

4 FREE BOOKLETS
YOUR SOLUTIONS MEMBERSHIP



Buffer Overflow Attacks

DETECT, EXPLOIT, PREVENT

Will the Code You Write Today Headline Tomorrow's BugTraq Mail List?

- Includes Numbered-by-Line Exploit Code Examples that Illustrate the Differences Between Stack Overflows, Heap Corruption, and Format String Bugs
- Provides Case Studies for Most Major Platforms and Environments, Including Windows, FreeBSD, FrontPage, and Linux
- Avoid Worm or Custom Exploits by Analyzing your Source Code to Detect Buffer Overflow Vulnerabilities

James C. Foster

Vitaly Osipov

Nish Bhalla

Niels Heinen

**FOREWORD
BY DAVE AITEL**
FOUNDER AND CEO
IMMUNITY, INC.

Register for Free Membership to

s o l u t i o n s @ s y n g r e s s . c o m

Over the last few years, Syngress has published many best-selling and critically acclaimed books, including Tom Shinder's *Configuring ISA Server 2000*, Brian Caswell and Jay Beale's *Snort 2.0 Intrusion Detection*, and Angela Orebaugh and Gilbert Ramirez's *Ethereal Packet Sniffing*. One of the reasons for the success of these books has been our unique **solutions@syngress.com** program. Through this site, we've been able to provide readers a real time extension to the printed book.

As a registered owner of this book, you will qualify for free access to our members-only solutions@syngress.com program. Once you have registered, you will enjoy several benefits, including:

- Four downloadable e-booklets on topics related to the book. Each booklet is approximately 20-30 pages in Adobe PDF format. They have been selected by our editors from other best-selling Syngress books as providing topic coverage that is directly related to the coverage in this book.
- A comprehensive FAQ page that consolidates all of the key points of this book into an easy to search web page, providing you with the concise, easy to access data you need to perform your job.
- A "From the Author" Forum that allows the authors of this book to post timely updates links to related sites, or additional topic coverage that may have been requested by readers.

Just visit us at **www.syngress.com/solutions** and follow the simple registration process. You will need to have this book with you when you register.

Thank you for giving us the opportunity to serve your needs. And be sure to let us know if there is anything else we can do to make your job easier.

S Y N G R E S S[®]

Application Defense

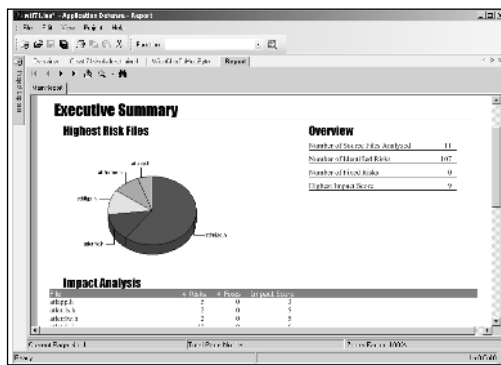
www.applicationdefense.com

Application Defense Specials

- Free Software with Purchase of Application Security Services Program
- \$1,000 Enterprise Language Special Until February 2005 with Proof of Purchase of Ultimate DeskRef.

Business Benefits

- Application Defense Developer Edition, strives to educate individual developers on proper secure programming techniques during the development cycle, thereby saving thousands in post-development consulting
- Developmental education approach on secure development strengthens your business at the core, its people
- Executive-level reporting allows your development team to visually depict trending improvements, vulnerability remediation, and high-risk segments of code
- Distributed Software Architecture permits development teams to review their code centrally by a QA or Auditing team or individually by the developers
- Industry-best multi-language support permits organizations to manage all their software development needs with one application



WideCharToMultiByte

Summary

This function translates a wide-character string to a multi-byte character string.

Description

The function attempts to translate a source wide-character string into a multi-byte character string. The function has eight input arguments: the code page to use for the conversion, the target buffer, the number of bytes available in the target buffer, the destination code page, a string for untranslatable characters, and a flag for unspecified code. The function will return an integer which can have several meanings. Depending on the size of a string, the flag setting, and whether the function was successful, it can return many different values. See the reference for more information.

