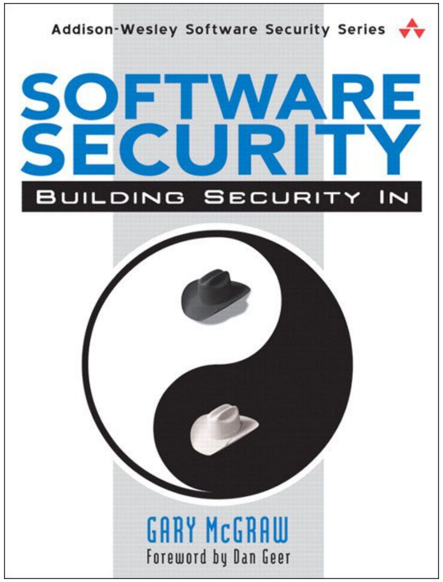
Bug Parades, Zombies, and the BSIMM:

A Decade of Software Security

Gary McGraw, Ph.D.
Chief Technology Officer, Cigital



Addison-Wesley Software Security Series

SOFTWARE SECURITY

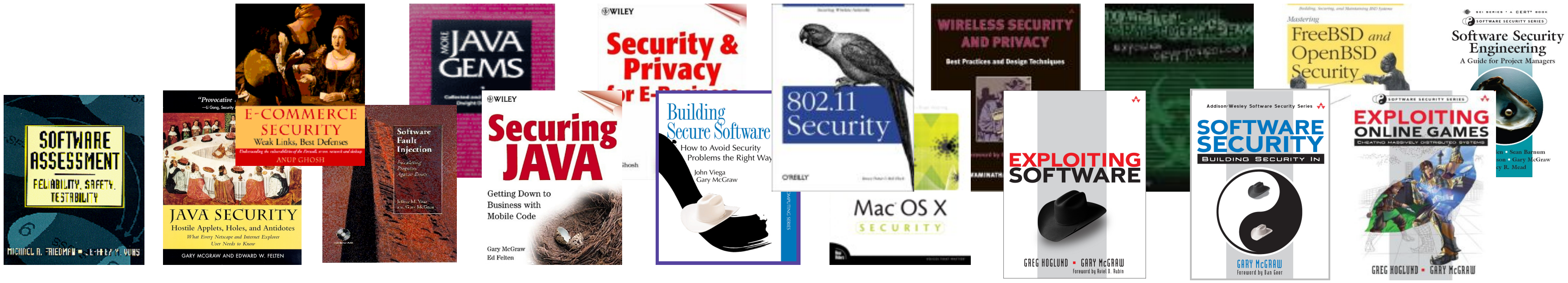
BUILDING SECURITY IN

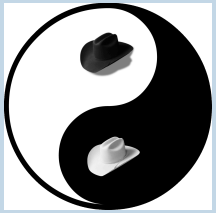
GARY McGRAW

Foreword by Dan Geer

Cigital

- Founded in 1992 to provide software security and software quality professional services
- Recognized experts in software security and software quality
  - Widely published in books, white papers, and articles
  - Industry thought leaders





in the beginning

software industry blooms in 1970s

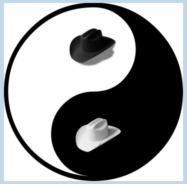
- IBM unbundles software and services from hardware in late 1960s
- Unbundling created inequality in system security
- Security shifts from consumers to producers

who should DO software security?

