



The Art of Computer

# Virus Research and Defense

*"Of all the computer-related books I've read recently, this one influenced my thoughts about security the most. There is very little trustworthy information about computer viruses. Peter Szor is one of the best virus analysts in the world and has the perfect credentials to write this book."*

—Halvar Flake, Reverse Engineer, SABRE Security GmbH

**Peter Szor**



symantec.  
press



The Art of Computer  
**Virus Research  
and Defense**

*"Of all the computer-related books I've read recently, this one influenced my thoughts about security the most. There is very little trustworthy information about computer viruses. Peter Szor is one of the best virus analysts in the world and has the perfect credentials to write this book."*

—Halvar Flake, Reverse Engineer, SABRE Security GmbH

**Peter Szor**





# **The Art of Computer Virus Research and Defense**

**Peter Szor**

**↕ Addison-Wesley**

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco  
New York • Toronto • Montreal • London • Munich • Paris • Madrid  
Capetown • Sydney • Tokyo • Singapore • Mexico City

~~Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of trademark claim, the designations have been printed with initial capital letters or in all capitals.~~

The author and publisher have taken care in the preparation of this book, but make no expressed implied warranty of any kind and assume no responsibility for errors or omissions. No liability assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

Symantec Press Publisher: Linda McCarthy  
Editor in Chief: Karen Gettman  
Acquisitions Editor: Jessica Goldstein  
Cover Designer: Alan Clements  
Managing Editor: Gina Kanouse  
Senior Project Editor: Kristy Hart  
Copy Editor: Christal Andry  
Indexers: Cheryl Lenser and Larry Sweazy  
Compositor: Stickman Studio  
Manufacturing Buyer: Dan Uhrig

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases special sales, which may include electronic versions and/or custom covers and content particular your business, training goals, marketing focus, and branding interests. For more information, please contact:

U. S. Corporate and Government Sales  
(800) 382-3419  
[corpsales@pearsontechgroup.com](mailto:corpsales@pearsontechgroup.com)

For sales outside the U. S., please contact:

International Sales  
[international@pearsoned.com](mailto:international@pearsoned.com)

Visit us on the Web: [www.awprofessional.com](http://www.awprofessional.com)

Library of Congress Number: 2004114972

Copyright © 2005 Symantec Corporation

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, write to:

Pearson Education, Inc.  
Rights and Contracts Department  
One Lake Street  
Upper Saddle River, NJ 07458

ISBN 0-32-130454-3

Text printed in the United States on recycled paper at Phoenix BookTech in Hagerstown, Maryland.



---

*to Natalia*







# Table of Contents

---

[About the Author](#)

[Preface](#)

[Acknowledgments](#)

## **[Part I. Strategies of the Attacker](#)**

### **[1. Introduction to the Games of Nature](#)**

#### [1.1. Early Models of Self-Replicating Structures](#)

[1.1.1. John von Neumann: Theory of Self-Reproducing Automata](#)

[1.1.2. Fredkin: Reproducing Structures](#)

[1.1.3. Conway: Game of Life](#)

[1.1.4. Core War: The Fighting Programs](#)

#### [1.2. Genesis of Computer Viruses](#)

#### [1.3. Automated Replicating Code: The Theory and Definition of Computer Viruses](#)

[References](#)

### **[2. The Fascination of Malicious Code Analysis](#)**

#### [2.1. Common Patterns of Virus Research](#)

#### [2.2. Antivirus Defense Development](#)

#### [2.3. Terminology of Malicious Programs](#)

[2.3.1. Viruses](#)

[2.3.2. Worms](#)

[2.3.3. Logic Bombs](#)

[2.3.4. Trojan Horses](#)

[2.3.5. Germs](#)

[2.3.6. Exploits](#)

[2.3.7. Downloaders](#)

[2.3.8. Dialers](#)

[2.3.9. Droppers](#)

[2.3.10. Injectors](#)

[2.3.11. Auto-Rooters](#)

[2.3.12. Kits \(Virus Generators\)](#)

[2.3.13. Spammer Programs](#)

