious Flag Combinations 471

11.7	Heuristic Analysis Using Neural Networks	472
11.8	Regular and Generic Disinfection Methods	474
11.8.1	Standard Disinfection	475
11.8.2	Generic Decryptors	477
11.8.3	How Does a Generic Disinfector Work?	477
11.8.4	How Can the Disinfector Be Sure That the File Is Infected?	477
11.8.5	Where Is the Original End of the Host File?	477
11.8.6	How Many Virus Types Can We Handle This Way?	478
11.8.7	Examples of Heuristics for Generic Repair	479
11.8.8	Generic Disinfection Examples	480
11.9	Inoculation	481
11.10	Access Control Systems	482
11.11	Integrity Checking	484
11.11.1	False Positives	484
11.11.2	Clean Initial State	485
11.11.3	Speed	485
11.11.4	Special Objects	485
11.11.5	Necessity of Changed Objects	486
11.11.6	Possible Solutions	486
11.12	Behavior Blocking	487
11.13	Sand-Boxing	489
11.14	Conclusion	491
	References	491
12	MEMORY SCANNING AND DISINFECTION	495
12.1	Introduction	497
12.2	The Windows NT Virtual Memory System	499
12.3	Virtual Address Spaces	501
12.4	Memory Scanning in User Mode	505
12.4.1	The Secrets of NtQuerySystemInformation()	506
12.4.2	Common Processes and Special System Rights	507
12.4.3	Viruses in the Win32 Subsystem	508
12.4.4	Win32 Viruses That Allocate Private Pages	510
12.4.5	Native Windows NT Service Viruses	512
12.4.6	Win32 Viruses That Use a Hidden Window Procedure	512
12.4.7	Win32 Viruses That Are Part of the Executed Image Itself	512

- 12.5 Memory Scanning and Paging 515
 - 12.5.1 Enumerating Processes and Scanning File Images 517
- 12.6 Memory Disinfection 517
 - 12.6.1 Terminating a Particular Process That Contains Virus Code 518
 - 12.6.2 Detecting and Terminating Virus Threads 518
 - 12.6.3 Patching the Virus Code in the Active Pages 522
 - 12.6.4 How to Disinfect Loaded DLLs and Running Applications 523
- 12.7 Memory Scanning in Kernel Mode 523
 - 12.7.1 Scanning the User Address Space of Processes 523
 - 12.7.2 Determining NT Service API Entry Points 524
 - 12.7.3 Important NT Functions for Kernel-Mode Memory Scanning 525
 - 12.7.4 Process Context 526
 - 12.7.5 Scanning the Upper 2GB of Address Space 527
 - 12.7.6 How Can You Deactivate a Filter Driver Virus? 527
 - 12.7.7 Dealing with Read-Only Kernel Memory 529
 - 12.7.8 Kernel-Mode Memory Scanning on 64-Bit Platforms 530
- 12.8 Possible Attacks Against Memory Scanning 532
- 12.9 Conclusion and Future Work 534
- References 535

13 WORM-BLOCKING TECHNIQUES AND HOST-BASED INTRUSION PREVENTION 537

- 13.1 Introduction 538
 - 13.1.1 Script Blocking and SMTP Worm Blocking 539
 - 13.1.2 New Attacks to Block: CodeRed, Slammer 542
- 13.2 Techniques to Block Buffer Overflow Attacks 543
 - 13.2.1 Code Reviews 544
 - 13.2.2 Compiler-Level Solutions 545
 - 13.2.3 Operating System-Level Solutions and Run-Time Extensions 552
 - 13.2.4 Subsystem Extensions—Libsafe 554
 - 13.2.5 Kernel Mode Extensions 554
 - 13.2.6 Program Shepherding 556
- 13.3 Worm-Blocking Techniques 557
 - 13.3.1 Injected Code Detection 557
 - 13.3.2 Send Blocking: An Example of Blocking Self-Sending Code 563
 - 13.3.3 Exception Handler Validation 565
 - 13.3.4 Other Return-to-LIBC Attack Mitigation Techniques 569

13.3.5	"GOT" and "IAT" Page Attributes	574
13.3.6	High Number of Connections and Connection Errors	574
13.4	Possible Future Worm Attacks	575
13.4.1	A Possible Increase of Retroworms	576
13.4.2	"Slow" Worms Below the Radar	576
13.4.3	Polymorphic and Metamorphic Worms	576
13.4.4	Large-scale Damage	577
13.4.5	Automated Exploit Discovery—Learning from the Environment	578
13.5	Conclusion	578
	References	580
14	NETWORK-LEVEL DEFENSE STRATEGIES	583
14.1	Introduction	584
14.2	Using Router Access Lists	585
14.3	Firewall Protection	588
14.4	Network-Intrusion Detection Systems	591
14.5	Honeypot Systems	593
14.6	Counterattacks	596
14.7	Early Warning Systems	598
14.8	Worm Behavior Patterns on the Network	598
14.8.1	Capturing the Blaster Worm	598
14.8.2	Capturing the Linux/Slapper Worm	600
14.8.3	Capturing the W32/Sasser.D Worm	603
14.8.4	Capturing the Ping Requests of the W32/Welchia Worm	605
14.8.5	Detecting W32/Slammer and Related Exploits	607
14.9	Conclusion	609
	References	609
15	MALICIOUS CODE ANALYSIS TECHNIQUES	611
15.1	Your Personal Virus Analysis Laboratory	612
15.1.1	How to Get the Software?	614
15.2	Information, Information, Information	615
15.2.1	Architecture Guides	615
15.2.2	Knowledge Base	615
15.3	Dedicated Virus Analysis on VMWARE	616

15.4	The Process of Computer Virus Analysis	618
15.4.1	Preparation	619
15.4.2	Unpacking	625
15.4.3	Disassembling and Decryption	626
15.4.4	Dynamic Analysis Techniques	634
15.5	Maintaining a Malicious Code Collection	661
15.6	Automated Analysis: The Digital Immune System	661
	References	665

16 CONCLUSION 667

	Further Reading	669
	Information on Security and Early Warnings	669
	Security Updates	669
	Computer Worm Outbreak Statistics	670
	Computer Virus Research Papers	670
	Contact Information for Antivirus Vendors	670
	Antivirus Testers and Related Sites	672

INDEX 675