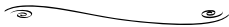


Contents

Understand how rogue applets can transmit bad code:



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.

Foreword	xxv
Chapter 1 Hacking Methodology	1
Introduction	2
Understanding the Terms	3
A Brief History of Hacking	4
Phone System Hacking	5
Computer Hacking	6
What Motivates a Hacker?	9
Ethical Hacking versus Malicious Hacking	10
Working with Security Professionals	11
Associated Risks with Hiring a Security Professional	12
Understanding Current Attack Types	13
DoS/DDoS	13
Virus Hacking	16
Trojan Horses	18
Worms	21
Rogue Applets	22
Stealing	23
Credit Card Theft	24
Theft of Identity	26
Information Piracy	27
Recognizing Web Application Security Threats	28
Hidden Manipulation	29
Parameter Tampering	29
Cross-Site Scripting	29
Buffer Overflow	30
Cookie Poisoning	31
	xiii

Thinking Creatively When Coding

- Be aware of outside influences on your code, expect the unexpected!
- Look for ways to minimize your code; keep the functionality in as small a core as possible.
- Review, review, review! Don't try to isolate your efforts or conceal mistakes.

Preventing Break-Ins by Thinking Like a Hacker	31
Summary	35
Solutions Fast Track	36
Frequently Asked Questions	40

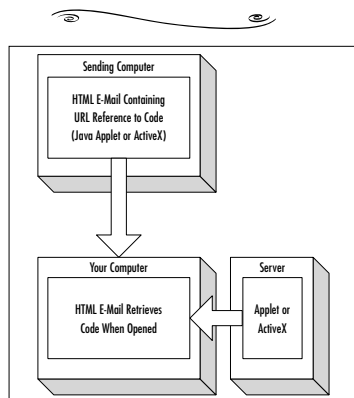
Chapter 2 How to Avoid Becoming a "Code Grinder" 43

Introduction	44
What Is a Code Grinder?	45
Following the Rules	49
Thinking Creatively When Coding	50
Allowing for Thought	53
Modular Programming Done Correctly	53
Security from the Perspective of a Code Grinder	56
Coding in a Vacuum	58
Building Functional and Secure Web Applications	59
But My Code Is Functional!	66
There Is More to an Application than	
Functionality	68
Let's Make It Secure and Functional	71
Summary	76
Solutions Fast Track	77
Frequently Asked Questions	78

Chapter 3 Understanding the Risks Associated with Mobile Code 81

Introduction	82
Recognizing the Impact of Mobile Code Attacks	83
Browser Attacks	83
Mail Client Attacks	84
Malicious Scripts or Macros	85
Identifying Common Forms of Mobile Code	86
Macro Languages: Visual Basic for	
Applications (VBA)	87
Security Problems with VBA	89
Protecting against VBA Viruses	92
JavaScript	93
JavaScript Security Overview	94

Understand how mobile code works for Java applets and ActiveX controls:



Mobile Code Residing on a Web Server

Security Problems	95
Exploiting Plug-In Commands	96
Web-Based E-Mail Attacks	96
Social Engineering	97
Lowering JavaScript Security Risks	97
VBScript	98
VBScript Security Overview	98
VBScript Security Problems	99
VBScript Security Precautions	101
Java Applets	101
Granting Additional Access to Applets	102
Security Problems with Java	103
Java Security Precautions	104
ActiveX Controls	105
ActiveX Security Overview	105
Security Problems with ActiveX	107
E-Mail Attachments and Downloaded Executables	110
Back Orifice 2000 Trojan	111
Protecting Your System from Mobile Code Attacks	115
Security Applications	115
ActiveX Manager	115
Back Orifice Detectors	115
Firewall Software	119
Web-Based Tools	119
Identifying Bad ActiveX Controls	119
Client Security Updates	120
Summary	121
Solutions Fast Track	122
Frequently Asked Questions	123
Chapter 4 Vulnerable CGI Scripts	125
Introduction	126
What Is a CGI Script, and What Does It Do?	127
Typical Uses of CGI Scripts	129
When Should You Use CGI?	135

**Tools & Traps...Beware
of User Input**

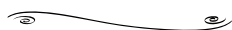
One of the most common methods of exploiting CGI scripts and programs is used when scripts allow user input, but the data that users are submitting is not checked. Controlling what information users are able to submit will reduce your chances of being hacked through a CGI script dramatically.