### [2.3.14. Flooders](#)

---

### [2.3.15. Keyloggers](#)

### [2.3.16. Rootkits](#)

## [2.4. Other Categories](#)

### [2.4.1. Joke Programs](#)

### [2.4.2. Hoaxes: Chain Letters](#)

### [2.4.3. Other Pests: Adware and Spyware](#)

## [2.5. Computer Malware Naming Scheme](#)

### [2.5.1. <family\\_name>](#)

### [2.5.2. <malware\\_type>://](#)

### [2.5.3. <platform>/](#)

### [2.5.4. .<group\\_name>](#)

### [2.5.5. <infective\\_length>](#)

### [2.5.6. <variant>](#)

### [2.5.7. \[<devolution>\]](#)

### [2.5.8. <modifiers>](#)

### [2.5.9. .:<locale\\_specifier>](#)

### [2.5.10. #<packer>](#)

### [2.5.11. @m or @mm](#)

### [2.5.12. !<vendor-specific\\_comment>](#)

## [2.6. Annotated List of Officially Recognized Platform Names](#)

## [References](#)

## **[3. Malicious Code Environments](#)**

### [3.1. Computer Architecture Dependency](#)

### [3.2. CPU Dependency](#)

### [3.3. Operating System Dependency](#)

### [3.4. Operating System Version Dependency](#)

### [3.5. File System Dependency](#)

#### [3.5.1. Cluster Viruses](#)

#### [3.5.2. NTFS Stream Viruses](#)

#### [3.5.3. NTFS Compression Viruses](#)

#### [3.5.4. ISO Image Infection](#)

## [3.6. File Format Dependency](#)

---

### [3.6.1. COM Viruses on DOS](#)

### [3.6.2. EXE Viruses on DOS](#)

### [3.6.3. NE \(New Executable\) Viruses on 16-bit Windows and OS/2](#)

### [3.6.4. LX Viruses on OS/2](#)

### [3.6.5. PE \(Portable Executable\) Viruses on 32-bit Windows](#)

### [3.6.6. ELF \(Executable and Linking Format\) Viruses on UNIX](#)

### [3.6.7. Device Driver Viruses](#)

### [3.6.8. Object Code and LIB Viruses](#)

## [3.7. Interpreted Environment Dependency](#)

### [3.7.1. Macro Viruses in Microsoft Products](#)

### [3.7.2. REXX Viruses on IBM Systems](#)

### [3.7.3. DCL \(DEC Command Language\) Viruses on DEC/VMS](#)

### [3.7.4. Shell Scripts on UNIX \(csh, ksh, and bash\)](#)

### [3.7.5. VBScript \(Visual Basic Script\) Viruses on Windows Systems](#)

### [3.7.6. BATCH Viruses](#)

### [3.7.7. Instant Messaging Viruses in mIRC, PIRCH scripts](#)

### [3.7.8. SuperLogo Viruses](#)

### [3.7.9. JScript Viruses](#)

### [3.7.10. Perl Viruses](#)

### [3.7.11. WebTV Worms in JellyScript Embedded in HTML Mail](#)

### [3.7.12. Python Viruses](#)

### [3.7.13. VIM Viruses](#)

### [3.7.14. EMACS Viruses](#)

### [3.7.15. TCL Viruses](#)

### [3.7.16. PHP Viruses](#)

### [3.7.17. MapInfo Viruses](#)

### [3.7.18. ABAP Viruses on SAP](#)

### [3.7.19. Help File Viruses on Windows—When You Press F1](#)

### [3.7.20. JScript Threats in Adobe PDF](#)

### [3.7.21. AppleScript Dependency](#)

### [3.7.22. ANSI Dependency](#)

### [3.7.23. Macromedia Flash ActionScript Threats](#)

### [3.7.24. HyperTalk Script Threats](#)

---