← Network security ops guys

NOBODY IN THE MIDDLE

Super rad developer dudes →

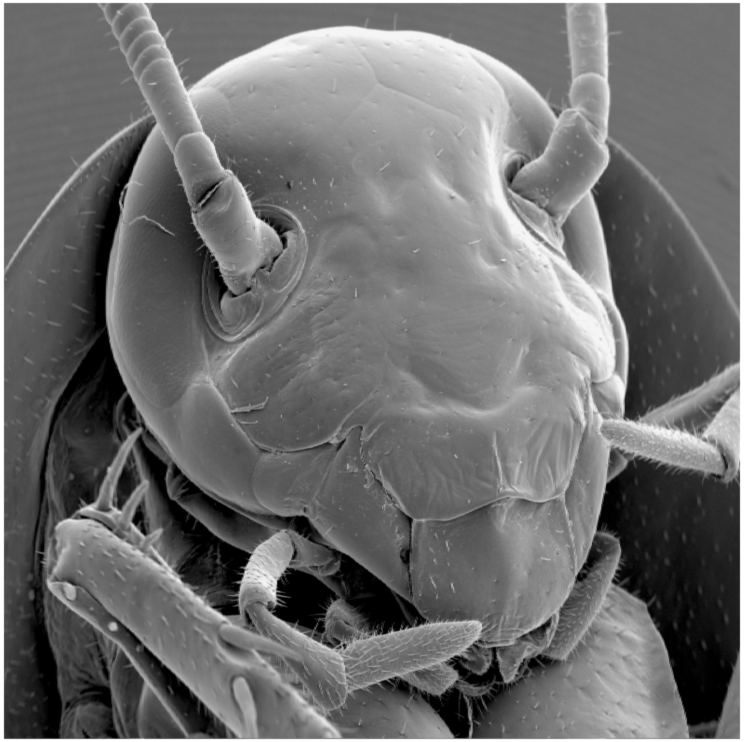


the bug parade

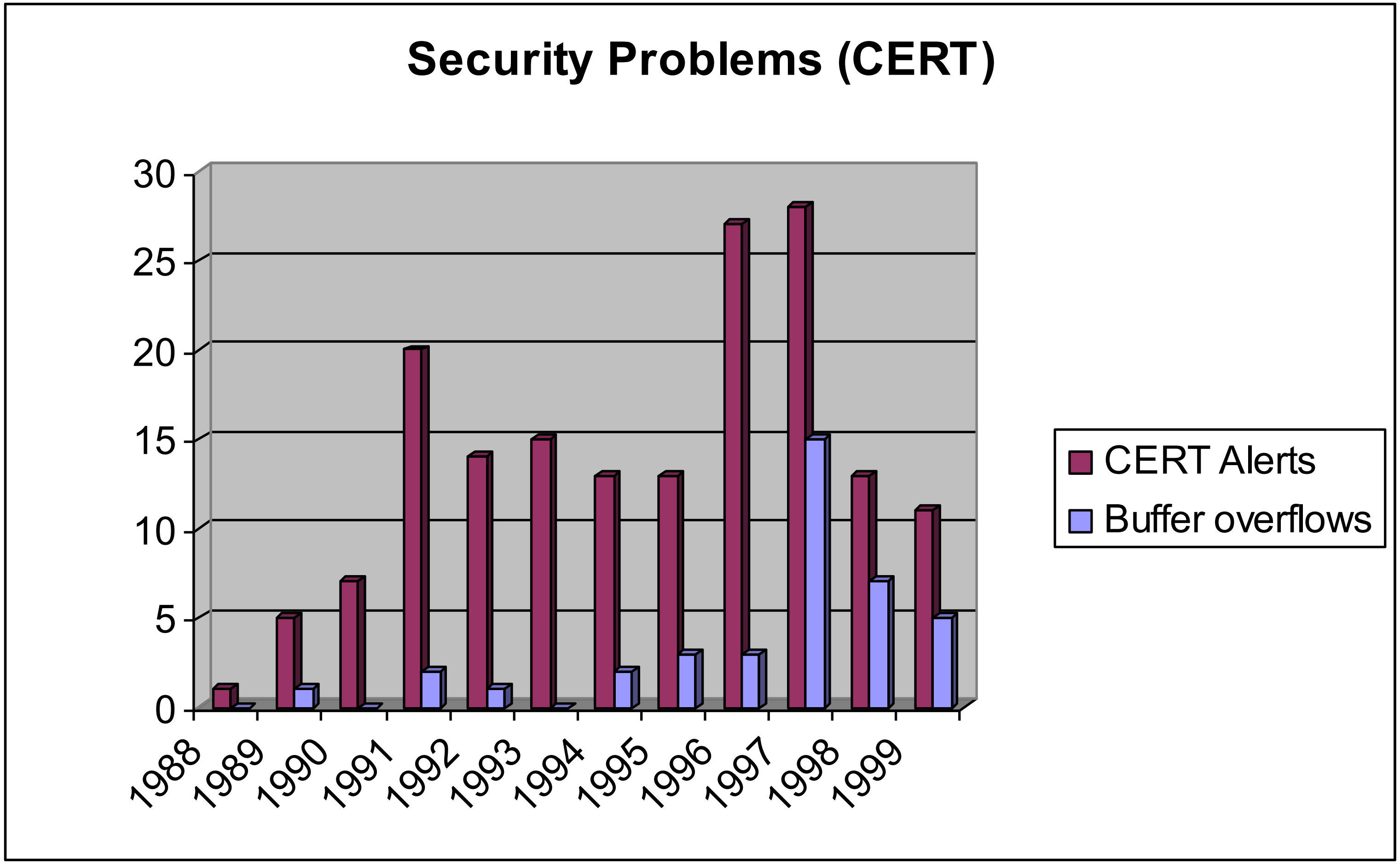
bug: the dreaded buffer overflow

- Overwriting the bounds of data objects
- Allocate some bytes, but the language doesn’t care if you try to use more
- char x[12]; x[12] = ‘\0’
- Why was this done? Efficiency!
  - (remember in the 70’s when code had to be tight?)

- The most pervasive security problem today in terms of reported bugs in the ‘90s



eleven years of CERT data



a classic error in C

```
void main() {
    char buf[1024];
    gets(buf);
}
```

- How not to get input
  - Attacker can send an infinite string!
  - Chapter 7 of K&R (page 164)

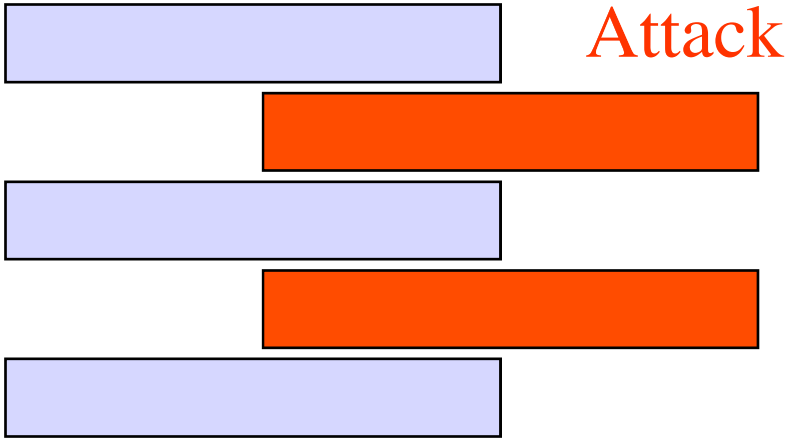
calls to avoid in C

- Very risky:
  - gets,strcpy,strcat,sprintf,scanf, sscanf,fscanf,vfscanf,vsprintf,vscanf, vsscanf,streadd,strecpy,realpath,syslog, getopt,getopt_long,getpass
- Risky:
  - strtrns,getchar,fgetc,getc,read
- Be wary:
  - bcopy,fgets,memcpy,snprintf, strccpy,strcadd,strncpy,vsnprintf

