

Contents

Learning to Appreciate the Tao of the Hack

Hackers can be categorized into a series of different types, for instance: Crackers, Script Kiddies or Kidiots, Phreakers, White Hats, Black Hats, and many more. Hackers can be many things—however one thing that all hackers have is a love of a challenge and the ability to stretch their computing knowledge—whether it be for noble or ignoble motivations.

Foreword	xix
Chapter 1 The Zen of Hack Proofing	1
Introduction	2
Learning to Appreciate the Tao of the Hack	2
Hacker	3
Cracker	4
Script Kiddie	5
Phreaker	7
Black Hat, White Hat, What's the Difference?	7
Gray Hat	8
The Role of the Hacker	10
Criminal	10
Magician	11
Security Professional	12
Consumer Advocate	13
Civil Rights Activist	14
Cyber Warrior	15
Motivations of a Hacker	16
Recognition	16
Admiration	17
Curiosity	17
Power and Gain	18
Revenge	19
The Hacker Code	21
Summary	22
Solutions Fast Track	23
Frequently Asked Questions	25

Chapter 2 Classes of Attack 27

Introduction 28

Identifying and Understanding the Classes of Attack 28

Denial of Service 29

Local Vector Denial of Service 29

Network Vector Denial of Service 32

Information Leakage 37

Service Information Leakage 38

Protocol Information Leakage 39

Leaky by Design 41

Leaky Web Servers 42

A Hypothetical Scenario 42

Why Be Concerned with Information Leakage? 43

Regular File Access 44

Permissions 44

Symbolic Link Attacks 45

Misinformation 47

Standard Intrusion Procedure 48

Special File/Database Access 50

Attacks against Special Files 50

Attacks against Databases 50

Remote Arbitrary Code Execution 53

The Attack 54

Code Execution Limitations 55

Elevation of Privileges 55

Remote Privilege Elevation 55

Identifying Methods of Testing for Vulnerabilities 58

Proof of Concept 58

Exploit Code 59

Automated Security Tools 59

Versioning 60

Standard Research Techniques 62

Whois 62

Domain Name System 66

Nmap 69

Web Indexing 70

The Seven Classes of Attack

- Denial of service
- Information leakage
- Regular file access
- Misinformation
- Special file/database access
- Remote arbitrary code execution
- Elevation of privileges

Well-Formed XML Documents

When developing an XML document, certain rules must be followed:

- The document must have exactly one root element.
- Each element must have a start-tag and end-tag.
- The elements must be properly nested.
- The first letter of an attribute's name must begin with a letter or with an underscore.
- A particular attribute name can appear only once in the same start-tag.

Answers to Your Frequently Asked Questions

Q: Can DTDs and schemas be used together?

A: Yes, they can. It's perfectly acceptable to define the structure of data with a DTD and constrain the contents of the structure with a schema.

Summary	73
Solutions Fast Track	75
Frequently Asked Questions	76

Chapter 3 Reviewing the Fundamentals of XML **79**

Introduction	80
An Overview of XML	80
The Goals of XML	81
What Does an XML Document Look Like?	81
Creating an XML Document	82
Creating an XML Document in VS.NET XML Designer	82
Empty Element	86
Structure of an XML Document	87
Well-Formed XML Documents	87
Transforming XML through XSLT	88
XSL Use of Patterns	92
XPath	95
Summary	97
Solutions Fast Track	97
Frequently Asked Questions	99

Chapter 4 Document Type: The Validation Gateway **101**

Introduction	102
Document Type Definitions and Well-Formed XML Documents	102
Schema and Valid XML Documents	106
XML Schema Data Types	110
Learning About Plain-Text Attacks	112
Plain-Text Attacks	113
Example: HTML Escape Codes	114
Unicode	116
Understanding How Validation Is Processed in XML	117
Validate the Input Text	118

Canonicalization	118
Validating Unicode	121
Validate the Document or Message	124
Is the XML Well Formed?	126
Using DTDs for Verifying the Proper Structure	126
Using Schema for Data Consistency	127
Online Validation Methods and Mechanisms	128
Summary	135
Solutions Fast Track	138
Frequently Asked Questions	140

Chapter 5 XML Digital Signatures 143

XML Signatures Can Be Applied in Three Basic Forms

- **Enveloped form** The signature is within the document.
- **Enveloping form** The document is within the signature, as shown in the following example.
- **Detached form** The signature references a document that is elsewhere through a universal resource identifier (URI).