### [3.7.25. AutoLisp Script Viruses](#)

### [3.7.26. Registry Dependency](#)

### [3.7.27. PIF and LNK Dependency](#)

### [3.7.28. Lotus Word Pro Macro Viruses](#)

### [3.7.29. AmiPro Document Viruses](#)

### [3.7.30. Corel Script Viruses](#)

### [3.7.31. Lotus 1-2-3 Macro Dependency](#)

### [3.7.32. Windows Installation Script Dependency](#)

### [3.7.33. AUTORUN.INF and Windows INI File Dependency](#)

### [3.7.34. HTML \(Hypertext Markup Language\) Dependency](#)

## [3.8. Vulnerability Dependency](#)

## [3.9. Date and Time Dependency](#)

## [3.10. JIT Dependency: Microsoft .NET Viruses](#)

## [3.11. Archive Format Dependency](#)

## [3.12. File Format Dependency Based on Extension](#)

## [3.13. Network Protocol Dependency](#)

## [3.14. Source Code Dependency](#)

### [3.14.1. Source Code Trojans](#)

## [3.15. Resource Dependency on Mac and Palm Platforms](#)

## [3.16. Host Size Dependency](#)

## [3.17. Debugger Dependency](#)

### [3.17.1. Intended Threats that Rely on a Debugger](#)

## [3.18. Compiler and Linker Dependency](#)

## [3.19. Device Translator Layer Dependency](#)

## [3.20. Embedded Object Insertion Dependency](#)

## [3.21. Self-Contained Environment Dependency](#)

## [3.22. Multipartite Viruses](#)

## [3.23. Conclusion](#)

## [References](#)

# **[4. Classification of Infection Strategies](#)**

## [4.1. Boot Viruses](#)

[4.1.1. Master Boot Record \(MBR\) Infection Techniques](#)

---

[4.1.2. DOS BOOT Record \(DBR\) Infection Techniques](#)

[4.1.3. Boot Viruses That Work While Windows 95 Is Active](#)

[4.1.4. Possible Boot Image Attacks in Network Environments](#)

[4.2. File Infection Techniques](#)

[4.2.1. Overwriting Viruses](#)

[4.2.2. Random Overwriting Viruses](#)

[4.2.3. Appending Viruses](#)

[4.2.4. Prepending Viruses](#)

[4.2.5. Classic Parasitic Viruses](#)

[4.2.6. Cavity Viruses](#)

[4.2.7. Fractionated Cavity Viruses](#)

[4.2.8. Compressing Viruses](#)

[4.2.9. Amoeba Infection Technique](#)

[4.2.10. Embedded Decryptor Technique](#)

[4.2.11. Embedded Decryptor and Virus Body Technique](#)

[4.2.12. Obfuscated Tricky Jump Technique](#)

[4.2.13. Entry-Point Obscuring \(EPO\) Viruses](#)

[4.2.14. Possible Future Infection Techniques: Code Builders](#)

[4.3. An In-Depth Look at Win32 Viruses](#)

[4.3.1. The Win32 API and Platforms That Support It](#)

[4.3.2. Infection Techniques on 32-Bit Windows](#)

[4.3.3. Win32 and Win64 Viruses: Designed for Microsoft Windows?](#)

[4.4. Conclusion](#)

[References](#)

## **[5. Classification of In-Memory Strategies](#)**

[5.1. Direct-Action Viruses](#)

[5.2. Memory-Resident Viruses](#)

[5.2.1. Interrupt Handling and Hooking](#)

[5.2.2. Hook Routines on INT 13h \(Boot Viruses\)](#)

[5.2.3. Hook Routines on INT 21h \(File Viruses\)](#)

[5.2.4. Common Memory Installation Techniques Under DOS](#)

[5.2.5. Stealth Viruses](#)