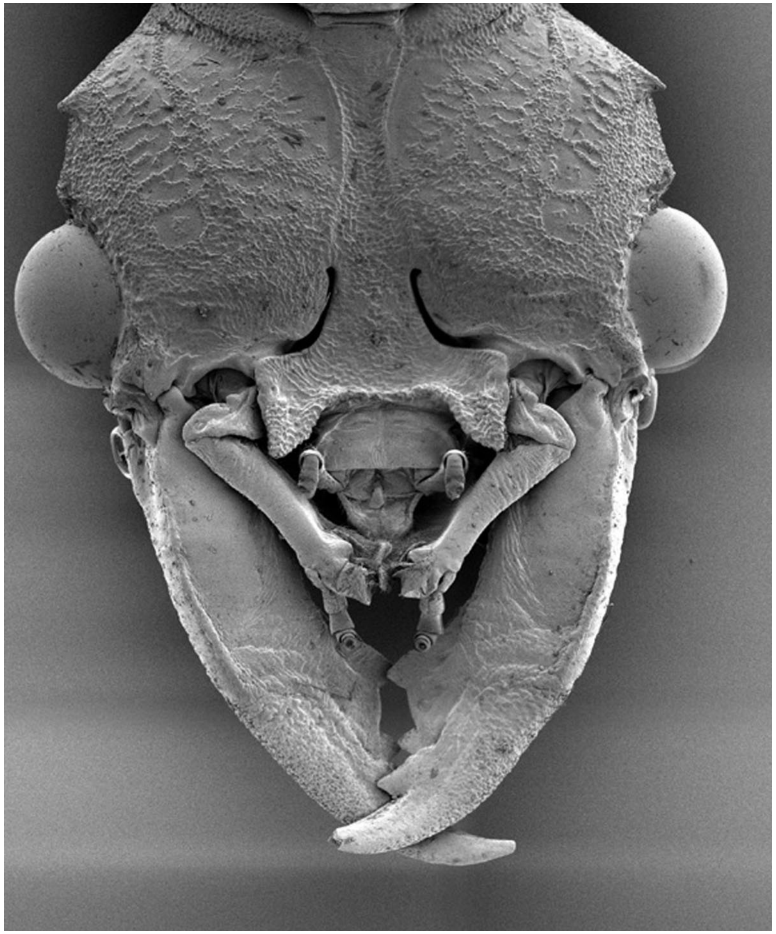
Big 1999 idea: Why not make a tool to find these for you??!

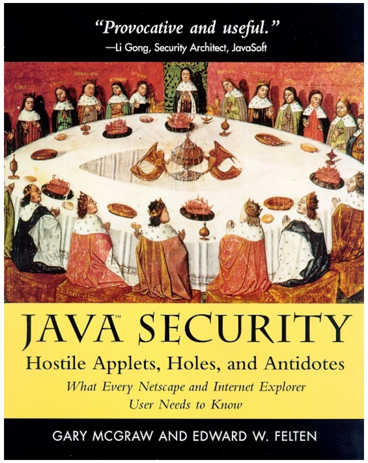
bug: race condition

- Time makes all the difference
- Atomic operations that are not atomic



bugs: Java security





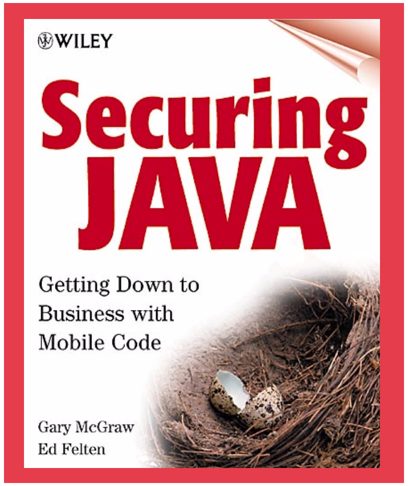
"Provocative and useful."
—Li Gong, Security Architect, JavaSoft

JAVA SECURITY

Hostile Applets, Holes, and Antidotes

What Every Netscape and Internet Explorer User Needs to Know

GARY MCGRAW AND EDWARD W. FELTEN



WILEY

Securing JAVA

Getting Down to Business with Mobile Code

Gary McGraw
Ed Felten

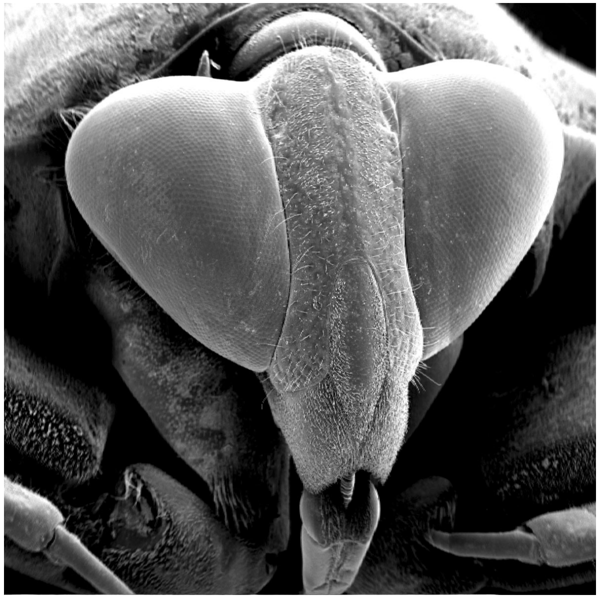
a chronology Java of attack applets

- February 96: DNS flaw in JDK 1.0.1
- March 96: Path name bug
- March 96: Princeton Class Loader bug
- May 96: type casting attack
- June 96: Array type implementation error
- July 96: More type casting problems
- August 96:Flaw in Microsoft’s Java VM
- February 97: Invasion of Privacy attack applets
- March 97: JVM hole
- April 97: Code signing flaw
- May 97: Verifier problems discovered in many VMs
- July 97: Vacuum bug
- August 97: redirect bug
- July 98: ClassLoader bug
- March 99: Verifier hole
- August 99: Race condition
- October 99: Verifier hole 2
- August 2000: Brown Orifice
- October 2000: ActiveX/Java

All of these bugs have been fixed.