Line	Function	File	Module
107	WideCharToMultiByte	C:\WINDOWS\system32\USER32.dll	USER32.dll
839	WideCharToMultiByte	C:\WINDOWS\system32\USER32.dll	USER32.dll
921	WideCharToMultiByte	C:\WINDOWS\system32\USER32.dll	USER32.dll
1043	WideCharToMultiByte	C:\WINDOWS\system32\USER32.dll	USER32.dll
1070	WideCharToMultiByte	C:\WINDOWS\system32\USER32.dll	USER32.dll
1085	WideCharToMultiByte	C:\WINDOWS\system32\USER32.dll	USER32.dll
1098	WideCharToMultiByte	C:\WINDOWS\system32\USER32.dll	USER32.dll
1102	WideCharToMultiByte	C:\WINDOWS\system32\USER32.dll	USER32.dll
1102	WideCharToMultiByte	C:\WINDOWS\system32\USER32.dll	USER32.dll
1102	WideCharToMultiByte	C:\WINDOWS\system32\USER32.dll	USER32.dll

Risk in Progress

Progress: 100%

Cancel

Risks

Line	Function	File	Module
107	WideCharToMultiByte	C:\WINDOWS\system32\USER32.dll	USER32.dll
839	WideCharToMultiByte	C:\WINDOWS\system32\USER32.dll	USER32.dll
921	WideCharToMultiByte	C:\WINDOWS\system32\USER32.dll	USER32.dll
1043	WideCharToMultiByte	C:\WINDOWS\system32\USER32.dll	USER32.dll
1070	WideCharToMultiByte	C:\WINDOWS\system32\USER32.dll	USER32.dll
1085	WideCharToMultiByte	C:\WINDOWS\system32\USER32.dll	USER32.dll
1098	WideCharToMultiByte	C:\WINDOWS\system32\USER32.dll	USER32.dll
1102	WideCharToMultiByte	C:\WINDOWS\system32\USER32.dll	USER32.dll
1102	WideCharToMultiByte	C:\WINDOWS\system32\USER32.dll	USER32.dll
1102	WideCharToMultiByte	C:\WINDOWS\system32\USER32.dll	USER32.dll

Project Overview

File: C:\Program Files\Microsoft Office\Office11\WINWORD.EXE

GUID: 00000000-0000-0000-0000-000000000000

Installed: 10/11/2004

File	Detected	# Vulnerable	# Ignored	# Fixed
WINWORD.EXE	107	0	0	0
WINWORD.EXE	839	0	0	0
WINWORD.EXE	921	0	0	0
WINWORD.EXE	1043	0	0	0
WINWORD.EXE	1070	0	0	0
WINWORD.EXE	1085	0	0	0
WINWORD.EXE	1098	0	0	0
WINWORD.EXE	1102	0	0	0
WINWORD.EXE	1102	0	0	0
WINWORD.EXE	1102	0	0	0

Application Defense Technology Features:

- Industry leading analysis engine can parse and examine entire software code base in under a minute
- Executive, technical, and trending reports allow information to be displayed for all audiences
- Flexible XML output allows easy integration with other enterprise applications
- Unique IDE allows you to update results in real-time or in batches to code base – no need to recreate code in multiple locations!
- Custom developer code is analyzed by proprietary artificial intelligence engine
- Project file storage allows developers to save analysis results for later review or to save for continued analysis
- Real-time bug tracking system
- Interactive software interface allows developers to make security decisions during analysis
- Able to input Visual Studio Project files
- Customizable reports allow you to specify company name, application, auditor, and more...

Buffer Overflow Attacks

DETECT, EXPLOIT, PREVENT

James C. Foster
Vitaly Osipov
Nish Bhalla
Niels Heinen

FOREWORD
BY DAVE AITEL
FOUNDER AND CEO
IMMUNITY, INC.

Syngress Publishing, Inc., the author(s), and any person or firm involved in the writing, editing, or production (collectively “Makers”) of this book (“the Work”) do not guarantee or warrant the results to be obtained from the Work.

There is no guarantee of any kind, expressed or implied, regarding the Work or its contents. The Work is sold AS IS and WITHOUT WARRANTY. You may have other legal rights, which vary from state to state.

In no event will Makers be liable to you for damages, including any loss of profits, lost savings, or other incidental or consequential damages arising out from the Work or its contents. Because some states do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

You should always use reasonable care, including backup and other appropriate precautions, when working with computers, networks, data, and files.

