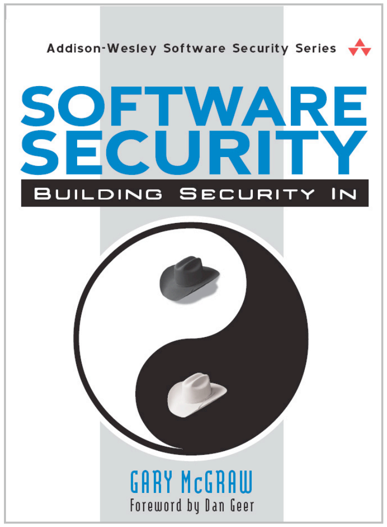
Software Security Testing:

Seeking security in an insecure world

Gary McGraw, Ph.D.
CTO, Cigital

http://www.cigital.com

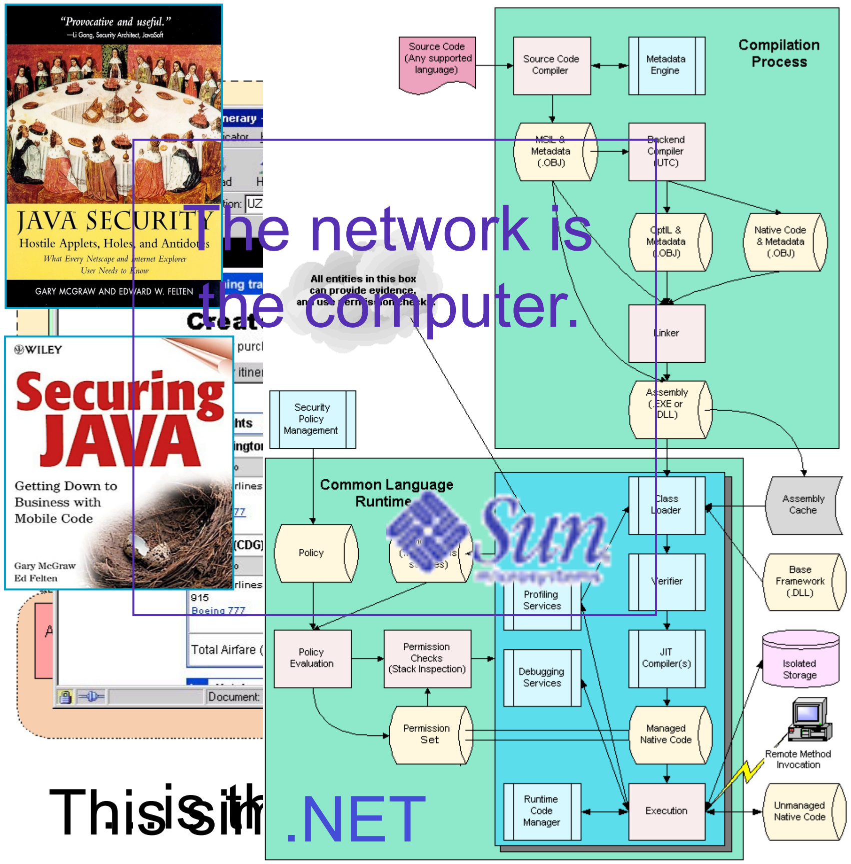


Systematic Techniques to Find Problems Fast
Web Security Testing Cookbook
O'REILLY
Paco Hope & Ben Walther

Software security is getting harder

The Trinity of Trouble

- **Connectivity**
  - The Internet is everywhere and most software is on it
- **Complexity**
  - Networked, distributed, mobile code is hard
- **Extensibility**
  - Systems evolve in unexpected ways and are changed on the fly



Old school security is reactive

- Defend the "perimeter" with a firewall
  - To keep stuff out
- Promulgate "penetrate and patch"
- "Review" products when they're complete
  - Throw it over the wall testing
  - Too much weight on penetration testing
- Over-rely on security functions
  - "We use SSL"

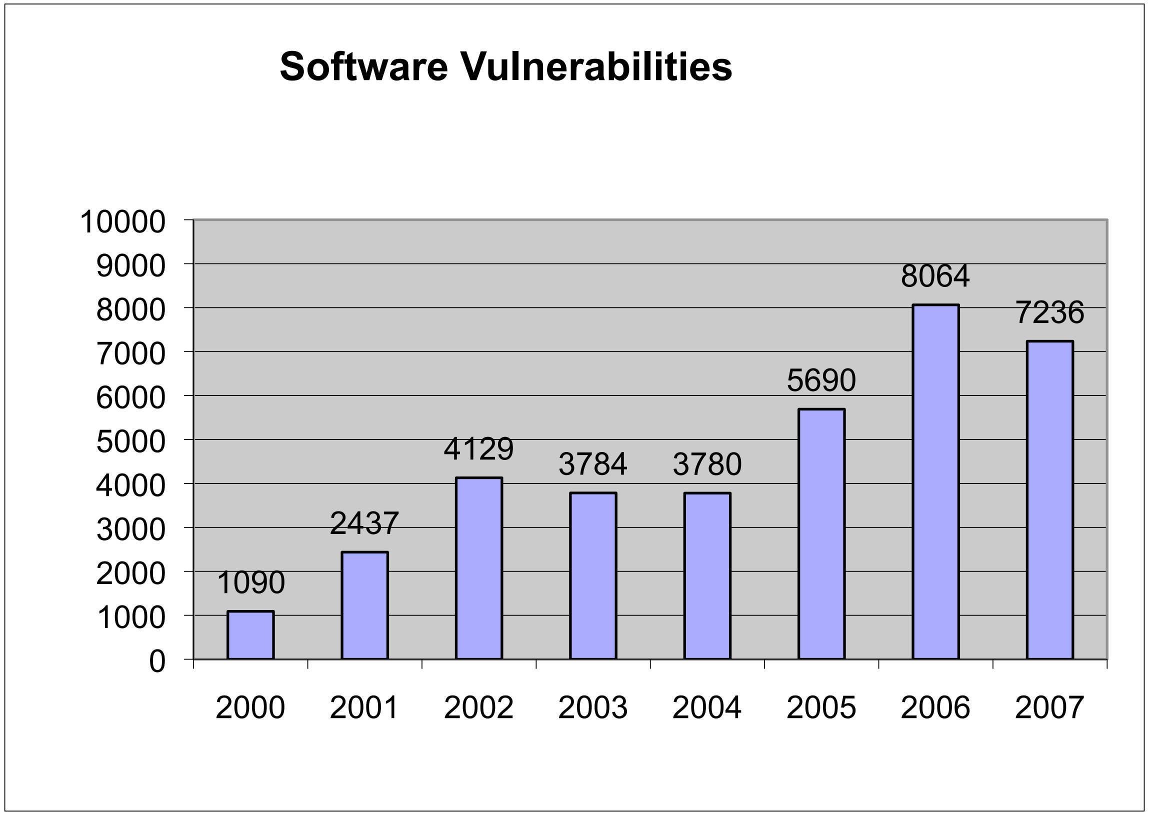
The "network guy with keys" does not really understand software testing. Builders are only recently getting involved in security.