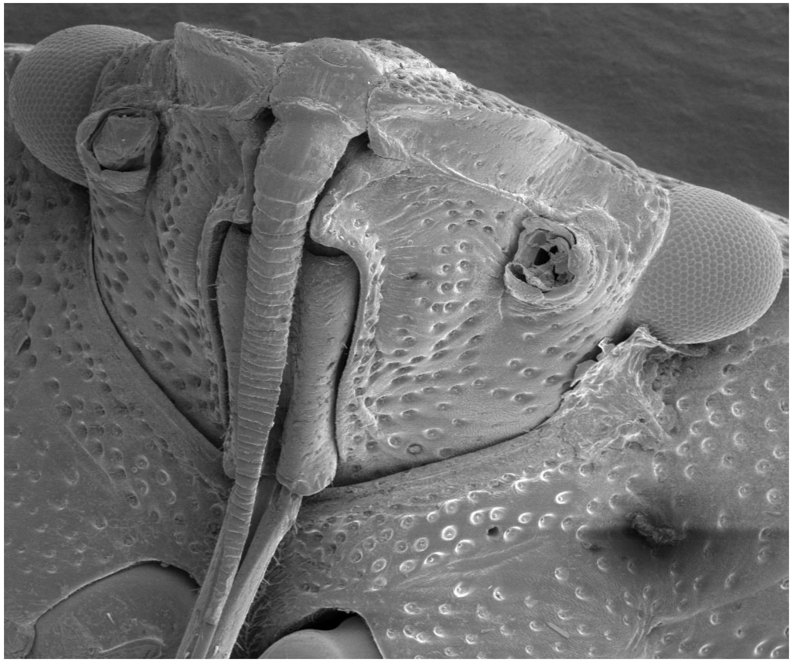
bug: SQL injection

- Enables an attacker to execute arbitrary SQL commands on back-end database
- Example:
  - PHP code inputs USERNAME and PASSWORD and passes to MySQL back-end
  - USERNAME is entered as bob
  - PASSWORD is entered as ’ or USERNAME= ‘bob
  - Back-end executes Select ID from USERS where USERNAME= ‘bob’ and PASSWORD= ‘’ or USERNAME= ‘bob’
  - Instead of Select ID from USERS where USERNAME= ‘bob’ and PASSWORD= ‘password’



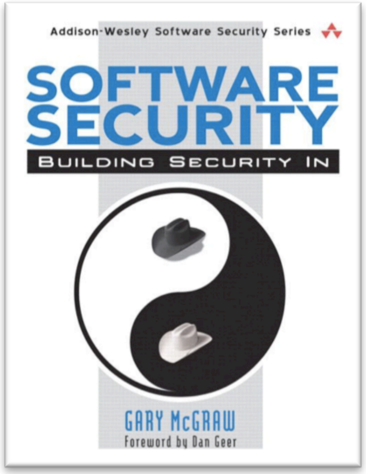
bug: XSS

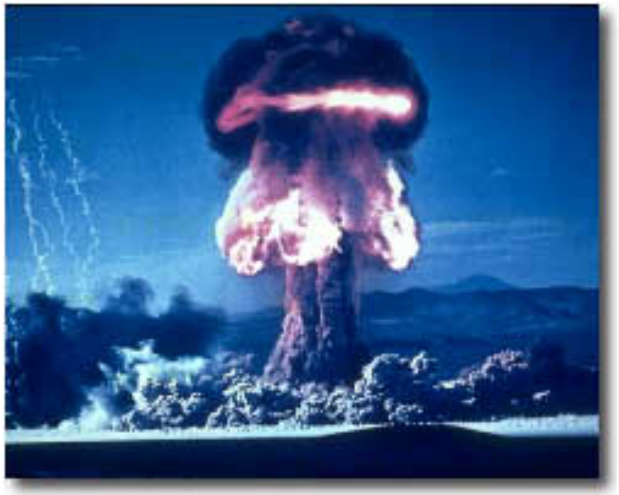
- Unaltered user-controlled content in a Web server response gives an attacker the opportunity to insert HTML and scripts
- This code gets rendered in a victim's browser
  - Reflected (malicious links)
  - Stored (by website)
- OWASP top ten bug



seven pernicious kingdoms (of bugs)

- Input validation and representation
- API abuse
- Security features
- Time and state
- Error handling
- Code quality
- Encapsulation
- Environment



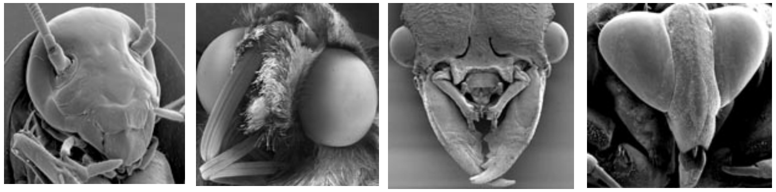


the bug parade FAIL

IMPLEMENTATION BUGS

50%

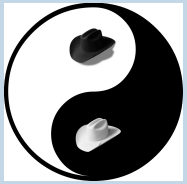
- Buffer overflow
  - String format
  - One-stage attacks
- Race conditions
  - TOCTOU (time of check to time of use)
- Unsafe environment variables
- Unsafe system calls
  - System()
- Untrusted input problems



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



software security zombies

zombie ideas need repeating

- Software security seems obvious to us, but it is still catching on
- The middle market is just beginning to emerge
- Time to scale!

ZOMBIE

- Network security FAIL
- More code more bugs
- SDLC integration
- Bugs and flaws
- Badness-ometers

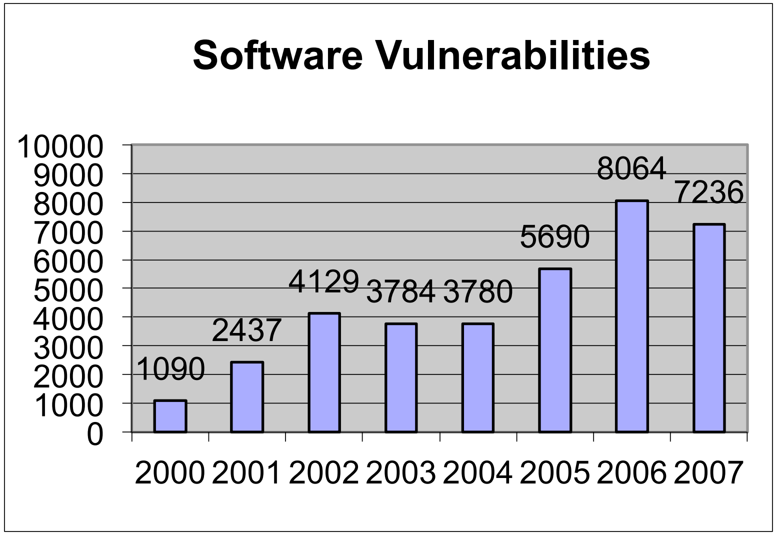
Experts in software security take things for granted. That's OK, but don't forget how far behind some firms are.