## 5.2.6. Disk Cache and System Buffer Infection

---

### 5.3. Temporary Memory-Resident Viruses

### 5.4. Swapping Viruses

### 5.5. Viruses in Processes (in User Mode)

### 5.6. Viruses in Kernel Mode (Windows 9x/Me)

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

### 5.8. In-Memory Injectors over Networks

### References

## **6. Basic Self-Protection Strategies**

### 6.1. Tunneling Viruses

#### 6.1.1. Memory Scanning for Original Handler

#### 6.1.2. Tracing with Debug Interfaces

#### 6.1.3. Code Emulation-Based Tunneling

#### 6.1.4. Accessing the Disk Using Port I/O

#### 6.1.5. Using Undocumented Functions

### 6.2. Armored Viruses

#### 6.2.1. Antidisassembly

#### 6.2.2. Encrypted Data

#### 6.2.3. Code Confusion to Avoid Analysis

#### 6.2.4. Opcode Mixing-Based Code Confusion

#### 6.2.5. Using Checksum

#### 6.2.6. Compressed, Obfuscated Code

#### 6.2.7. Antidebugging

#### 6.2.8. Antiheuristics

#### 6.2.9. Antiemulation Techniques

#### 6.2.10. Antigoat Viruses

### 6.3. Aggressive Retroviruses

### References

## **7. Advanced Code Evolution Techniques and Computer Virus Generator Kits**

### 7.1. Introduction

### 7.2. Evolution of Code

### 7.3. Encrypted Viruses



## [7.4. Oligomorphic Viruses](#)

---

### [7.5. Polymorphic Viruses](#)

#### [7.5.1. The 1260 Virus](#)

#### [7.5.2. The Dark Avenger Mutation Engine \(MtE\)](#)

#### [7.5.3. 32-Bit Polymorphic Viruses](#)

### [7.6. Metamorphic Viruses](#)

#### [7.6.1. What Is a Metamorphic Virus?](#)

#### [7.6.2. Simple Metamorphic Viruses](#)

#### [7.6.3. More Complex Metamorphic Viruses and Permutation Techniques](#)

#### [7.6.4. Mutating Other Applications: The Ultimate Virus Generator?](#)

#### [7.6.5. Advanced Metamorphic Viruses: Zmist](#)

#### [7.6.6. { W32, Linux } /Simile: A Metamorphic Engine Across Systems](#)

#### [7.6.7. The Dark Future—MSIL Metamorphic Viruses](#)

### [7.7. Virus Construction Kits](#)

#### [7.7.1. VCS \(Virus Construction Set\)](#)

#### [7.7.2. GenVir](#)

#### [7.7.3. VCL \(Virus Creation Laboratory\)](#)

#### [7.7.4. PS-MPC \(Phalcon-Skism Mass-Produced Code Generator\)](#)

#### [7.7.5. NGVCK \(Next Generation Virus Creation Kit\)](#)

#### [7.7.6. Other Kits and Mutators](#)

#### [7.7.7. How to Test a Virus Construction Tool?](#)

### [References](#)

## **[8. Classification According to Payload](#)**

### [8.1. No-Payload](#)

### [8.2. Accidentally Destructive Payload](#)

### [8.3. Nondestructive Payload](#)

### [8.4. Somewhat Destructive Payload](#)

### [8.5. Highly Destructive Payload](#)

#### [8.5.1. Viruses That Overwrite Data](#)

#### [8.5.2. Data Diddlers](#)

#### [8.5.3. Viruses That Encrypt Data: The “Good,” the Bad, and the Ugly](#)

#### [8.5.4. Hardware Destroyers](#)

### [8.6. DoS \(Denial of Service\) Attacks](#)

## 8.7. Data Stealers: Making Money with Viruses

---

### 8.7.1. Phishing Attacks

### 8.7.2. Backdoor Features

## 8.8. Conclusion

## References

## **9. Strategies of Computer Worms**

### 9.1. Introduction

### 9.2. The Generic Structure of Computer Worms

#### 9.2.1. Target Locator

#### 9.2.2. Infection Propagator

#### 9.2.3. Remote Control and Update Interface

#### 9.2.4. Life-Cycle Manager

#### 9.2.5. Payload

#### 9.2.6. Self-Tracking

### 9.3. Target Locator

#### 9.3.1. E-Mail Address Harvesting

#### 9.3.2. Network Share Enumeration Attacks

#### 9.3.3. Network Scanning and Target Fingerprinting

### 9.4. Infection Propagators

#### 9.4.1. Attacking Backdoor-Compromised Systems

#### 9.4.2. Peer-to-Peer Network Attacks

#### 9.4.3. Instant Messaging Attacks

#### 9.4.4. E-Mail Worm Attacks and Deception Techniques

#### 9.4.5. E-Mail Attachment Inserters

#### 9.4.6. SMTP Proxy-Based Attacks

#### 9.4.7. SMTP Attacks

#### 9.4.8. SMTP Propagation on Steroids Using MX Queries

#### 9.4.9. NNTP (Network News Transfer Protocol) Attacks

### 9.5. Common Worm Code Transfer and Execution Techniques

#### 9.5.1. Executable Code-Based Attacks

#### 9.5.2. Links to Web Sites or Web Proxies

#### 9.5.3. HTML-Based Mail

[9.5.4. Remote Login-Based Attacks](#)