Making software behave is hard

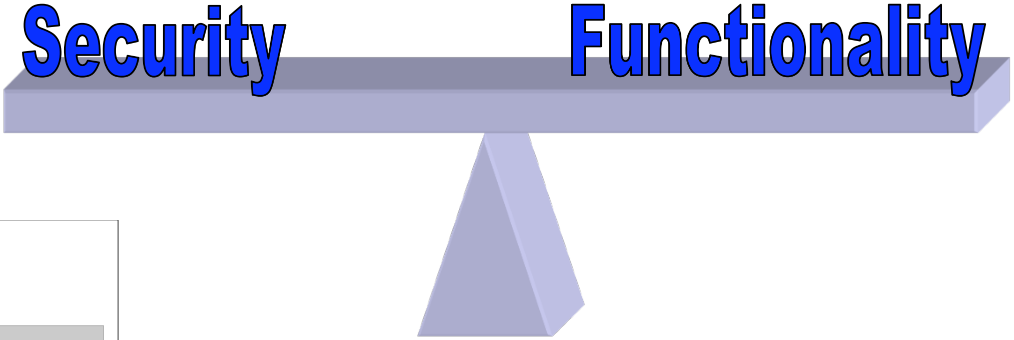
- Can you test in quality?
- How do you find (adaptive) defects in code?
- What about bad guys doing evil on purpose?

- What’s the difference between security testing and functional testing?
- How can you analyze security design?
- How can you codify non-functional, emergent requirements like security?
- Can you measure security?

Software vulnerability growth

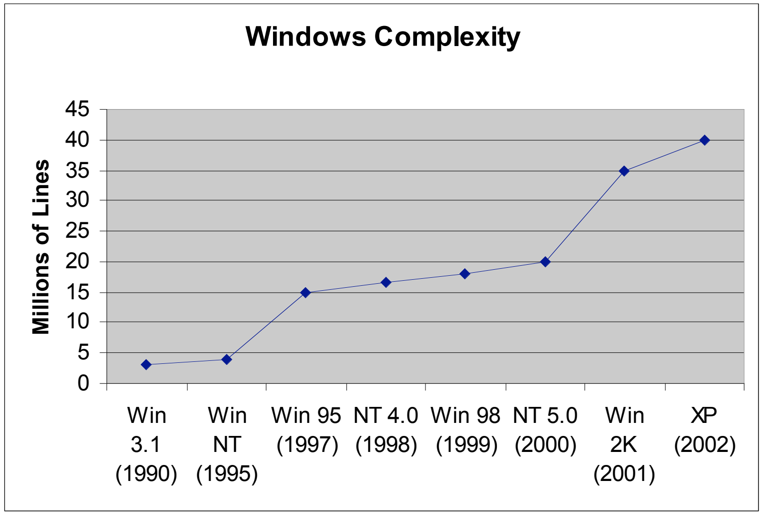


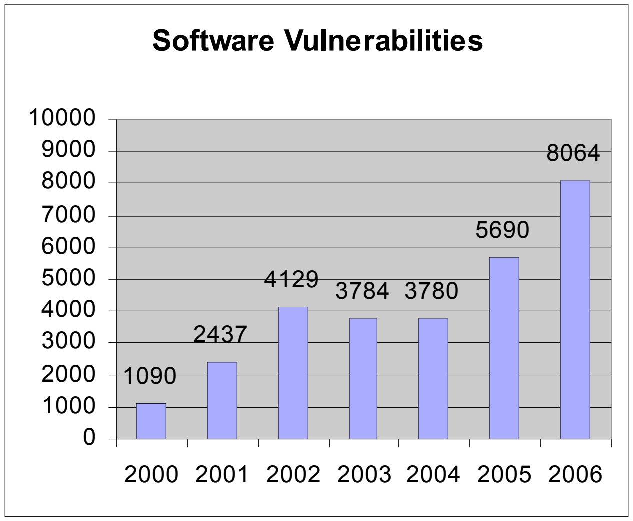
The classic security tradeoff



Security

Functionality





Security problems are complicated

IMPLEMENTATION BUGS

50%

- Buffer overflow
  - String format
  - One-stage attacks
- Race conditions
  - TOCTOU (time of check to time of use)
- Unsafe environment variables
- Unsafe system calls
  - System()
- Untrusted input problems

ARCHITECTURAL FLAWS

50%

- Misuse of cryptography
- Compartmentalization problems in design
- Privileged block protection failure (DoPrivilege())
- Catastrophic security failure (fragility)
- Type safety confusion error
- Insecure auditing
- Broken or illogical access control (RBAC over tiers)
- Method over-riding problems (subclass issues)
- Signing too much code

Security = breaking stuff + building stuff

- Security requires two hats
  - Offense and defense
  - Building and breaking
- Security design based on software engineering
- Security analysis based on attack
- Testing has two flavors
  - Functional security testing (constructive)
  - Risk-based security testing (destructive)

A Software Security Framework

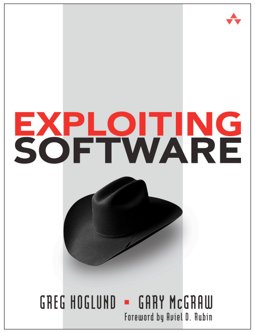
Governance	Intelligence	SDL Touchpoints	Deployment
Strategy and Metrics	Attack Models	Architecture Analysis	Penetration Testing
Compliance and Policy	Security Features and Design	Code Review	Software Environment
Training	Standards and Requirements	Security Testing	Configuration Management and Vulnerability Management

- Four domains
- Twelve practices
- See informIT article
- http://www.informit.com/articles/article.aspx?p=1271382