CGI Script Hosting Issues	136
Break-Ins Resulting from Weak CGI Scripts	137
How to Write “Tighter” CGI Scripts	139
Searchable Index Commands	143
CGI Wrappers	144
Whisker	145
Languages for Writing CGI Scripts	149
Unix Shell	150
Perl	151
C/C++	151
Visual Basic	152
Advantages of Using CGI Scripts	153
Rules for Writing Secure CGI Scripts	153
Storing CGI Scripts	157
Summary	161
Solutions Fast Track	161
Frequently Asked Questions	165

Chapter 5 Hacking Techniques and Tools 167

Introduction	168
A Hacker’s Goals	169
Minimize the Warning Signs	170
Maximize the Access	172
Damage, Damage, Damage	175
Turning the Tables	177
The Five Phases of Hacking	178
Creating an Attack Map	179
Building an Execution Plan	182
Establishing a Point of Entry	183
Continued and Further Access	184
The Attack	186
Social Engineering	188
Sensitive Information	188
E-Mail or Messaging Services	189
Telephones and Documents	191
Credentials	193
The Intentional “Back Door” Attack	195

Answers All Your Questions About Hacking Techniques



Q: What should I do if I stumble across a back door in my code base?

A: First and most importantly, determine that it is a genuine back door. Segments of code often appear to have no authentication aspect and can do some rather powerful things, but nonetheless had proper authentication performed prior to their being called. If your best research still indicates that it is a back door, contact an associate in your security department who understands the language in which you're coding and request a review of the code. If that person determines it is a back door, it should be investigated to determine whether the code was introduced simply due to poor planning or actual malice.

Hard-Coding a Back Door Password	195
Exploiting Inherent Weaknesses in Code or Programming Environments	198
The Tools of the Trade	199
Hex Editors	199
Debuggers	201
Disassemblers	202
Windows-Based Tools	202
Quick View	204
DOS-Based Tools	204
Summary	206
Solutions Fast Track	207
Frequently Asked Questions	211

Chapter 6 Code Auditing and Reverse Engineering 215

Introduction	216
How to Efficiently Trace through a Program	216
Auditing and Reviewing Selected Programming Languages	220
Reviewing Java	220
Reviewing Java Server Pages	221
Reviewing Active Server Pages	221
Reviewing Server Side Includes	222
Reviewing Python	222
Reviewing Tool Command Language	222
Reviewing Practical Extraction and Reporting Language	222
Reviewing PHP: Hypertext Preprocessor	223
Reviewing C/C++	223
Reviewing ColdFusion	224
Looking for Vulnerabilities	224
Getting the Data from the User	225
Looking for Buffer Overflows	226
The str* Family of Functions	227
The strn* Family of Functions	228
The *scanf Family of Functions	228

How to Efficiently Trace through a Program

- ☑ Tracing a program's execution from start to finish is too time-intensive.
- ☑ You can save time by instead going directly to problem areas.
- ☑ This approach allows you to skip benign application processing/calculation logic.

Other Functions Vulnerable to Buffer Overflows	229
Checking the Output Given to the User	230
Format String Vulnerabilities	230
Cross-Site Scripting	232
Information Disclosure	234
Checking for File System Access/Interaction	235
Checking External Program and Code Execution	238
Calling External Programs	239
Dynamic Code Execution	240
External Objects/Libraries	241
Checking Structured Query Language (SQL)/Database Queries	242
Checking Networking and Communication Streams	245
Pulling It All Together	247
Summary	248
Solutions Fast Track	248
Frequently Asked Questions	250

Chapter 7 Securing Your Java Code **253**

Introduction	254
Overview of the Java Security Architecture	255
The Java Security Model	257
The Sandbox	259
Security and Java Applets	260
How Java Handles Security	264
Class Loaders	265
The Applet Class Loader	266
Adding Security to a Custom Class Loader	266
Byte-Code Verifier	269
Java Protected Domains	275
Java Security Manager	276
Policy Files	277
The SecurityManager Class	284

Complete coverage of the Java Security Model:

- Class loaders
- Byte-code verification
- Security managers
- Digital signatures
- Authentication using certificates
- JAR signing
- Encryption

Damage & Defense: Debugging XSL

The interaction of a style sheet with an XML document can be a complicated process, and unfortunately, style sheet errors can often be cryptic. Microsoft has an HTML-based XSL debugger you can use to walk through the execution of your XSL. You can also view the source code to make your own improvements. You can find the XSL Debugger at http://msdn.microsoft.com/downloads/samples/internet/xml/sxl_debugger/default.asp.

Potential Weaknesses in Java	285
DoS Attack/Degradation of Service Attacks	285
Third-Party Trojan Horse Attacks	289
Coding Functional but Secure Java Applets	290
Message Digests	291
Digital Signatures	295
Generating a Key Pair	298
Obtaining and Verifying a Signature	301
Authentication	303
X.509 Certificate Format	305
Obtaining Digital Certificates	305
Protecting Security with JAR Signing	311
Encryption	315
Cryptix Installation Instructions	319
Sun Microsystems Recommendations	
for Java Security	322
Privileged Code Guidelines	323
Java Code Guidelines	324
C Code Guidelines	325
Summary	326
Solutions Fast Track	327
Frequently Asked Questions	329

Chapter 8 Securing XML 331

Introduction	332
Defining XML	332
Logical Structure	334
Elements	335
Attributes	336
Well-Formed Documents	337
Valid Document	337
XML and XSL/DTD Documents	339
XSL Use of Templates	339
XSL Use of Patterns	340
DTD	344
Schemas	345
Creating Web Applications Using XML	347