---

[9.5.5. Code Injection Attacks](#)

[9.5.6. Shell Code-Based Attacks](#)

[9.6. Update Strategies of Computer Worms](#)

[9.6.1. Authenticated Updates on the Web or Newsgroups](#)

[9.6.2. Backdoor-Based Updates](#)

[9.7. Remote Control via Signaling](#)

[9.7.1. Peer-to-Peer Network Control](#)

[9.8. Intentional and Accidental Interactions](#)

[9.8.1. Cooperation](#)

[9.8.2. Competition](#)

[9.8.3. The Future: A Simple Worm Communication Protocol?](#)

[9.9. Wireless Mobile Worms](#)

[References](#)

## **[10. Exploits, Vulnerabilities, and Buffer Overflow Attacks](#)**

[10.1. Introduction](#)

[10.1.1. Definition of Blended Attack](#)

[10.1.2. The Threat](#)

[10.2. Background](#)

[10.3. Types of Vulnerabilities](#)

[10.3.1. Buffer Overflows](#)

[10.3.2. First-Generation Attacks](#)

[10.3.3. Second-Generation Attacks](#)

[10.3.4. Third-Generation Attacks](#)

[10.4. Current and Previous Threats](#)

[10.4.1. The Morris Internet Worm, 1988 \(Stack Overflow to Run Shellcode\)](#)

[10.4.2. Linux/ADM, 1998 \(“Copycatting” the Morris Worm\)](#)

[10.4.3. The CodeRed Outbreak, 2001 \(The Code Injection Attack\)](#)

[10.4.4. Linux/Slapper Worm, 2002 \(A Heap Overflow Example\)](#)

[10.4.5. W32/Slammer Worm, January 2003 \(The Mini Worm\)](#)

[10.4.6. Blaster Worm, August 2003 \(Shellcode-Based Attack on Win32\)](#)

[10.4.7. Generic Buffer Overflow Usage in Computer Viruses](#)

[10.4.8. Description of W32/Badtrans.B@mm](#)

[10.4.9. Exploits in W32/Nimda.A@mm](#)

---

[10.4.10. Description of W32/Bolzano](#)

[10.4.11. Description of VBS/Bubbleboy](#)

[10.4.12. Description of W32/Blebla](#)

[10.5. Summary](#)

[References](#)

## **Part II. Strategies of the Defender**

### **11. Antivirus Defense Techniques**

[11.1. First-Generation Scanners](#)

[11.1.1. String Scanning](#)

[11.1.2. Wildcards](#)

[11.1.3. Mismatches](#)

[11.1.4. Generic Detection](#)

[11.1.5. Hashing](#)

[11.1.6. Bookmarks](#)

[11.1.7. Top-and-Tail Scanning](#)

[11.1.8. Entry-Point and Fixed-Point Scanning](#)

[11.1.9. Hyperfast Disk Access](#)

[11.2. Second-Generation Scanners](#)

[11.2.1. Smart Scanning](#)

[11.2.2. Skeleton Detection](#)

[11.2.3. Nearly Exact Identification](#)

[11.2.4. Exact Identification](#)

[11.3. Algorithmic Scanning Methods](#)

[11.3.1. Filtering](#)

[11.3.2. Static Decryptor Detection](#)

[11.3.3. The X-RAY Method](#)

[11.4. Code Emulation](#)

[11.4.1. Encrypted and Polymorphic Virus Detection Using Emulation](#)

[11.4.2. Dynamic Decryptor Detection](#)

[11.5. Metamorphic Virus Detection Examples](#)

[11.5.1. Geometric Detection](#)