Syngress Media®, Syngress®, “Career Advancement Through Skill Enhancement®,” “Ask the Author UPDATE®,” and “Hack Proofing®,” are registered trademarks of Syngress Publishing, Inc. “Syngress: The Definition of a Serious Security Library”™, “Mission Critical™,” and “The Only Way to Stop a Hacker is to Think Like One™” are trademarks of Syngress Publishing, Inc. Brands and product names mentioned in this book are trademarks or service marks of their respective companies.

KEY	SERIAL NUMBER
001	HJIRTCV764
002	PO9873D5FG
003	829KM8NJH2
004	HJBC43288N
005	CVPLQ6WQ23
006	VBP965T5T5
007	HJJJ863WD3E
008	2987GVTWMK
009	629MP5SDJT
010	IMWQ295T6T

PUBLISHED BY

Syngress Publishing, Inc.
800 Hingham Street
Rockland, MA 02370

Buffer Overflow Attacks: Detect, Exploit, Prevent

Copyright © 2005 by Syngress Publishing, Inc. All rights reserved. Printed in the United States of America. Except as permitted under the Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher, with the exception that the program listings may be entered, stored, and executed in a computer system, but they may not be reproduced for publication.

Printed in the United States of America

1 2 3 4 5 6 7 8 9 0

ISBN: **1-932266-67-4**

Publisher: Andrew Williams
Acquisitions Editor: Jaime Quigley
Technical Editor: James C. Foster
Cover Designer: Michael Kavish

Page Layout and Art: Patricia Lupien
Copy Editor: Mike McGee
Indexer: Richard Carlson



Acknowledgments

Syngress would like to acknowledge the following people for their kindness and support in making this book possible.

Syngress books are now distributed in the United States and Canada by O'Reilly Media, Inc. The enthusiasm and work ethic at O'Reilly is incredible and we would like to thank everyone there for their time and efforts to bring Syngress books to market: Tim O'Reilly, Laura Baldwin, Mark Brokering, Mike Leonard, Donna Selenko, Bonnie Sheehan, Cindy Davis, Grant Kikkert, Opol Matsutaro, Steve Hazelwood, Mark Wilson, Rick Brown, Leslie Becker, Jill Lothrop, Tim Hinton, Kyle Hart, Sara Winge, C. J. Rayhill, Peter Pardo, Leslie Crandell, Valerie Dow, Regina Aggio, Pascal Honscher, Preston Paull, Susan Thompson, Bruce Stewart, Laura Schmier, Sue Willing, Mark Jacobsen, Betsy Waliszewski, Dawn Mann, Kathryn Barrett, John Chodacki, Rob Bullington, and Aileen Berg.

The incredibly hard working team at Elsevier Science, including Jonathan Bunkell, Ian Seager, Duncan Enright, David Burton, Rosanna Ramacciotti, Robert Fairbrother, Miguel Sanchez, Klaus Beran, Emma Wyatt, Rosie Moss, Chris Hossack, Mark Hunt, and Krista Leppiko, for making certain that our vision remains worldwide in scope.

David Buckland, Marie Chieng, Lucy Chong, Leslie Lim, Audrey Gan, Pang Ai Hua, and Joseph Chan of STP Distributors for the enthusiasm with which they receive our books.

Kwon Sung June at Acorn Publishing for his support.

David Scott, Tricia Wilden, Marilla Burgess, Annette Scott, Andrew Swaffer, Stephen O'Donoghue, Bec Lowe, and Mark Langley of Woodslane for distributing our books throughout Australia, New Zealand, Papua New Guinea, Fiji Tonga, Solomon Islands, and the Cook Islands.

Winston Lim of Global Publishing for his help and support with distribution of Syngress books in the Philippines.

For the men and woman who proudly
“serve in silence,”
dedicating their lives to
Mission, Workmate, and Country.



Lead Author

James C. Foster, Fellow, is the Deputy Director of Global Security Solution Development for Computer Sciences Corporation. Foster is responsible for directing and managing the vision, technology, and operational design of all security services within CSC. Prior to joining CSC, Foster was the Director of Research and Development for Foundstone Inc., and was responsible for all aspects of product, consulting, and corporate R&D initiatives. Prior to joining Foundstone, Foster was a Senior Advisor and Research Scientist with Guardent Inc., and an editor at Information Security Magazine, subsequent to working as an Information Security and Research Specialist for the Department of Defense. Foster's core competencies include high-tech management, international software development and expansion, web-based application security, cryptography, protocol analysis, and search algorithm technology. Foster has conducted numerous code reviews for commercial OS components, Win32 application assessments, and reviews of commercial and government cryptography implementations.