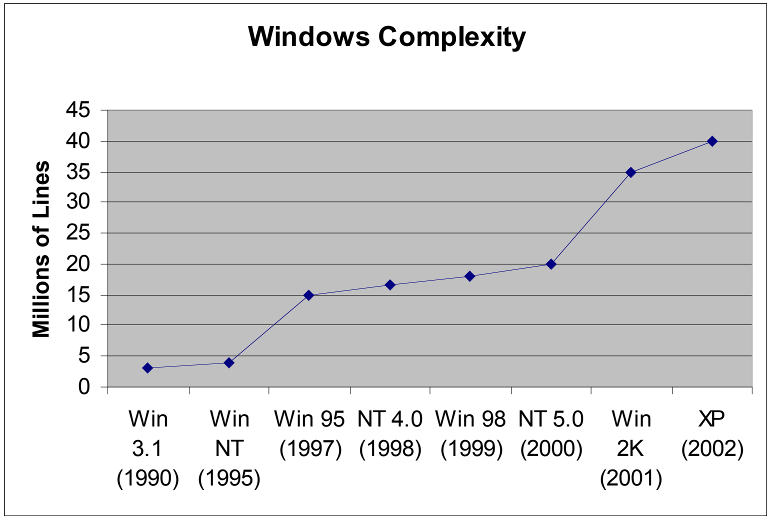
zombie: old school security is reactive

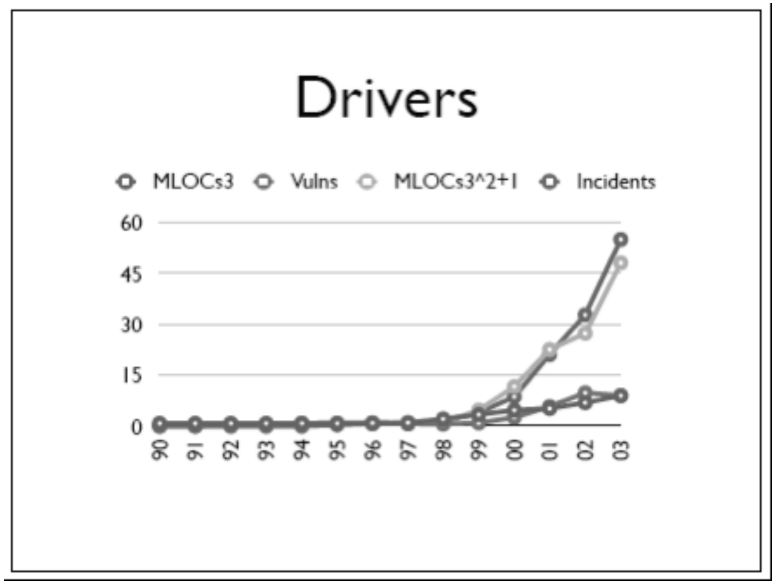
- Defend the “perimeter” with a firewall
  - To keep stuff out
- Promulgate “penetrate and patch”
- “Review” products when they’ re complete
  - Throw it over the wall testing
  - Too much weight on penetration testing
- Over-rely on security functions
  - “We use SSL”

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

zombie: more code, more bugs

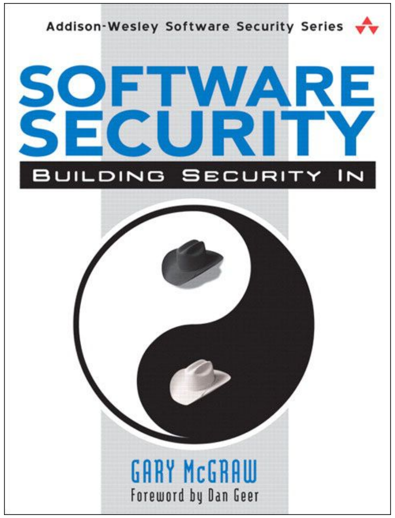


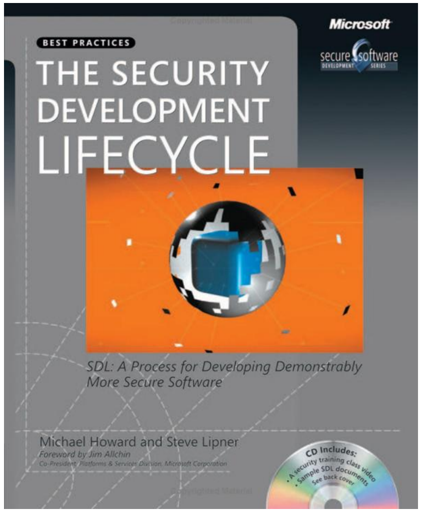




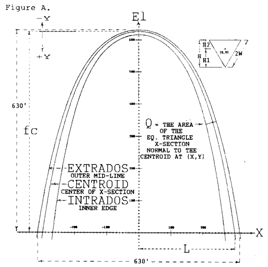
zombie: SDLC integration

- Integrating best practices into large organizations
  - Microsoft’s SDL
  - Cigital’s touchpoints
  - OWASP CLASP/SAMM