## [11.5.2. Disassembling Techniques](#)

---

### [11.5.3. Using Emulators for Tracing](#)

## [11.6. Heuristic Analysis of 32-Bit Windows Viruses](#)

### [11.6.1. Code Execution Starts in the Last Section](#)

### [11.6.2. Suspicious Section Characteristics](#)

### [11.6.3. Virtual Size Is Incorrect in PE Header](#)

### [11.6.4. Possible “Gap” Between Sections](#)

### [11.6.5. Suspicious Code Redirection](#)

### [11.6.6. Suspicious Code Section Name](#)

### [11.6.7. Possible Header Infection](#)

### [11.6.8. Suspicious Imports from KERNEL32.DLL by Ordinal](#)

### [11.6.9. Import Address Table Is Patched](#)

### [11.6.10. Multiple PE Headers](#)

### [11.6.11. Multiple Windows Headers and Suspicious KERNEL32.DLL Imports](#)

### [11.6.12. Suspicious Relocations](#)

### [11.6.13. Kernel Look-Up](#)

### [11.6.14. Kernel Inconsistency](#)

### [11.6.15. Loading a Section into the VMM Address Space](#)

### [11.6.16. Incorrect Size of Code in Header](#)

### [11.6.17. Examples of Suspicious Flag Combinations](#)

## [11.7. Heuristic Analysis Using Neural Networks](#)

## [11.8. Regular and Generic Disinfection Methods](#)

### [11.8.1. Standard Disinfection](#)

### [11.8.2. Generic Decryptors](#)

### [11.8.3. How Does a Generic Disinfector Work?](#)

### [11.8.4. How Can the Disinfector Be Sure That the File Is Infected?](#)

### [11.8.5. Where Is the Original End of the Host File?](#)

### [11.8.6. How Many Virus Types Can We Handle This Way?](#)

### [11.8.7. Examples of Heuristics for Generic Repair](#)

### [11.8.8. Generic Disinfection Examples](#)

## [11.9. Inoculation](#)

## [11.10. Access Control Systems](#)

## [11.11. Integrity Checking](#)

[11.11.1. False Positives](#)

---

[11.11.2. Clean Initial State](#)

[11.11.3. Speed](#)

[11.11.4. Special Objects](#)

[11.11.5. Necessity of Changed Objects](#)

[11.11.6. Possible Solutions](#)

[11.12. Behavior Blocking](#)

[11.13. Sand-Boxing](#)

[11.14. Conclusion](#)

[References](#)

## **[12. Memory Scanning and Disinfection](#)**

[12.1. Introduction](#)

[12.2. The Windows NT Virtual Memory System](#)

[12.3. Virtual Address Spaces](#)

[12.4. Memory Scanning in User Mode](#)

[12.4.1. The Secrets of NtQuerySystemInformation\(\)](#)

[12.4.2. Common Processes and Special System Rights](#)

[12.4.3. Viruses in the Win32 Subsystem](#)

[12.4.4. Win32 Viruses That Allocate Private Pages](#)

[12.4.5. Native Windows NT Service Viruses](#)

[12.4.6. Win32 Viruses That Use a Hidden Window Procedure](#)

[12.4.7. Win32 Viruses That Are Part of the Executed Image Itself](#)

[12.5. Memory Scanning and Paging](#)

[12.5.1. Enumerating Processes and Scanning File Images](#)

[12.6. Memory Disinfection](#)

[12.6.1. Terminating a Particular Process That Contains Virus Code](#)

[12.6.2. Detecting and Terminating Virus Threads](#)

[12.6.3. Patching the Virus Code in the Active Pages](#)

[12.6.4. How to Disinfect Loaded DLLs and Running Applications](#)

[12.7. Memory Scanning in Kernel Mode](#)

[12.7.1. Scanning the User Address Space of Processes](#)

[12.7.2. Determining NT Service API Entry Points](#)

[12.7.3. Important NT Functions for Kernel-Mode Memory Scanning](#)

---

[12.7.4. Process Context](#)

[12.7.5. Scanning the Upper 2GB of Address Space](#)

[12.7.6. How Can You Deactivate a Filter Driver Virus?](#)

[12.7.7. Dealing with Read-Only Kernel Memory](#)

[12.7.8. Kernel-Mode Memory Scanning on 64-Bit Platforms](#)

[12.8. Possible Attacks Against Memory Scanning](#)

[12.9. Conclusion and Future Work](#)