Security as Knowledge Intensive

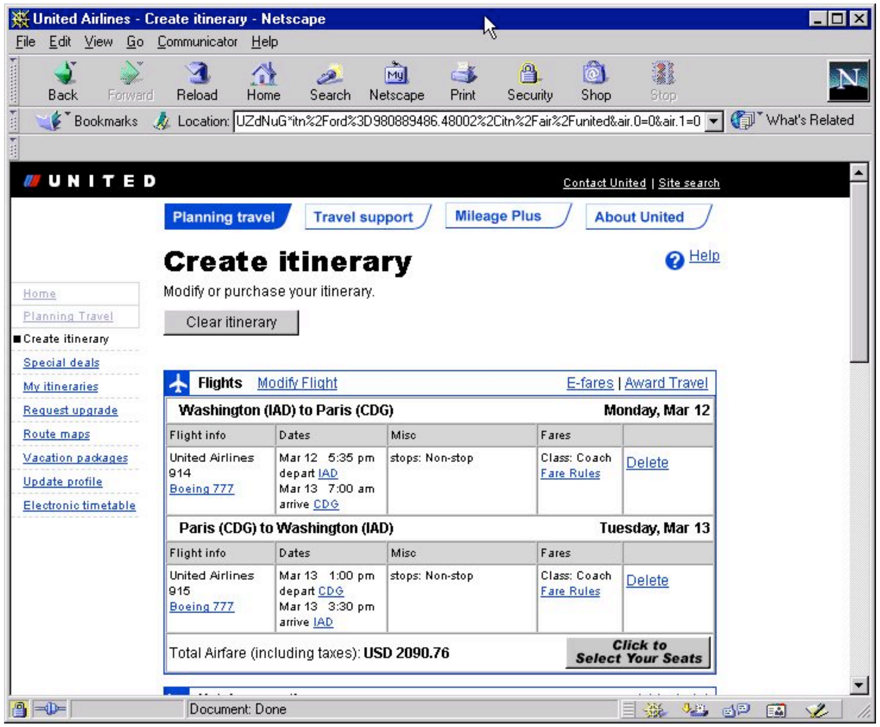
Knowledge: 48 Attack Patterns

- Make the Client Invisible
- Target Programs That Write to Privileged OS Resources
- Use a User-Supplied Configuration File to Run Commands That Elevate Privilege
- Make Use of Configuration File Search Paths
- Direct Access to Executable Files
- Embedding Scripts within Scripts
- Leverage Executable Code in Nonexecutable Files
- Argument Injection
- Command Delimiters
- Multiple Parsers and Double Escapes
- User-Supplied Variable Passed to File System Calls
- Postfix NULL Terminator
- Postfix, Null Terminate, and Backslash
- Relative Path Traversal
- Client-Controlled Environment Variables
- User-Supplied Global Variables (DEBUG=1, PHP Globals, and So Forth)
- Session ID, Resource ID, and Blind Trust
- Analog In-Band Switching Signals (aka “Blue Boxing”)
- Attack Pattern Fragment: Manipulating Terminal Devices
- Simple Script Injection
- Embedding Script in Nonscript Elements
- XSS in HTTP Headers
- HTTP Query Strings
- User-Controlled Filename
- Passing Local Filenames to Functions That Expect a URL
- Meta-characters in E-mail Header
- File System Function Injection, Content Based
- Client-side Injection, Buffer Overflow
- Cause Web Server Misclassification
- Alternate Encoding the Leading Ghost Characters
- Using Slashes in Alternate Encoding
- Using Escaped Slashes in Alternate Encoding
- Unicode Encoding
- UTF-8 Encoding
- URL Encoding
- Alternative IP Addresses
- Slashes and URL Encoding Combined
- Web Logs
- Overflow Binary Resource File
- Overflow Variables and Tags
- Overflow Symbolic Links
- MIME Conversion
- HTTP Cookies
- Filter Failure through Buffer Overflow
- Buffer Overflow with Environment Variables
- Buffer Overflow in an API Call
- Buffer Overflow in Local Command-Line Utilities
- Parameter Expansion
- String Format Overflow in syslog()



Attack pattern 1: Make the client invisible

- Remove the client from the communications loop and talk directly to the server
- Leverage incorrect trust model (never trust the client)
- Example: hacking browsers that lie (opera cookie foo)



Location: UZdNuG*itn%2Ford%3D980889486.48002%2Citn%2Fair%2Funited&air.0=0&air.1=0

Create itinerary

Modify or purchase your itinerary.

Flights Modify Flight | E-fares | Award Travel

Washington (IAD) to Paris (CDG) **Monday, Mar 12**

Flight info	Dates	Misc	Fares	
United Airlines 914 Boeing 777	Mar 12 5:35 pm depart IAD Mar 13 7:00 am arrive CDG	stops: Non-stop	Class: Coach Fare Rules	Delete

Paris (CDG) to Washington (IAD) **Tuesday, Mar 13**

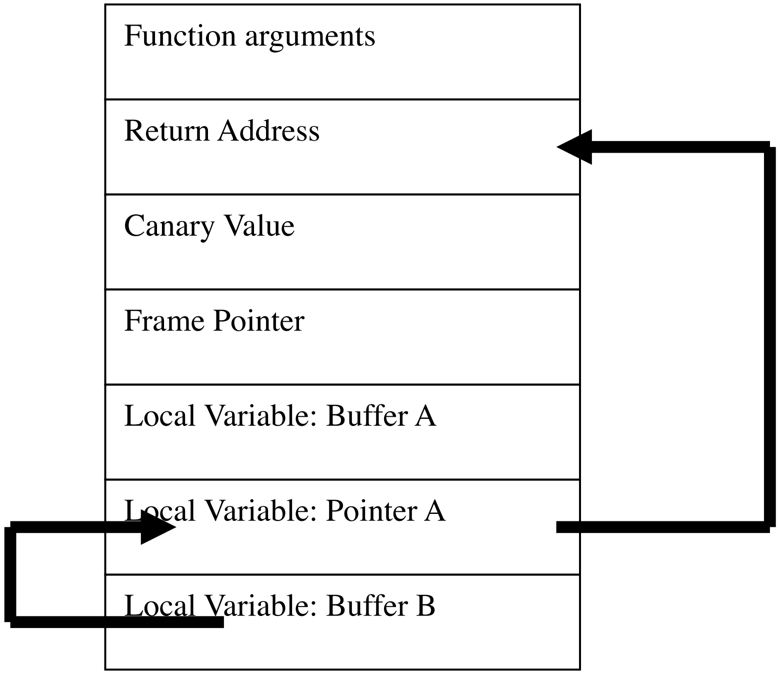
Flight info	Dates	Misc	Fares	
United Airlines 915 Boeing 777	Mar 13 1:00 pm depart CDG Mar 13 3:30 pm arrive IAD	stops: Non-stop	Class: Coach Fare Rules	Delete

Total Airfare (including taxes): **USD 2090.76**

Attacker’s toolkit: buffer overflow foo

- Find targets with static analysis
- Change program control flow
  - Heap attacks
  - Stack smashing
  - Trampolining
  - Arc injection
- Particular examples
  - Overflow binary resource files (used against Netscape)
  - Overflow variables and tags (Yamaha MidiPlug)
  - MIME conversion fun (Sendmail)
  - HTTP cookies (apache)

- Trampolining past a canary



Warning! Knowledge can be easily misused

Software security

- Requires input into design and implementation
- High expertise
- Design software to be secure
- Build secure code
- Security analysis
- Security testing
- Inside → Out

"Application security"

- Works for COTS software
- Low expertise
- Protect installed software from harm
- Protection against malicious code
- Policy issues
- Outside → In

badness-ometer

Top 11 reasons why top 10 lists don’t work

1. Executives don’t care about technical bugs
2. Too much focus on bugs
3. Vulnerability lists help auditors more than developers
4. One person’s bug is another person’s yawner
5. Using bug parade lists for training leads to awareness but does not educate.
6. Bug lists change with the prevailing technology winds
7. Top ten lists mix levels
8. Automated tools can find bugs---let them
9. Metrics built on top ten lists are misleading
10. When it comes to testing, security requirements are more important than vulnerability lists.
11. Ten is not enough.

http://www.informit.com/articles/article.aspx?p=1322398

BSIMM-Ten surprising things

1. Bad metrics hurt
2. Secure-by default frameworks
3. Nobody uses WAFs
4. QA can’t do software security
5. Evangelize over audit
6. ARA is hard
7. Practitioners don’t talk attacks
8. Training is advanced
9. Pen testing is diminishing
10. Fuzz testing