Foster is a seasoned speaker and has presented throughout North America at conferences, technology forums, security summits, and research symposiums, including the Microsoft Security Summit, BlackHat, MIT Wireless Research Forum, SANS, MilCon, TechGov, InfoSec World 2001, and the Thomson Security Conference. He frequently comments on pertinent security issues and has been cited in USA Today, Information Security Magazine, Baseline, Computerworld, Secure Computing, and the MIT Technologist. Foster holds degrees in Business Administration, Software Engineering, and Management of Information Systems. He has attended the Yale School of Business, Harvard University, and the University of Maryland. He is currently a Fellow at the University of Pennsylvania's Wharton School of Business.

Foster has written many commercial and educational papers. He has also contributed to several books, including: *Snort 2.0*, *Snort 2.1 2nd Edition*, *Hacking Exposed 4th Ed and 5th Edition*, *Special Ops Security*, *Anti-Hacker Toolkit 2nd Ed*, *Advanced Intrusion Detection*, *Hacking the Code*, *Anti-Spam Toolkit*, *Programmer's Ultimate Security DeskRef*, *Google for Penetration Testers*, *Buffer Overflow Attacks*, and *Sockets/Porting/and Shellcode*.



Contributing Authors

Vitaly Osipov (CISSP, CISA) is currently managing intrusion detection systems for a Big 5 global investment bank in Sydney, Australia. He previously worked as a security specialist for several European companies in Dublin, Prague and Moscow. Vitaly has co-authored books on firewalls, IDS, and security including *Special Ops: Host and Network Security for Microsoft, UNIX and Oracle* (ISBN: 1-931836-69-8) and *Snort 2.0: Intrusion Detection* (ISBN: 1-931836-74-4). Vitaly's background includes a long history of designing and implementing information security systems for financial institutions, ISPs, telecoms, and consultancies. He is currently studying for his second postgraduate degree in mathematics.

Niels Heinen is a security researcher at a European security firm. Niels has researched exploitation techniques and specializes in writing position independent assembly code used for changing program execution flows. While the main focus of his research is Intel Systems, he's also experienced with MIPS, HPPA and PIC processors. Niels, enjoys writing his own polymorphic exploits, wardrive scanners and OS fingerprint tools. His day-to-day job involves in-depth analysis of security products.

Nishchal Bhalla is a specialist in product testing, code reviews and web application testing. He is the lead consultant at Security Compass, providing consulting services for major software corporations & Fortune 500 companies. He's a contributing author to *Windows XP Professional Security and Hack Notes*. Prior to joining Security Compass, Nish worked for Foundstone, TD Waterhouse, Axa Group and Lucent. He holds a master's degree in parallel processing from Sheffield University, is a post-graduate in finance from Strathclyde University, and received a bachelor's degree in commerce from Bangalore University.



Additional Area Experts

Marshall Beddoe is a Research Scientist at McAfee, and conducts extensive research in passive network mapping, remote promiscuous detection, OS fingerprinting, FreeBSD internals, and new exploitation techniques. Marshall has spoken at such security conferences as Black Hat, Defcon, and Toorcon.

Tony Bettini leads the McAfee Foundstone R&D team and has worked for other security firms, including Foundstone, Guardent, and Bindview. He specializes in Windows security and vulnerability detection, as well as programs in Assembly, C, and various others. Tony has identified new vulnerabilities in PGP, ISS Scanner, Microsoft Windows XP, and Winamp.



Author's Acknowledgements

Most importantly, I'd like to thank my family for continuously believing in me and my ambitious goals. Mom, Dad, Steve – to you all I am grateful.

I'd like to thank everyone who helped contribute to this book, especially Vitaly (you rock!), Nish, Niels, Marshall, Tony, Dave Aitel, Johnny Long, Conrad Smith, and last but certainly not least, Stuart McClure. You guys' talent will no doubt ensure the lasting success of this book.

Sincere thanks go out to the true professionals at the software security vendors who helped make Chapter 9 a success, specifically Fortify Software's Brian Chess Ph.D, Chris Prevost, and Steve Garrity; Ounce Labs' Robert Gottlieb, John Peyton, Ellen Sinett, and Chris McClean; and Secure Software's John Viega, Dale R. Gardner, and Joel Greenberg.