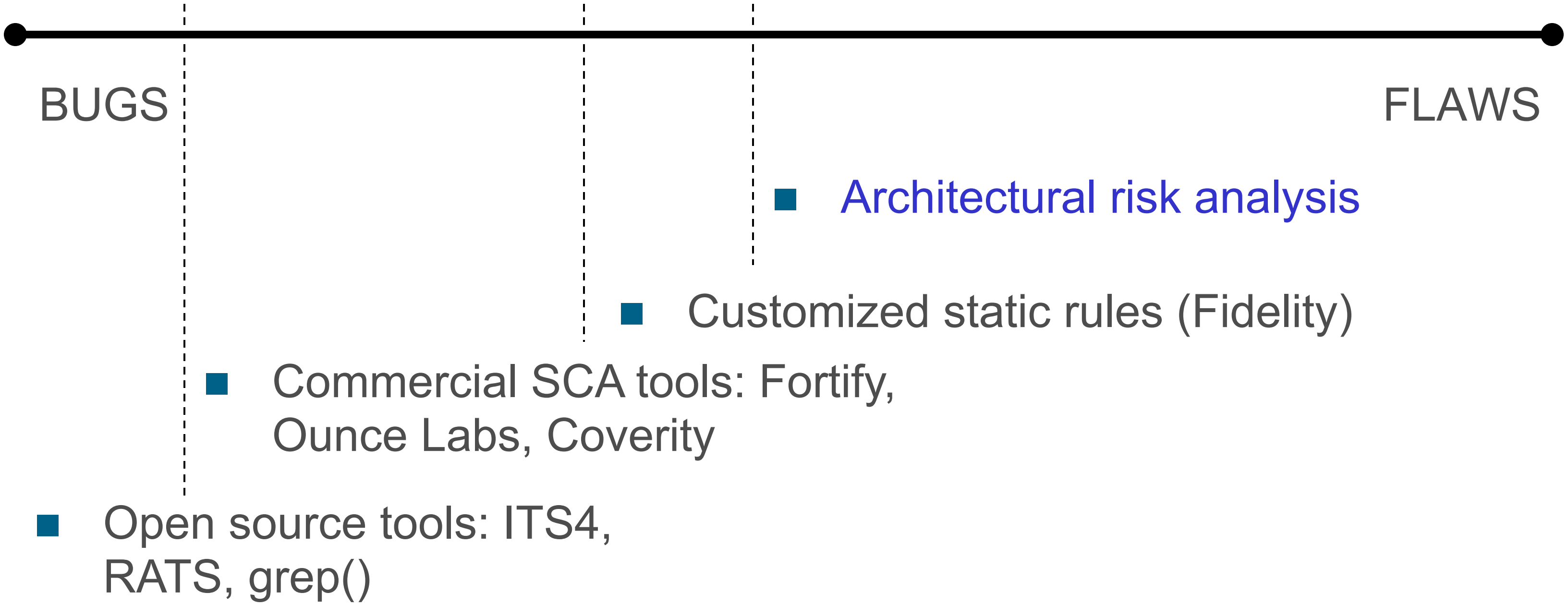

Microsoft
secure software
BEST PRACTICES
THE SECURITY DEVELOPMENT LIFECYCLE
SDL: A Process for Developing Demonstrably More Secure Software
Michael Howard and Steve Lipner
Foreword by Jim Allchin
CD Includes:

zombie: bugs AND flaws



gets()

attacker in the middle



BUGS

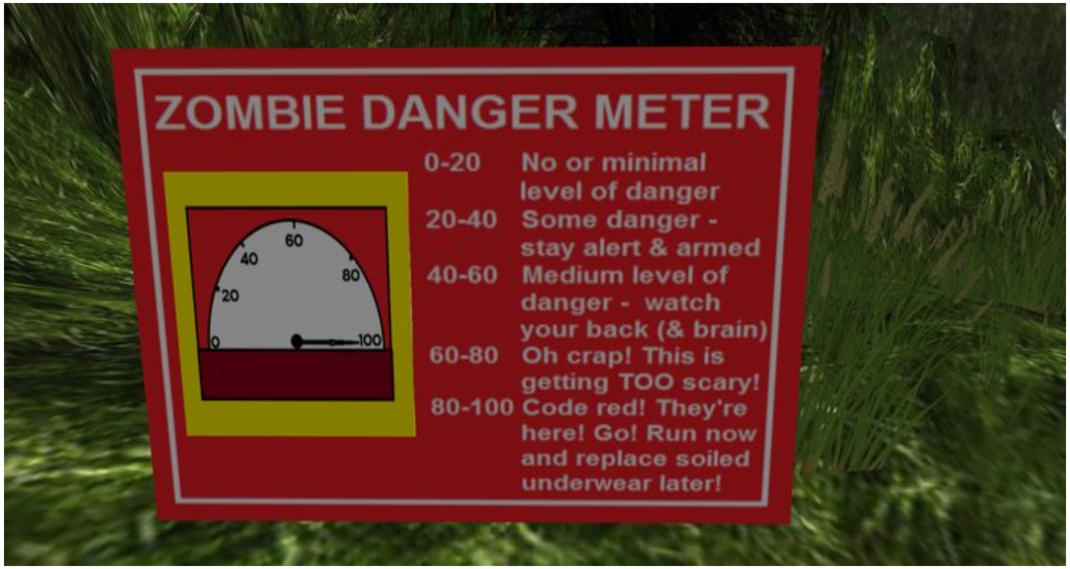
FLAWS

- Architectural risk analysis
- Customized static rules (Fidelity)
- Commercial SCA tools: Fortify, Ounce Labs, Coverity
- Open source tools: ITS4, RATS, grep()

zombie: badness-ometer

Deep Trouble
Who Knows

badness-ometer

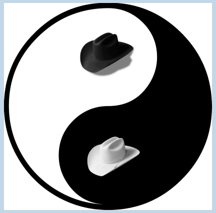


ZOMBIE DANGER METER

0-20	No or minimal level of danger
20-40	Some danger - stay alert & armed
40-60	Medium level of danger - watch your back (& brain)
60-80	Oh crap! This is getting TOO scary!
80-100	Code red! They're here! Go! Run now and replace soiled underwear later!

zombie baby: fix the dang software

- Software security and application security today are about finding bugs
- The time has come to stop looking for new bugs to add to the list
- Which bugs in this pile should I fix?



software security touchpoints

rise of the software security group

- Cigital SSG turned fifteen in 2012
- Microsoft adopts the Secure Development Lifecycle
- **Many** companies have a group devoted to software security