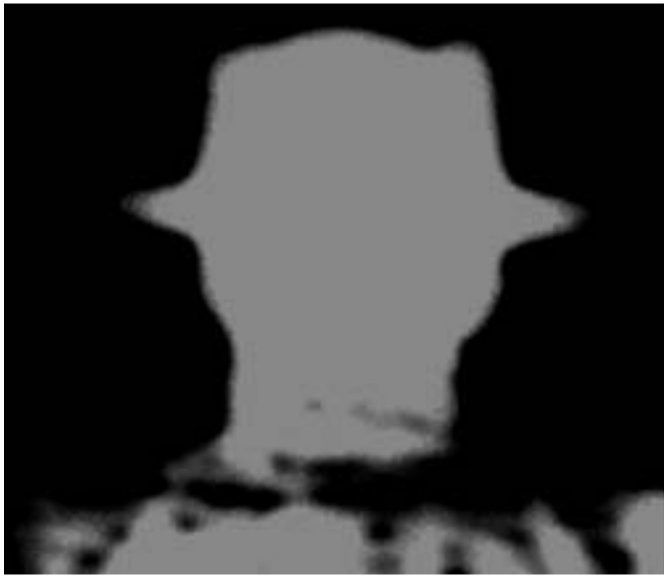
- http://www.informit.com/articles/article.aspx?p=1315431

Attackers are Software People

Attackers do not distinguish bugs and flaws

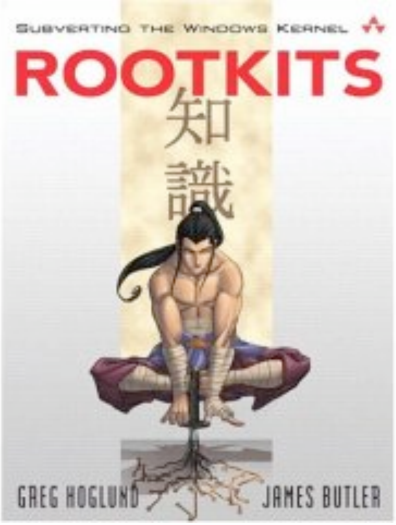
- Both bugs and flaws lead to vulnerabilities that can be exploited

- Attackers write code to break code
- Defenders are network operations people
  - Code?! What code?



The attacker’s toolkit

- The standard attacker’s toolkit has lots of (software analysis) stuff
  - Disassemblers and decompilers
  - Binary scanners
  - Control flow, data flow, and coverage tools
  - APISPY32
  - Breakpoint setters and monitors
  - Buffer overflow kits
  - Shell code, payloads (multi-platform)
  - Rootkits (kernel, hardware)



Attacker's toolkit: other miscellaneous tools

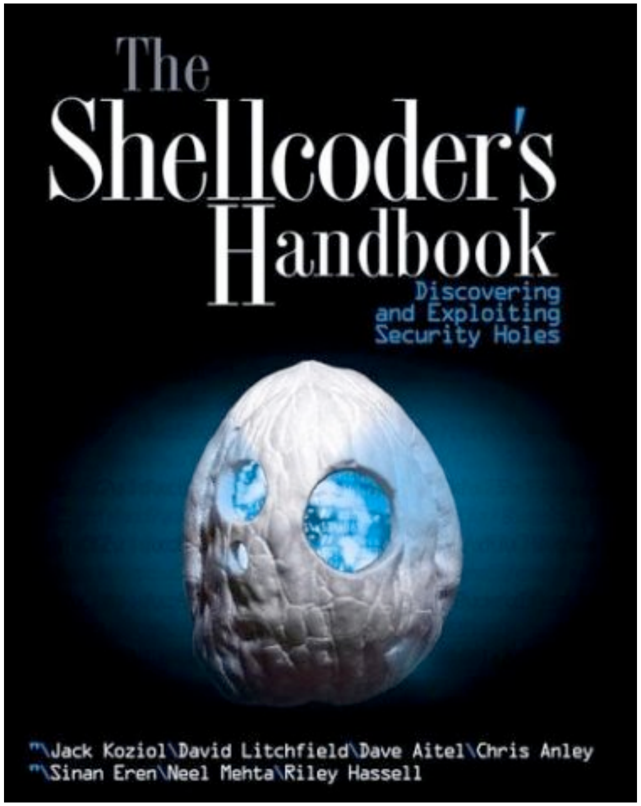
- Debuggers (user-mode)
- Kernel debuggers
  - SoftIce
- Fault injection tools
  - FUZZ
  - Failure simulation tool
  - Hailstorm
  - Holodeck
- Boron tagging
- The "depends" tool
- Grammar rewriters

Is FUZZ good?

- Wisconsin academics invent the notion of sending random noise to UNIX utilities
  - Fuzz I: 1990
  - Fuzz II: ten years later
- Fifteen years later, security people hit on the same idea

- FUZZ can be useful
  - SPIKE
  - Peachfuzz
  - Mangle (HTML)
  - FileFuzz
  - beStrorm
  - Codenomicon
- White box testing is better

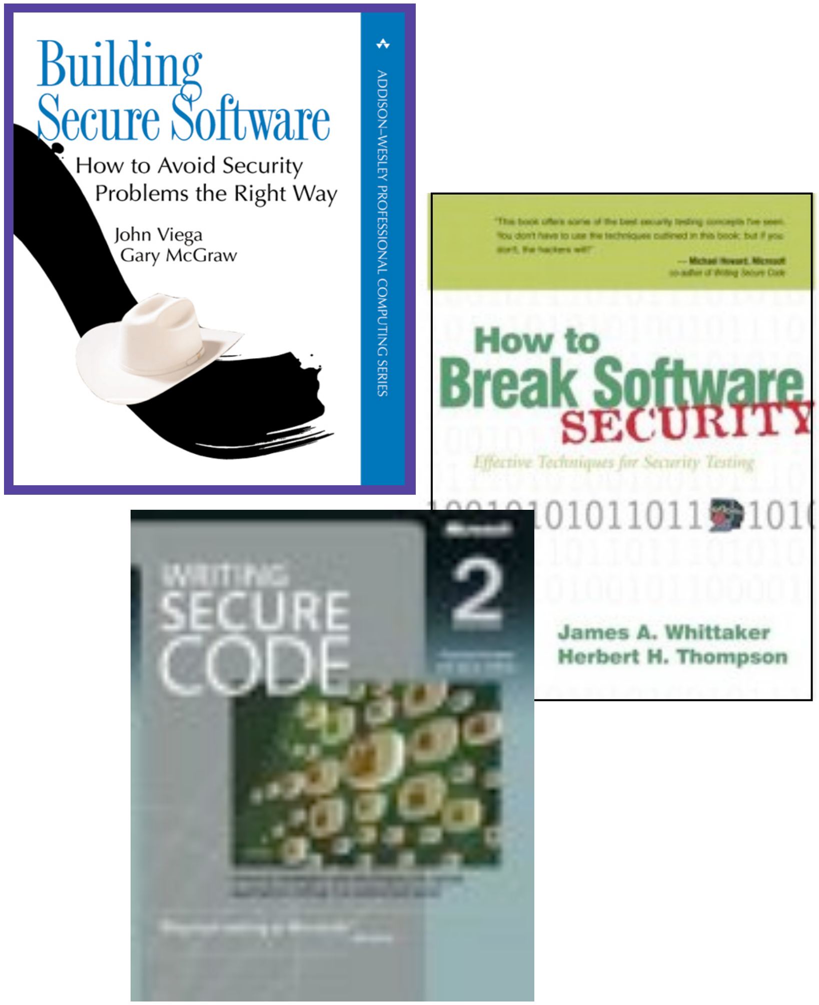
Breaking stuff is important


The
Shellcoder's
Handbook
Discovering
and Exploiting
Security Holes
Jack Koziol\David Litchfield\Dave Aitel\Chris Anley
Sinan Eren\Neel Mehta\Riley Hassell