An additional thank you goes out to Computer Sciences Corporation for allowing this publication to take place. Reg – you are still the man! Additional well-deserved thanks go out to Chris, Jason, Ron, Jen, and Mary.

Last but certainly not least, I'd like to thank the Syngress Publishing team. Jaime, thanks for putting up with "Foster" content and making sure this book stayed on schedule; you're an outstanding editor. Andrew – your vision is what spawned this book so thank you and kudos! Syngress continues to be my publishing company of choice.

*-James Foster
December, 2004*

Contents

Forewordxxi
Part 1 Expanding on Buffer Overflows1
Chapter 1 Buffer Overflows: The Essentials3
Introduction3
The Challenge of Software Security4
Microsoft Software Is Not Bug Free6
The Increase in Buffer Overflows8
Exploits vs. Buffer Overflows10
Madonna Hacked!10
Definitions12
Hardware12
Software13
Security18
Summary20
Solutions Fast Track20
Frequently Asked Questions23
Chapter 2 Understanding Shellcode25
Introduction25
An Overview of Shellcode26
The Tools26
The Assembly Programming Language27
Analysis28
Analysis29
Analysis29
Windows vs. Unix Assembly31
The Addressing Problem32
Using the “call” and “jmp” Trick32
Analysis32
Analysis33
Pushing the Arguments33
The NULL Byte Problem34
Implementing System Calls35
System Call Numbers35
System Call Arguments36
Analysis36
Analysis37

Analysis	.37
System Call Return Values	.38
Remote Shellcode	.38
Port-Binding Shellcode	.38
Analysis	.40
Socket Descriptor Reuse Shellcode	.40
Analysis	.40
Local Shellcode	.41
execve Shellcode	.41
setuid Shellcode	.43
chroot Shellcode	.44
Summary	.49
Solutions Fast Track	.49
Links to Sites	.51
Mailing Lists	.51
Frequently Asked Questions	.52
Chapter 3 Writing Shellcode	.55
Introduction	.55
Shellcode Examples	.56
The Write System Call	.58
Analysis	.60
Analysis	.61
execve Shellcode	.63
Analysis	.63
Analysis	.64
Analysis	.66
Analysis	.68
Analysis	.70
Analysis	.71
Port-Binding Shellcode	.72
Analysis	.73
Analysis	.75
Analysis	.76
Analysis	.76
Analysis	.77
Analysis	.78
Analysis	.81
Reverse Connection Shellcode	.83
Analysis	.85
Socket Reusing Shellcode	.87
Analysis	.88
Analysis	.88
Reusing File Descriptors	.89
Analysis	.89

Analysis	.91
Analysis	.92
Analysis	.93
Analysis	.93
Analysis	.94
Analysis	.95
Encoding Shellcode	.96
Analysis	.97
Analysis	.99
Analysis	.101
Reusing Program Variables	.102
Open Source Programs	.102
Analysis	.103
Closed Source Programs	.104
Analysis	.105
Analysis	.105
OS-Spanning Shellcode	.106
Analysis	.107
Understanding Existing Shellcode	.107
Analysis	.109
Summary	.112
Solutions Fast Track	.112
Links to Sites	.113
Mailing Lists	.114
Frequently Asked Questions	.114
Chapter 4 Win32 Assembly	.117
Introduction	.117
Application Memory Layout	.118
Application Structure	.120
Memory Allocation—Stack	.121
Memory Allocation—Heap	.122
Heap Structure	.123
Windows Assembly	.124
Registers	.124
Indexing Registers	.125
Stack Registers	.125
Other General-Purpose Registers	.125
EIP Register	.126
Data Type	.126
Operations	.126
Hello World	.127
Summary	.129
Solutions Fast Track	.130
Frequently Asked Questions	.131