The Risks Associated with Using XML	352
Confidentiality Concerns	353
Securing XML	354
XML Encryption	355
XML Digital Signatures	362
Summary	366
Solutions Fast Track	367
Frequently Asked Questions	369

Chapter 9 Building Safe ActiveX Internet Controls 371

Use ActiveX and understand the Authenticode Security Warning



Introduction	372
Dangers Associated with Using ActiveX	373
Avoiding Common ActiveX Vulnerabilities	375
Lessening the Impact of ActiveX Vulnerabilities	378
Protection at the Network Level	379
Protection at the Client Level	379
Methodology for Writing Safe ActiveX Controls	382
Object Safety Settings	383
Securing ActiveX Controls	385
Control Signing	385
Using Microsoft Authenticode	387
Control Marking	389
Using Safety Settings	389
Using IObjectSafety	390
Marking the Control in the Windows Registry	395
Summary	397
Solutions Fast Track	398
Frequently Asked Questions	400

Chapter 10 Securing ColdFusion 403

Introduction	404
How Does ColdFusion Work?	404
Utilizing the Benefit of Rapid Development	406

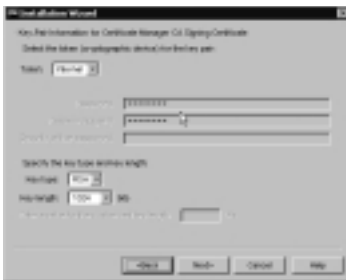
Write Secure ColdFusion Code:

When writing a ColdFusion application, you must look out for a number of tags that involve the movement of data in ways that can be attacked. In most cases, validating the data sent to a page will prevent them from being misused. In others, not allowing attributes to be set dynamically is the answer. For each tag we examine, another solution may be to just turn the tag off (an option controlled by the administration panel). Other tags can not be turned off and must be coded properly.

Understanding ColdFusion Markup Language	408
Scalable Deployment	410
Open Integration	410
Preserving ColdFusion Security	411
Secure Development	414
CFINCLUDE	414
Queries	419
Uploaded Files	425
Denial of Service	425
Turning Off Tags	426
Secure Deployment	427
ColdFusion Application Processing	428
Checking for Existence of Data	428
Checking Data Types	430
Data Evaluation	433
Risks Associated with Using ColdFusion	435
Using Error Handling Programs	438
Monitor.cfm Example	441
Using Per-Session Tracking	444
Summary	447
Solutions Fast Track	448
Frequently Asked Questions	450

Chapter 11 Developing Security-Enabled Applications 451

Select Cryptography Token, Key Type, and Key Length



Introduction	452
The Benefits of Using Security-Enabled Applications	453
Types of Security Used in Applications	454
Digital Signatures	455
Pretty Good Privacy	456
Secure Multipurpose Internet Mail Extension	459
Secure Sockets Layer	460
Server Authentication	462
Client Authentication	462
Digital Certificates	466

	Reviewing the Basics of PKI	468
	Certificate Services	471
	iPlanet by Sun/Netscape	472
	Using PKI to Secure Web Applications	472
	Implementing PKI in Your Web Infrastructure	473
	Microsoft Certificate Services	474
	Netscape Certificate Server	478
	Installation of Netscape Certificate Server	478
	Administering Netscape CMS	483
	PKI for Apache Server	486
	PKI and Secure Software Toolkits	487
	Testing Your Security Implementation	488
	Summary	492
	Solutions Fast Track	493
	Frequently Asked Questions	497
Set up a checklist of defects not easily detected through standard testing methods for working in a Java environment:		
■ Excessive copying of strings—unnecessary copies of immutable objects		
■ Failure to clone returned objects		
■ Unnecessary cloning		
■ Copying arrays by hand		
■ Copying the wrong thing or making only a partial copy		
■ Testing new for null		
■ Using == instead of .equals		
■ The confusion of nonatomic and atomic operations		
■ The addition of unnecessary catchblocks		
■ Failure to implement equals, clone or hashCode		
	Chapter 12 Cradle to Grave: Working with a Security Plan	499
	Introduction	500
	Examining Your Code	501
	Code Reviews	502
	Peer-to-Peer Code Reviews	504
	Being Aware of Code Vulnerabilities	508
	Testing, Testing, Testing	510
	Using Common Sense When Coding	512
	Planning	513
	Coding Standards	514
	Header Comments	514
	Variable Declaration Comments	515
	The Tools	516
	Rule-Based Analyzers	516
	Debugging and Error Handling	517
	Version Control and Source Code	
	Tracking	518
	Creating a Security Plan	520
	Security Planning at the Network Level	522
	Security Planning at the Application Level	523

Security Planning at the Desktop Level	523
Web Application Security Process	524
Summary	527
Solutions Fast Track	528
Frequently Asked Questions	530
Appendix Hack Proofing Your Web Applications Fast Track	533
Index	561