- Learning how to think like an attacker is essential (especially for good testing)
- Think hard about the “can’ts” and “won’ts”
- Do not shy away from teaching attacks
  - Engineers learn from stories of failure
  - Testers must deeply understand how things break

Resources on security testing

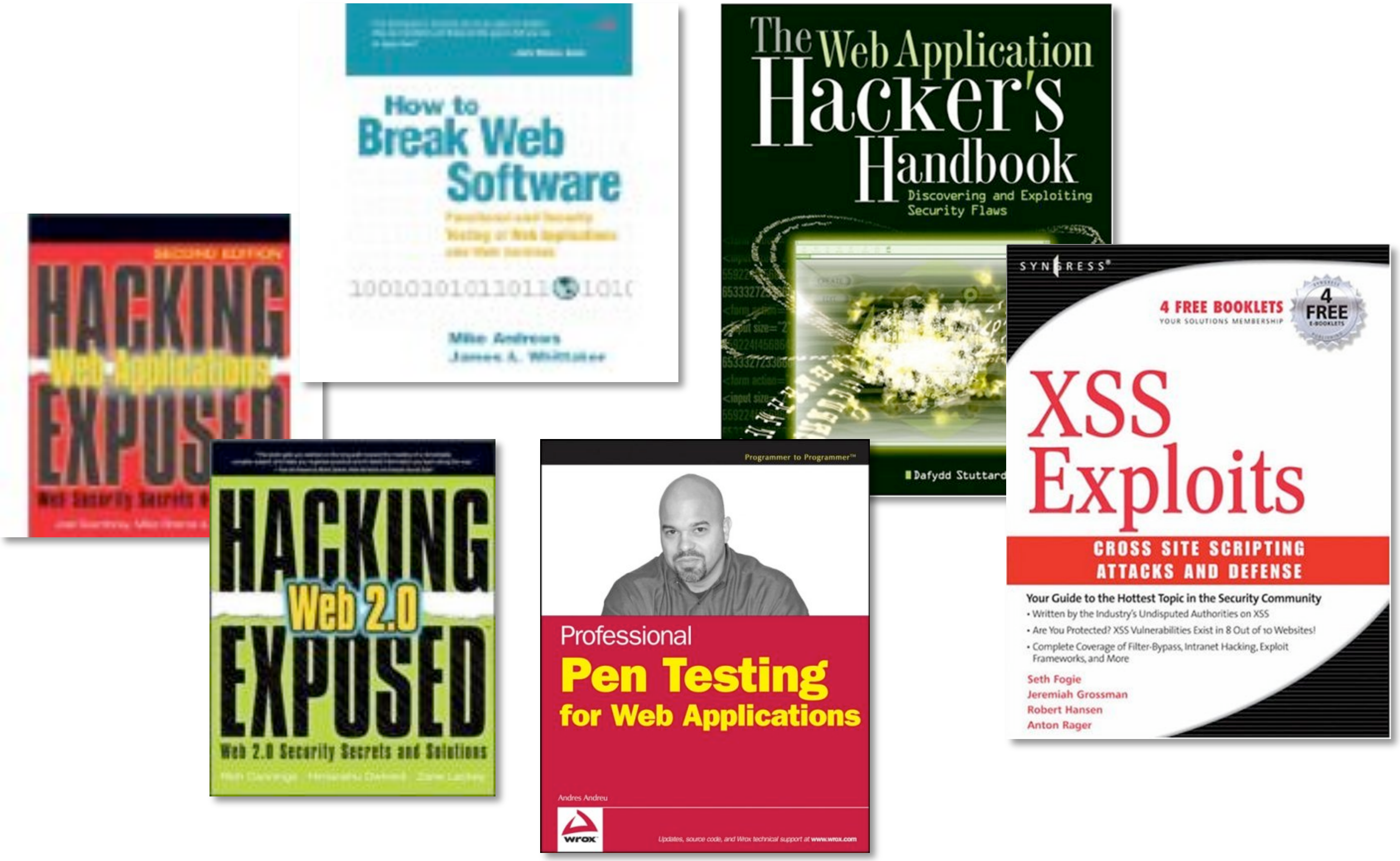
- Building Secure Software (Viega/McGraw)
- Writing Secure Code (Howard/ LeBlanc)
- How to Break Software Security (Whittaker/Thompson)
- Web Security Testing Cookbook (Hope/Walther)


Building Secure Software
How to Avoid Security Problems the Right Way
John Viega
Gary McGraw
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES
How to Break Software SECURITY
Effective Techniques for Security Testing
James A. Whittaker
Herbert H. Thompson
WRITING SECURE CODE
2

Web Security Testing Cookbook

Systematic Techniques to Find Problems Fast
Web Security Testing Cookbook
O'REILLY
Paco Hope & Ben Walther

What it ain't


SECOND EDITION
HACKING Web Applications EXPOSED
Web Security Secrets &
How to Break Web Software
Functional and Security Testing of Web Applications and Web Services
Mike Andrews
James A. Whittaker
The Web Application Hacker's Handbook
Discovering and Exploiting Security Flaws
Dafydd Stuttard
HACKING Web 2.0 EXPOSED
Web 2.0 Security Secrets and Solutions
Programmer to Programmer™
Professional
Pen Testing for Web Applications
Andres Andreu
wrox
SYNGRESS®
4 FREE BOOKLETS
YOUR SOLUTIONS MEMBERSHIP
4 FREE E-BOOKLETS
XSS Exploits
CROSS SITE SCRIPTING ATTACKS AND DEFENSE
Your Guide to the Hottest Topic in the Security Community
• Written by the Industry's Undisputed Authorities on XSS
• Are You Protected? XSS Vulnerabilities Exist in 8 Out of 10 Websites!
• Complete Coverage of Filter-Bypass, Intranet Hacking, Exploit Frameworks, and More
Seth Fogie
Jeremiah Grossman
Robert Hansen
Anton Rager