Section 1 Case Studies

Case Study 1.1 FreeBSD NN Exploit Code	133
Overview	133
Exploitation Code Dump	134
Analysis	136
References	137
Case Study 1.2 xlockmore User Supplied Format String Vulnerability	138
Overview	138
xlockmore Vulnerability Details	139
Exploitation Code Dump	139
Analysis	141
References	141
Case Study 1.3 Frontpage Denial of Service Utilizing WinSock	142
Overview	142
Code Dump	143
Analysis	144
Application Defense Hack.h Code Dump	145
Analysis	151
References	152
Case Study 1.4 cURL buffer overflow on FreeBSD	154
Overview	154
Exploit Code	155
Analysis	157
References	158
Part II Exploiting Buffer Overflows	159
Chapter 5 Stack Overflows	161
Introduction	161
Intel x86 Architecture and Machine Language Basics	163
Registers	164
Stacks and Procedure Calls	165
Storing Local Variables	167
Calling Conventions and Stack Frames	172
Introduction to the Stack Frame	172
Passing Arguments to a Function	173
Stack Frames and Calling Syntaxes	180
Process Memory Layout	181
Stack Overflows and Their Exploitation	183
Simple Overflow	185
Creating an Example Program with an Exploitable Overflow	189
Writing Overflowable Code	189
Disassembling the Overflowable Code	190
Performing the Exploit	192

General Exploit Concepts	.192
Buffer Injection Techniques	.193
Methods to Execute Payload	.194
Designing Payload	.198
What Is an Off-by-One Overflow?	.204
Functions That Can Produce Buffer Overflows	.211
Functions and Their Problems, or Never Use gets()	.211
gets() and fgets()	.211
strcpy() and strncpy(), strcat(), and strncat()	.212
(v)sprintf() and (v)snprintf()	.213
sscanf(), vscanf(), and fscanf()	.213
Other Functions	.214
Challenges in Finding Stack Overflows	.215
Lexical Analysis	.217
Semantics-Aware Analyzers	.218
Application Defense!	.220
OpenBSD 2.8 ftpd Off-by-One	.220
Apache httpd Buffer Overflow	.221
Summary	.222
Solutions Fast Track	.224
Links to Sites	.225
Frequently Asked Questions	.227
Chapter 6 Heap Corruption	.229
Introduction	.229
Simple Heap Corruption	.230
Using the Heap—malloc(), calloc(), realloc()	.231
Simple Heap and BSS Overflows	.232
Corrupting Function Pointers in C++	.235
Advanced Heap Corruption—Doug Lea malloc	.238
Overview of Doug Lea malloc	.238
Memory Organization—Boundary Tags, Bins, Arenas	.239
The free() Algorithm	.244
Fake Chunks	.246
Example Vulnerable Program	.248
Exploiting frontlink()	.250
Off-by-One and Off-by-Five on the Heap	.251
Advanced Heap Corruption—System V malloc	.252
System V malloc Operation	.252
Tree Structure	.253
Freeing Memory	.255
The realloc() Function	.257
The t_delete Function—The Exploitation Point	.260
Application Defense!	.263
Fixing Heap Corruption Vulnerabilities in the Source	.263

Summary	.266
Solutions Fast Track	.267
Links to Sites	.268
Frequently Asked Questions	.270
Chapter 7 Format String Attacks	.273
Introduction	.273
What Is a Format String?	.274
C Functions with Variable Numbers of Arguments	.274
Ellipsis and <code>va_args</code>	.275
Functions of Formatted Output	.278
Using Format Strings	.280
<code>printf()</code> Example	.280
Format Tokens and <code>printf()</code> Arguments	.281
Types of Format Specifiers	.282
Abusing Format Strings	.284
Playing with Bad Format Strings	.286
Denial of Service	.287
Direct Argument Access	.287
Reading Memory	.288
Writing to Memory	.291
Simple Writes to Memory	.291
Multiple Writes	.294
Challenges in Exploiting Format String Bugs	.296
Finding Format String Bugs	.296
What to Overwrite	.299
Destructors in <code>.dtors</code>	.300
Global Offset Table entries	.302
Structured Exception Handlers	.304
Operating System Differences	.305
Difficulties in Exploiting Different Systems	.308
Application Defense!	.308
The Whitebox and Blackbox Analysis of Applications	.309
Summary	.311
Solutions Fast Track	.311
Links to Sites	.313
Frequently Asked Questions	.314
Chapter 8 Windows Buffer Overflows	.317
Introduction	.317
Background	.318
Basic Stack Overflow	.318
Analysis	.322
Writing Windows Shellcode	.327
Overcoming Special Characters (Example: NULL)	.333
Client Server Application	.338