Introduction	144
Understanding How a Digital Signature Works	144
Basic Digital Signature and Authentication Concepts	144
Why a Signature Is Not a MAC	145
Public and Private Keys	145
Why a Signature Binds Someone to a Document	146
Learning the W3C XML Digital Signature	146
Applying XML Digital Signatures to Security	149
Examples of XML Signatures	150
An Enveloping Signature Example	152
An Example of an Enveloped Signature	154
A Detached Signature Example	157
All Together Now: An Example of Multiple References	161
Signing Parts of Documents	163
Using XPath to Transform a Document	164
Using XSLT to Transform a Document	166
Using Manifests to Manage Lists of Signed Elements	169
Establishing Identity By Using X509	172

Required and Recommended Algorithms	173
Cautions and Pitfalls	175
Vendor Toolkits	176
Summary	178
Solutions Fast Track	179
Frequently Asked Questions	181

Chapter 6 Encryption in XML 183

Introduction	184
Understanding the Role of	
Encryption in Messaging Security	184
Security Needs of Messaging	185
Privacy and Confidentiality	185
Authentication and Integrity	186
Nonrepudiation	190
Encryption Methods	191
AES	191
DES and 3-DES	193
RSA and RC4	195
Stream and Block Ciphers	196
Key Management Schemes	197
Learning How to Apply Encryption to XML	199
XML Transforms Before Encryption	204
Canonicalization	205
Flowchart of Encryption Process	207
Understanding Practical Usage of Encryption	207
Signing in Plain Text, Not Cipher Text	207
XPath Transforms	210
Signing the Cipher-Text Version	
Prevents Encryption Key Changes	210
Authentication by MAC Works on	
Cipher Text	210
Cipher Text Cannot Validate Plain Text	211
Encryption Might Not Be Collision	
Resistant	211
Summary	213
Solutions Fast Track	213
Frequently Asked Questions	214

Tools & Traps...

IBM's XML Security Suite

Although IBM is planning to release a new version relatively soon, we cover some points of XML Security Suite here:

- **XML signatures** Verify a digital signature, canonicalize a document, and verify its form as well as XPath transformations.
- **Nonrepudiation** It is designed to provide nonrepudiation.
- **Java** It is written in Java, hence, you must be running Java to use the security suite.

Tools & Traps...**Viewing XML Files**

If you want to view an XML file as it would be parsed, simply use your Web browser to open the file. Most current Web browsers have built-in XML parsers that allow you to view XML files in an expandable/collapsible format. In addition, some even support the use of DTD files to verify the format of your XML file.

.NET Code Access Security Model

The .NET code access security model is built around a number of characteristics:

- Stack walking
- Code identity
- Code groups
- Declarative and imperative security
- Requesting permissions
- Demanding permissions
- Overriding security checks
- Custom permissions

Chapter 7 Role-Based Access Control 215

Introduction	216
Learning About Stateful Inspection	216
Packet Filtering	216
Application Layer Gateway	217
The FTP Process	219
Firewall Technologies and XML	220
First, You Inspect the State	221
Baselines	222
Evaluating State Changes	223
Default Behavior Affects Security	225
Learning About Role-Based Access Control and Type Enforcement Implementations	227
NSA: The Flask Architecture	229
SELinux	232
Applying Role-Based Access Control Ideas in XML	238
Know When to Evaluate	243
Protect Data Integrity	244
RBAC and Java	245
Fencing in JavaScript	246
Validate Your Java Code	246
Validate Your ActiveX Objects	247
Tools to Implement RBAC Efforts	248
Summary	254
Solutions Fast Track	255
Frequently Asked Questions	256

Chapter 8 Understanding .NET and XML Security 257

Introduction	258
The Risks Associated with Using XML in the .NET Framework	258
Confidentiality Concerns	259
.NET Internal Security as a Viable Alternative	260
Permissions	261
Principal	262

Authentication	263
Authorization	263
Security Policy	263
Type Safety	264
Code Access Security	264
.NET Code Access Security Model	264
Stack Walking	265
Code Identity	266
Code Groups	267
Declarative and Imperative Security	270
Requesting Permissions	271
Demanding Permissions	275
Overriding Security Checks	277
Custom Permissions	282
Role based Security	283
Principals	284
WindowsPrincipal	284
GenericPrincipal	286
Manipulating Identity	287
Role-Based Security Checks	288
Security Policies	291
Creating a New Permission Set	294
Modifying the Code Group Structure	299
Remoting Security	305
Cryptography	306
Security Tools	309
Securing XML—Best Practices	311
XML Encryption	311
XML Digital Signatures	317
Summary	320
Solutions Fast Track	321
Frequently Asked Questions	326
Chapter 9 Reporting Security Problems	331
Introduction	332
Understanding Why Security Problems Need to Be Reported	332

Deciding How Much Detail to Publish

- Take great care in deciding whether or not you want to provide exploit code with your NSF report. Be aware that there are times when exploit code is necessary for reporting the problem.
- You must be prepared to take a slight risk when reporting security flaws. You could end up facing the vendor's wrath or imposing undue risk on the public at large.
- Be extra cautious in describing any security flaw that requires the circumvention of a vendor's copyright protection mechanisms, as this is a very gray area for the time being.

Full Disclosure	333
Determining When and to Whom to Report the Problem	337
Whom to Report Security Problems to?	337
How to Report a Security Problem to a Vendor	340
Deciding How Much Detail to Publish	341
Publishing Exploit Code	341
Problems	342
Repercussions from Vendors	342
Reporting Errors	344
Risk to the Public	344
Summary	345
Solutions Fast Track	346
Frequently Asked Questions	347

Hack Proofing XML Fast Track 351

Index 369