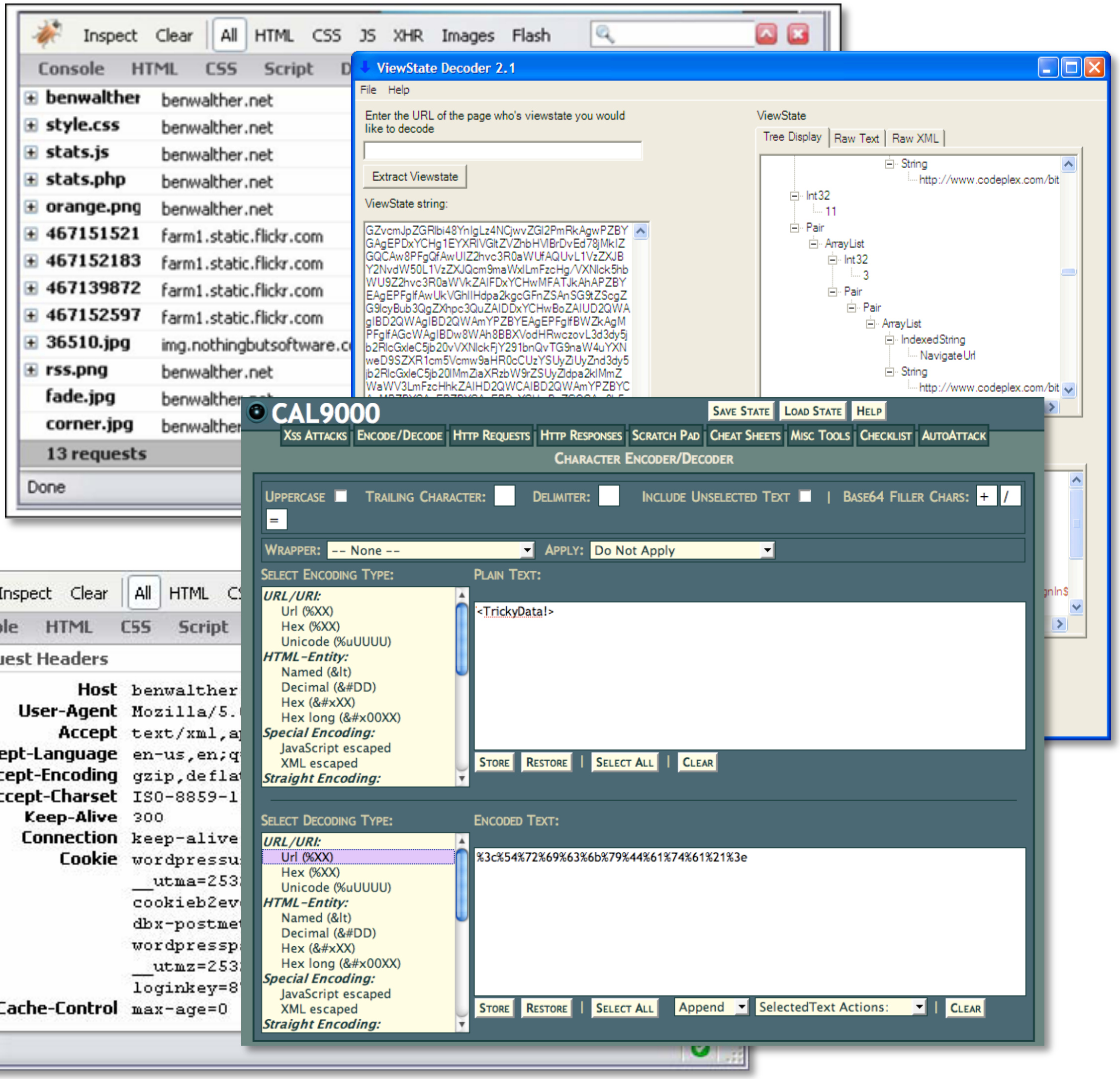
What it IS

- Reference and job aid for web testers
  - Exploratory testing
  - Regression testing
  - Automated testing
  - Unit testing
- Starts *really* basic
- Ends rather complicated

Systematic Techniques to Find Problems Fast
Web Security Testing Cookbook
O'REILLY
Paco Hope & Ben Walther

Table of Contents

1. Introduction
2. Installing Free Tools
3. Basic Observation
4. Web-Oriented Data Encoding
5. Tampering with Input
6. Automated Bulk Scanning
7. Automating Tasks with cURL
8. Automating Tasks with LibWWWPerl
9. Seeking Design Flaws
10. Attacking AJAX
11. Manipulating Sessions
12. Multifaceted Tests



benwalther	benwalther.net
style.css	benwalther.net
stats.js	benwalther.net
stats.php	benwalther.net
orange.png	benwalther.net
467151521	farm1.static.flickr.com
467152183	farm1.static.flickr.com
467139872	farm1.static.flickr.com
467152597	farm1.static.flickr.com
36510.jpg	img.nothingbutsoftware.c
rss.png	benwalther.net
fade.jpg	benwalther
corner.jpg	benwalther
13 requests	

Request Headers

Host	benwalther
User-Agent	Mozilla/5.
Accept	text/xml,a
Accept-Language	en-us,en;q
Accept-Encoding	gzip,defla
Accept-Charset	ISO-8859-1
Keep-Alive	300
Connection	keep-alive
Cookie	wordpressu __utma=253 cookieb2ev dbx-postme wordpressp __utmz=253 loginkey=8
Cache-Control	max-age=0

Enter the URL of the page who's viewstate you would like to decode

ViewState string:

GZvcmJpZGRlbi48YnIgLz4NCjwvZGl2PmRkAgwPZBY
GAgEPDxYCHg1EYXRlVGltZVZhbHVlBrDvEd78jMkIZ
GQCAw8PFgQfAwUIZ2hvc3R0aWUfAQUvL1VzZXJB
Y2NvdW50L1VzZXJQcm9maWxlLmFzcHg/VXNlck5hb
WU9Z2hvc3R0aWVkZAIFDxYCHwMFATJkAhAPZBY
EAgEPFgIfAwUkVGhlIHdpa2kgcGFnZSAnSG9tZScgZ
G9lcyBub3QgZXhpc3QuZAIDDxYCHwBoZAIUD2QWA
gIBD2QWAgIBD2QWAmYPZBYEAgEPFgIfBWZkAgM
PFgIfAGcWAgIBDw8WAh8BBXVodHRwczovL3d3dy5j
b2RlcGxleC5jb20vVXNlckFjY291bnQvTG9naW4uYXN
weD9SZXR1cm5Vcmw9aHR0cCUzYSUyZiUyZnd3dy5
b2RlcGxleC5jb20lMmZiaXRzbW9rZSUyZldpa2klMmZ
WaWV3LmFzcHhkZAIHD2QWCAIBD2QWAmYPZBYC

String
http://www.codeplex.com/bit
Int32
11
Pair
ArrayList
Int32
3
Pair
Pair
ArrayList
IndexedString
NavigateUrl
String
http://www.codeplex.com/bit

Uppercase □ Trailing Character: Delimiter: Include Unselected Text □ | Base64 Filler Chars: + / =

Wrapper: -- None --
Apply: Do Not Apply

Select Encoding Type:
URL/URI:
Url (%XX)
Hex (%XX)
Unicode (%uUUUU)
HTML-Entity:
Named (&lt)
Decimal (&#DD)
Hex (&#xXX)
Hex long (&#x00XX)
Special Encoding:
JavaScript escaped
XML escaped
Straight Encoding:

Plain Text:
<TrickyData!>

Select Decoding Type:
URL/URI:
Url (%XX)
Hex (%XX)
Unicode (%uUUUU)
HTML-Entity:
Named (&lt)
Decimal (&#DD)
Hex (&#xXX)
Hex long (&#x00XX)
Special Encoding:
JavaScript escaped
XML escaped
Straight Encoding:

Encoded Text:
%3c%54%72%69%63%6b%79%44%61%74%61%21%3e

Example: Login to eBay

- Series of commands
- Build up state/ cookies
- Check for username in output

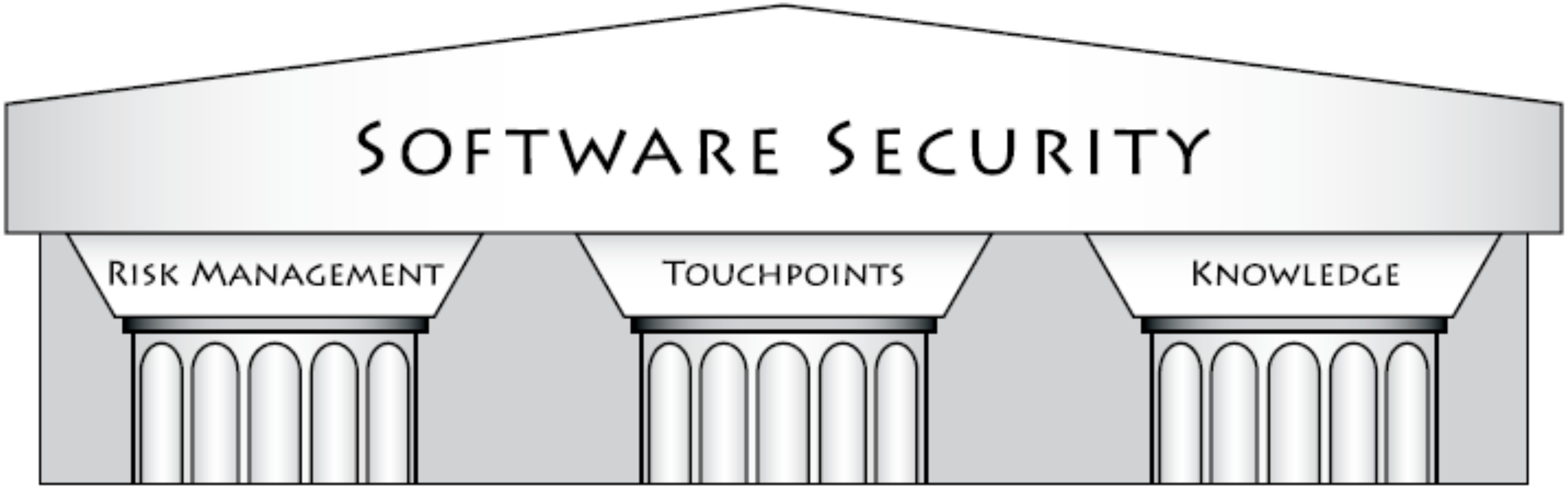
```
${CURL} -s -L -A "${UA}" -c "${JAR}"\
   -b "${JAR}" -e ";auto” \
   -d MfcISAPICommand=SignInWelcome \
   -d siteid=0 -d co_partnerId=2 -d UsingSSL=1 \
   -d ru= -d pp= -d pa1= -d pa2= -d pa3=       \
   -d i1=-1 -d pageType=-1 -d rtmData=         \
   -d userid="${USER}"              \
   -d pass="${PASS}"                \
   -o "step-${step}.html"           \
  "https://signin.ebay.com/ws/..."

if [ $? = 0 ]; then
     step=$step+1
     echo -n "OK] [${step} "
 else
     echo "FAIL]"
     exit 1
 fi
```

Other topics

- LDAP injection
- Zip of death
- Billion laughs
- Pathological XML
- Malicious cookies

- All of these are scripted
- All are repeatable
- All can be part of a routine QA process

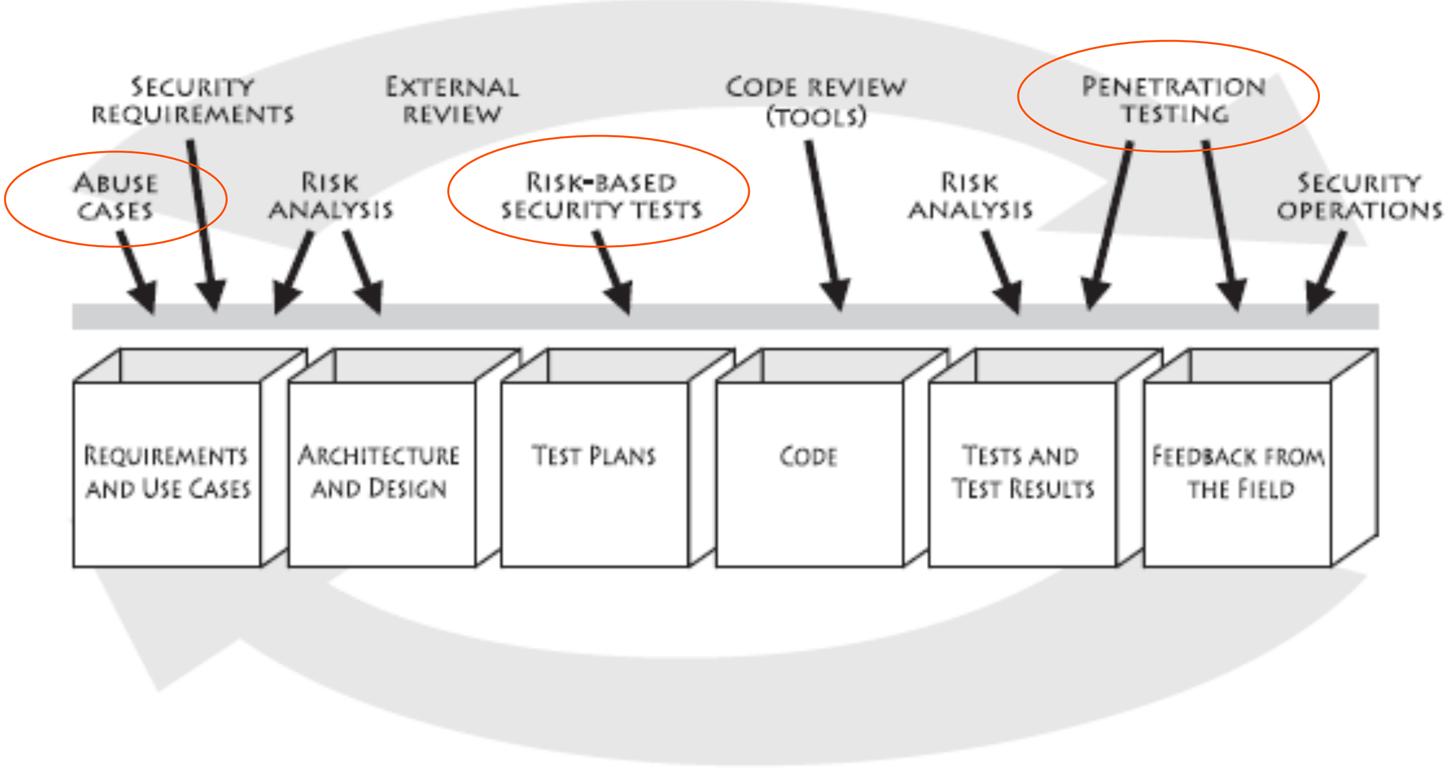
Stuff that Works for Cigital



Three pillars of software security

- Risk management framework
- Touchpoints
- Knowledge

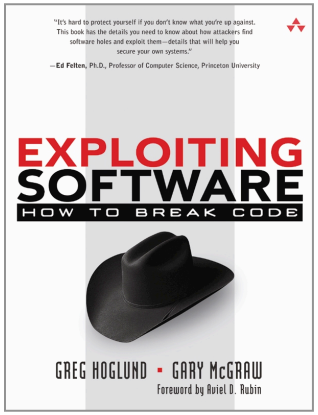
Software security touchpoints



Touchpoint: Abuse cases

- Use cases formalize normative behavior (and assume correct usage)
- Describing non-normative behavior is a good idea
  - Prepare for abnormal behavior (attack)
  - Misuse or abuse cases do this
  - Uncover exceptional cases
- Leverage the fact that designers know more about their system than potential attackers do
- Document explicitly what the software will do in the face of illegitimate use