Using/Abusing the Structured Exception Handler	350
Summary	355
Solutions Fast Track	355
Frequently Asked Questions	357
Section 2 Case Studies	
Case Study 2.1 cURL Buffer Overflow on Linux	359
Overview	359
Exploit Code	360
Analysis	362
References	363
Case Study 2.2 SSLv2 Malformed Client Key Remote Buffer Overflow Vuln.	364
Overview	364
OpenSSL Vulnerability Details	365
Exploitation Details	365
The Complication	366
Analysis	367
Improving the Exploit	368
Much Improved... but More to Come!	368
Complete Exploit Code for OpenSSL SSLv2 Malformed Client Key Remote Buffer Overflow	369
References	375
Case Study 2.3 X11R6 4.2 XLOCALEDIR Overflow	376
Overview	376
XLOCALEDIR Vulnerability Details and Analysis	377
Exploitation Code Dump	379
Analysis	381
References	381
Case Study 2.4 Microsoft MDAC Denial of Service	382
Overview	382
Code Dump	383
Analysis	385
Application Defense Hack.h Code Dump	386
Analysis	392
References	394
Case Study 2.5 Local UUX Buffer Overflow on HPUX	395
Overview	395
Exploit Code	396
Analysis	397
References	399
Part III Finding Buffer Overflows	401
Chapter 9 Finding Buffer Overflows in Source	403
Introduction	403
Source Code Analysis	404

Free Open Source Tools	406
Application Defense Snapshot	406
RATS	408
Flawfinder	412
Flawfinder Text Output	414
ITS4	417
Application Defense—Enterprise Developer	418
Secure Software	423
Architecture and Deployment	423
Vulnerability Knowledgebase	424
Using CodeAssure	425
Setting Up Projects	427
Performing the Analysis	427
Vulnerability Review and Reporting	428
Managing Results	430
Remediation	432
Ounce Labs	432
Prexis’ Science of Automated Analysis	433
Prexis Architecture	434
Prexis Assessment Capability	434
Prexis Reporting and Remediation Capabilities	434
Prexis in Action	435
Vulnerability Assessment with Prexis	436
Project Configuration	436
Running an Assessment	436
Examining Assessment Results	437
Filtering Assessments	438
Remediation	438
Fortify Software	440
Fortify’s Source Code Analysis Suite	441
Using the Source Code Analysis Engine	441
Integrating with the Build Process	442
Running the Analysis	443
Understanding the Raw Output	443
Audit Workbench	444
Audit Guide	445
Software Security Manager	447
Summary	449
Solutions Fast Track	450
Links to Sites	452
Frequently Asked Questions	453

Section 3 Case Studies

Case Study 3.1 InlineEgg I455
Overview455
Inline Egg Code456
Analysis456
References457
Case Study 3.2 InlineEgg II458
Overview458
Inline Egg Code459
Analysis460
References461
Case Study 3.3 Seti@Home Exploit Code462
Overview462
Exploitation Code Dump462
Analysis467
References469
Case Study 3.4 Microsoft CodeBlue Exploit Code470
Overview470
Exploitation Code Dump471
Analysis474
References475
Appendix A The Complete Data Conversion Table477
Appendix B Useful Syscalls485
exit(int)485
open(file, flags, mode)485
close(filedescriptor)485
read(filedescriptor, pointer to buffer, amount of bytes)486
write(filedescriptor, pointer to buffer, amount of bytes)486
execve(file, file + arguments, environment data)486
socketcall(callnumber, arguments)486
socket(domain, type, protocol)487
bind(file descriptor, sockaddr struct, size of arg 2)487
listen (file descriptor, number of connections allowed in queue)	.487
accept (file descriptor, sockaddr struct, size of arg 2)487
Index489



Foreword

I can recall with crystal clarity three times relating to buffer overflows. I remember the first moments of when I used a buffer overflow in the wild and it worked (imapd for Linux). I remember the first buffer overflow that I found and exploited all by myself (a Linux local). And I remember the first time I got into a host by writing a buffer overflow.

When most people read Aleph1's seminal paper on buffer overflows "Buffer overflows for fun and profit," they think largely on the "profit." Someone skilled in the art of writing buffer overflows can make \$90-120K USD a year as a consultant for any of the large companies in the market.