[References](#)

## **[13. Worm-Blocking Techniques and Host-Based Intrusion Prevention](#)**

[13.1. Introduction](#)

[13.1.1. Script Blocking and SMTP Worm Blocking](#)

[13.1.2. New Attacks to Block: CodeRed, Slammer](#)

[13.2. Techniques to Block Buffer Overflow Attacks](#)

[13.2.1. Code Reviews](#)

[13.2.2. Compiler-Level Solutions](#)

[13.2.3. Operating System-Level Solutions and Run-Time Extensions](#)

[13.2.4. Subsystem Extensions—Libsafe](#)

[13.2.5. Kernel Mode Extensions](#)

[13.2.6. Program Shepherding](#)

[13.3. Worm-Blocking Techniques](#)

[13.3.1. Injected Code Detection](#)

[13.3.2. Send Blocking: An Example of Blocking Self-Sending Code](#)

[13.3.3. Exception Handler Validation](#)

[13.3.4. Other Return-to-LIBC Attack Mitigation Techniques](#)

[13.3.5. “GOT” and “IAT” Page Attributes](#)

[13.3.6. High Number of Connections and Connection Errors](#)

[13.4. Possible Future Worm Attacks](#)

[13.4.1. A Possible Increase of Retroworms](#)

[13.4.2. “Slow” Worms Below the Radar](#)

[13.4.3. Polymorphic and Metamorphic Worms](#)

[13.4.4. Largescale Damage](#)

[13.4.5. Automated Exploit Discovery—Learning from the Environment](#)

## 13.5. Conclusion

---

### References

## **14. Network-Level Defense Strategies**

### 14.1. Introduction

### 14.2. Using Router Access Lists

### 14.3. Firewall Protection

### 14.4. Network-Intrusion Detection Systems

### 14.5. Honeypot Systems

### 14.6. Counterattacks

### 14.7. Early Warning Systems

### 14.8. Worm Behavior Patterns on the Network

#### 14.8.1. Capturing the Blaster Worm

#### 14.8.2. Capturing the Linux/Slapper Worm

#### 14.8.3. Capturing the W32/Sasser.D Worm

#### 14.8.4. Capturing the Ping Requests of the W32/Welchia Worm

#### 14.8.5. Detecting W32/Slammer and Related Exploits

### 14.9. Conclusion

### References

## **15. Malicious Code Analysis Techniques**

### 15.1. Your Personal Virus Analysis Laboratory

#### 15.1.1. How to Get the Software?

### 15.2. Information, Information, Information

#### 15.2.1. Architecture Guides

#### 15.2.2. Knowledge Base

### 15.3. Dedicated Virus Analysis on VMWARE

### 15.4. The Process of Computer Virus Analysis

#### 15.4.1. Preparation

#### 15.4.2. Unpacking

#### 15.4.3. Disassembling and Decryption

#### 15.4.4. Dynamic Analysis Techniques

### 15.5. Maintaining a Malicious Code Collection

### 15.6. Automated Analysis: The Digital Immune System



- [download The Perils of Pleasure \(Pennyroyal Green, Book 1\)](#)
- [read Dark Apprentice \(Star Wars: The Jedi Academy Trilogy, Book 2\)](#)
- [click After the Honeymoon: How Conflict Can Improve Your Relationship \(Revised Edition\) pdf, azw \(kindle\), epub, doc, mobi](#)
- [read DOM Enlightenment book](#)
- [read online Dimiter.pdf](#)
  
- <http://pittiger.com/lib/Programming-Role-Playing-Games-with-DirectX--The-Premier-Press-Game-Development-Series-.pdf>
- <http://nexson.arzamaszev.com/library/Dark-Apprentice--Star-Wars--The-Jedi-Academy-Trilogy--Book-2-.pdf>
- <http://test.markblaustein.com/library/After-the-Honeymoon--How-Conflict-Can-Improve-Your-Relationship--Revised-Edition-.pdf>
- <http://conexdx.com/library/Biocentrism--How-Life-and-Consciousness-Are-the-Keys-to-Understanding-the-True-Nature-of-the-Universe.pdf>
- <http://ramazotti.ru/library/Mistress.pdf>