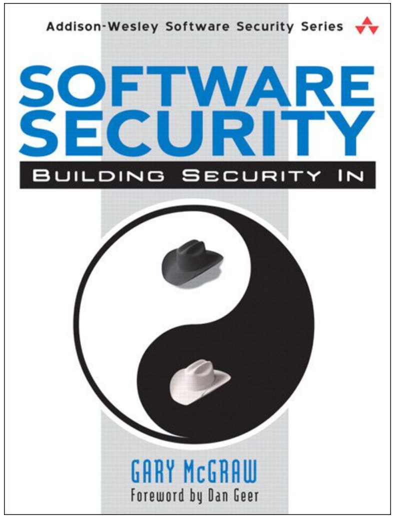
- microsoft
- dtcc
- emc
- fidelity
- adobe
- wells fargo
- goldman sachs
- google
- qualcomm
- morgan stanley
- usaf
- dell
- pershing
- the hartford
- barclays capital
- bank of tokyo
- ups
- bank of montreal
- sterling commerce
- time warner

- cisco
- bank of america
- walmart
- finra
- vanguard
- college board
- oracle
- state street
- omgeo
- motorola
- general electric
- lockheed martin
- intuit
- vmware
- amex
- bank of ny mellon
- harris bank
- paypal
- symantec

- visa europe
- thomson/reuters
- BP
- SAP
- nokia
- ebay
- mckesson
- ABN/amro
- ING
- telecom italia
- swift
- standard life
- cigna
- AON
- coke
- mastercard
- apple
- AOL
- CA

from philosophy to HOW TO circa 2006

- Integrating best practices into large organizations
  - Microsoft’s SDL
  - Cigital’s touchpoints
  - OWASP adopts CLASP



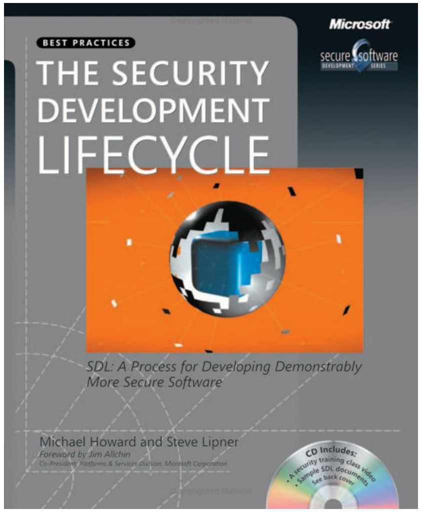
Addison-Wesley Software Security Series

SOFTWARE SECURITY

BUILDING SECURITY IN

GARY McGRAW

Foreword by Dan Geer



Microsoft

secure software development series

BEST PRACTICES

THE SECURITY DEVELOPMENT LIFECYCLE

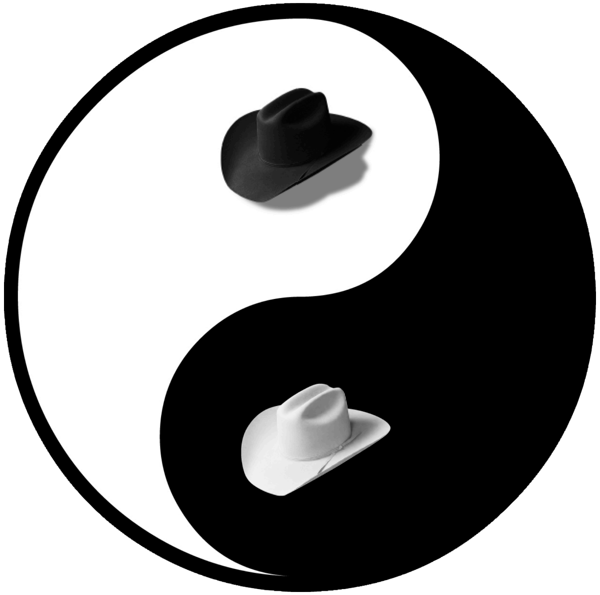
SDL: A Process for Developing Demonstrably More Secure Software