Getting that skill, on the other hand, has one major roadblock: most people think that they can "learn" it. Many knacks in the IT world can be learned. Once you realize them, they're yours to keep. But writing a buffer overflow is not like that. It's more like weight lifting, or judo. You can learn the basics from a book, or perhaps a short class, but the landscape you work in changes constantly. Microsoft is adding new protections to their code every day, as hackers are constantly finding new ways to make exploits more reliable, and finding ways to develop new types of bugs. Three months after you stop writing buffer overflows, your skills are ancient history. The hard part about writing buffer overflows is staring an endless treadmill of work in the face.

The tools you once used to use to write overflows are changing, too. It used to be something you could do alone, with nothing but a cracked copy of Softice or GDB. These days, Immunity invests heavily into infrastructure in order to write even a simple buffer overflow. For example, we have a proprietary debugger all our own, which allows us to query and script running programs. We have special purpose compilers which allow us to create and tune our shellcode to be exactly what the vulnerability needs. We purchase or create reverse engineering tools specialized for various problems. We have complete MySQL and SSL libraries written entirely in Python. A moderately complex exploit will involve our entire team working in unison on various parts of the problem.

And of course, afterwards, there's the inevitable paper on each difficult exploit, which must be written to share the knowledge gained with the difficult exploits within the team. It's no longer an individual sport.

The most powerful of buffer overflows are never going to be in worms. Custom exploits are what gets you in, and what keeps you in. If you see a truly top notch hacker going after someone, he'll have a complete mirror of the target hosts' environments, all for one purpose: To create a buffer overflow he only plans on using once.

There are levels in the game. James Foster will take you through basic training, giving you the broad view and skills you need to figure out what you want to specialize in. From there, you can walk away into management, confident that although you're not on the front line, you at least know how the fundamentals and can thereby make

informed decisions. Or you can dedicate yourself to the work, commit to the code, and develop the craft. If this is your choice, then I have a few maxims for you:

- Never be afraid. Microsoft has an entire army of marketing people telling you how hard it is to find and write buffer overflows against their newest software. My strategy is to fantasize about what I'm going to do with the exploit once it's working – it keeps me going. Writing exploits is going to require mastery of a lot of difficult and boring technology. HP-UX uses wacky quadrant memory access. Irix has inane caches. And learning the thousands of assembly languages this requires is no small feat. If you think you can do it, you can.
- Don't take yourself too seriously. Realize that no matter how good you get, some fifteen year-old in Sweden is dedicating twenty hours a day to be better than you. It's not a competition, and you'll burn out quick if you play the game as such.
- Find some friends. Like weight lifting or judo, this is not a skill you're going to keep up with if you do it all yourself. And you need people to tell you where you're weak.

Regardless of your goals, use this book as a worksheet, rather than a novel. Don't read it without a computer next to you. A book on buffer overflows can't deliver that hacker high with knowledge alone. Work along with the chapters and you'll find that you can't stop fiddling with your exploit when it's not working right. You'll give up food, sleep, or money to make it that much cleaner.

My belief is this: In the end, an exploit is a complex statement of truth. What you're saying is "This is possible." And saying it truthfully makes it beautiful.

Someday, I hope to appreciate your code as art.

—*Dave Aitel*
Founder, CEO
Immunity, Inc.

Part 1

Expanding on Buffer Overflows

- [click Revisiting the Jewish Question for free](#)
- [read *Blowback: Adventures of a Dope Smuggler*](#)
- [download online Moscow Sting \(Finn Series, Book 2\) online](#)
- [read online Quantum Shift in the Global Brain: How the New Scientific Reality Can Change Us and Our World pdf, azw \(kindle\), epub](#)
- [read *Classico e Moderno: Essential Italian Cooking*](#)
- [*It Had to Be You \(Gossip Girl, Book 0.5\) pdf, azw \(kindle\)*](#)

- <http://www.mmastyles.com/books/Revisiting-the-Jewish-Question.pdf>
- <http://crackingscience.org/?library/Blowback--Adventures-of-a-Dope-Smuggler.pdf>
- <http://unpluggedtv.com/lib/Moscow-Sting--Finn-Series--Book-2-.pdf>
- <http://sidenoter.com/?ebooks/Collected-Works--Volume-42--Letters-1864-68.pdf>
- <http://www.gateaerospaceforum.com/?library/Classico-e-Moderno--Essential-Italian-Cooking.pdf>
- <http://reseauplatoparis.com/library/Closing-Techniques--That-Really-Work----4th-Edition-.pdf>