- Abuse cases are great for test planning


"It's hard to protect yourself if you don't know what you're up against. This book has the details you need to know about how attackers find software holes and exploit them—details that will help you secure your own systems."
—Ed Felten, Ph.D., Professor of Computer Science, Princeton University
EXPLOITING
SOFTWARE
HOW TO BREAK CODE
GREG HOGLUND • GARY McGRAW
Foreword by Aviel D. Rubin

Touchpoint: Security testing

- Test security functionality
  - Cover non-functional requirements
  - Security software probing

- Risk-based testing
  - Use architectural risk analysis results to drive scenario-based testing
  - Concentrate on what “you can’t do”
  - Think like an attacker
  - Informed red teaming

Touchpoint: Risk-based testing

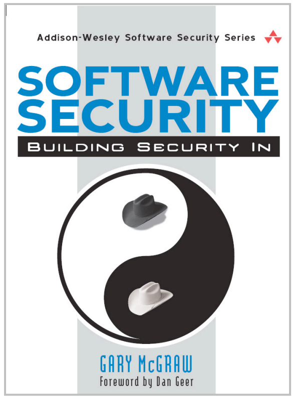
- Identify areas of potential risk in the system
  - Requirements
  - Design
  - Architecture
- Use abuse cases to drive testing according to risk
- Build attack and exploit scenarios based on identified risks
- Test risk conditions explicitly

- Example: Overly complex object-sharing system in Java Card

Touchpoint: Penetration testing

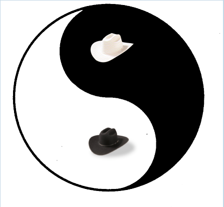
- A very good idea since software is bound in an environment
- How does the complete system work in practice?
  - Interaction with network security mechanisms
  - Firewalls
  - Applied cryptography
- Penetration testing should be driven by risks uncovered throughout the lifecycle

- Not a silver bullet!



Always: External review

- Having outside eyes look at your system is essential
  - Designers and developers naturally get blinders on
  - External just means outside of the project
  - This is knowledge intensive
- Outside eyes make it easier to “assume nothing”
  - Find assumptions, make them go away
- Red teaming is a weak form of external review
  - Penetration testing is too often driven by outside→in perspective
  - External review must include architecture analysis
- Security expertise and experience really helps



Where to Learn More

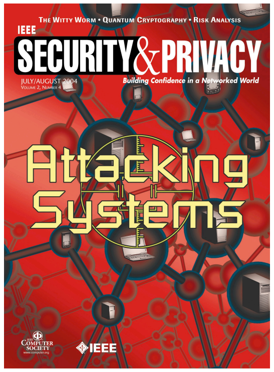
informIT & Justice League

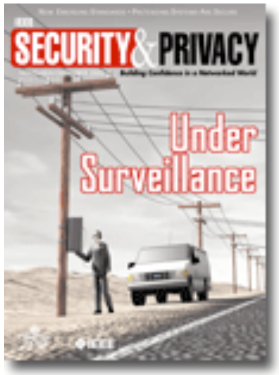
- www.informIT.com
- No-nonsense monthly security column by Gary McGraw

- www.cigital.com/justiceleague
- In-depth thought leadership blog from the Cigital Principals
  - Scott Matsumoto
  - Gary McGraw
  - Sammy Migues
  - Craig Miller
  - John Steven

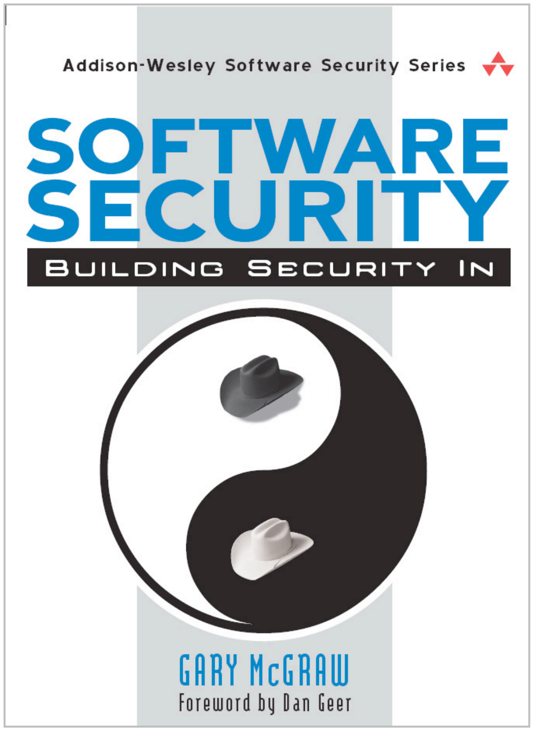
IEEE Security & Privacy Magazine + 2 Podcasts

- Building Security In
- Software Security Best Practices column edited by John Steven
- www.computer.org/security/bsisub/



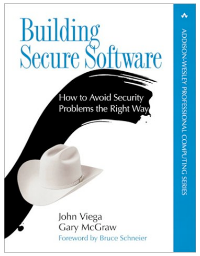


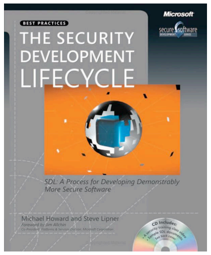
- www.cigital.com/silverbullet
- www.cigital.com/realitycheck

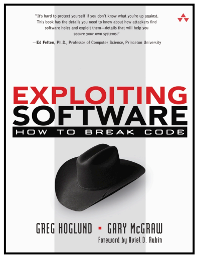


Software Security: the book

- How to DO software security
  - Best practices
  - Tools
  - Knowledge
- Cornerstone of the Addison-Wesley Software Security Series
- www.swsec.com

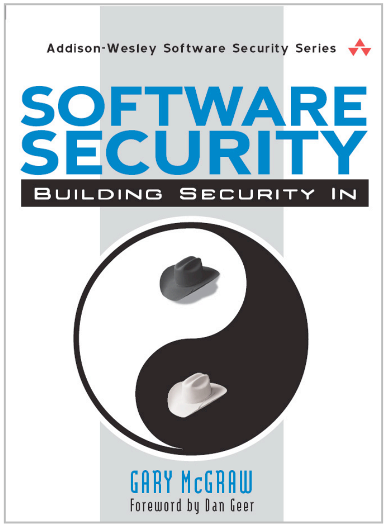

Building Secure Software
How to Avoid Security Problems the Right Way
John Viega
Gary McGraw
Foreword by Bruce Schneier
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES


BEST PRACTICES
Microsoft
THE SECURITY DEVELOPMENT LIFECYCLE
SDL: A Process for Developing Demonstrably More Secure Software
Michael Howard and Steve Lipner



Addison Wesley

- Cigital's Software Security Group invents and delivers Software Quality Management
- WE NEED GREAT PEOPLE
- See the Addison-Wesley Software Security series
- Send e-mail: gem@cigital.com



Addison-Wesley Software Security Series

SOFTWARE SECURITY

BUILDING SECURITY IN

GARY McGRAW

Foreword by Dan Geer

For more

cigital

"So now, when we face a choice between adding features and resolving security issues, we need to choose security."

-Bill Gates