Michael Howard and Steve Lipner

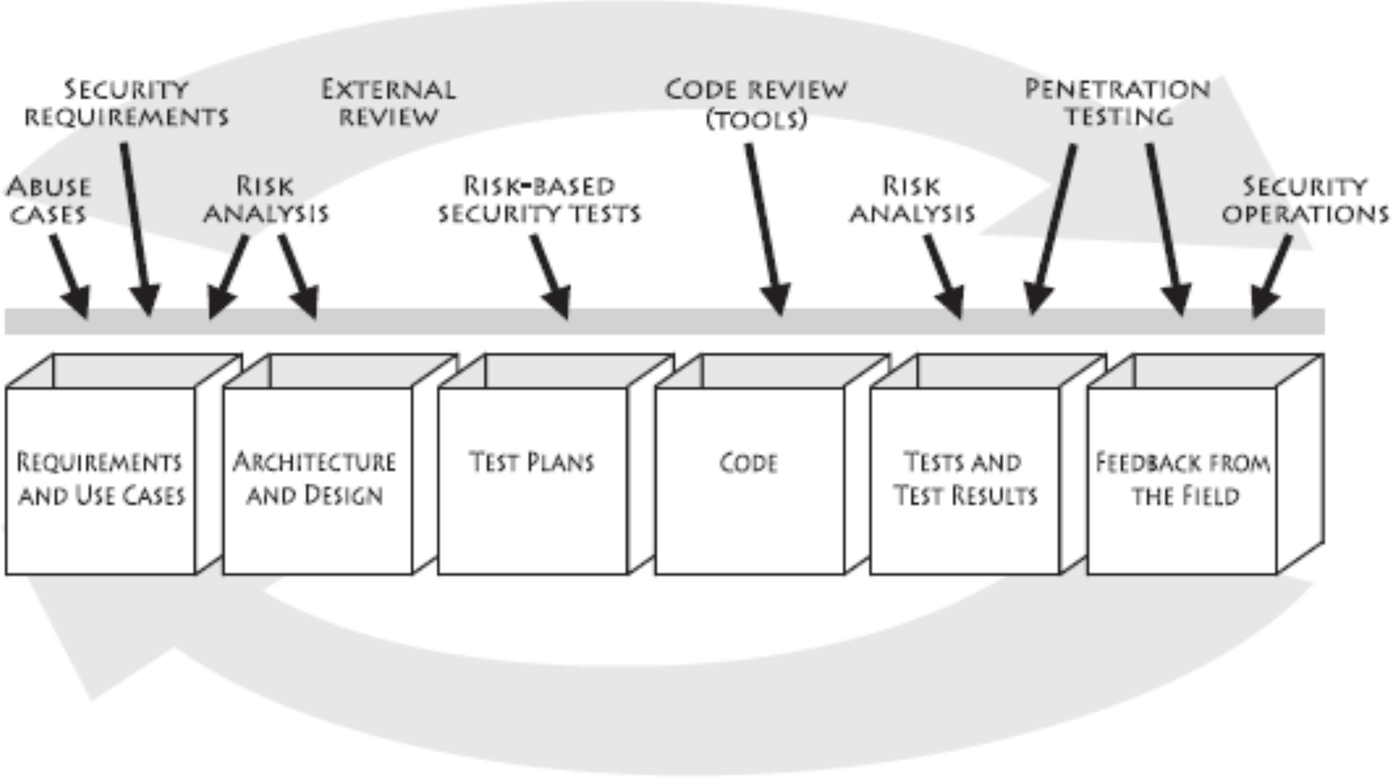
Foreword by Jim Allchin

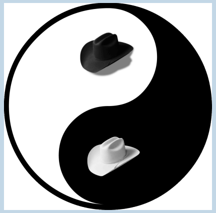
Co-President, Platforms & Services Division, Microsoft Corporation

CD Includes:
• A security training class video
• Sample SDL documents
See back cover



software security touchpoints





the BSIMM

BSIMM: software security measurement

- Real data from (42) real initiatives
- 81 measurements
- McGraw, Chess, & Migues

cigital

PlexLogic

FORTIFY

CSO

Minded security

VIRTUALFORGE we harden your software

Adobe

AON

Bank of America

Capital One

DTCC
The Depository Trust & Clearing Corporation

EMC²
where information lives

Google

FannieMae

Intel

Intuit

McKESSON
Empowering Healthcare

Microsoft

NOKIA
Connecting People

QUALCOMM

STANDARD LIFE

SAP

SallieMae

scrippsnetworks
interactive

Sony Ericsson

Symantec.

SWIFT

TELECOM
ITALIA

VISA

vmware

WELLS FARGO

zynga

THOMSON REUTERS

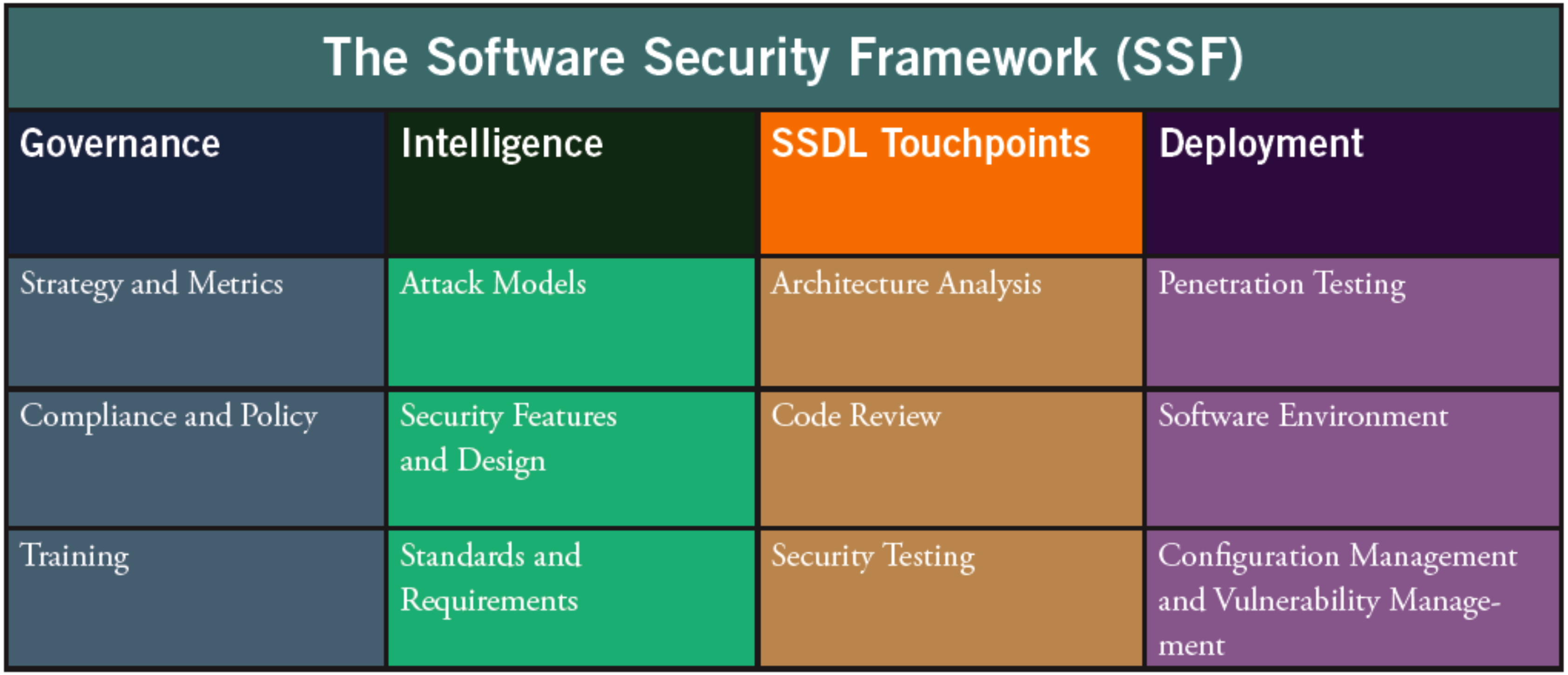
Fidelity
INVESTMENTS

+ 14 anonymous firms

monkeys eat bananas

- BSIMM is not about good or bad ways to eat bananas or banana best practices
- BSIMM is about observations
- BSIMM is descriptive, not prescriptive

software security framework

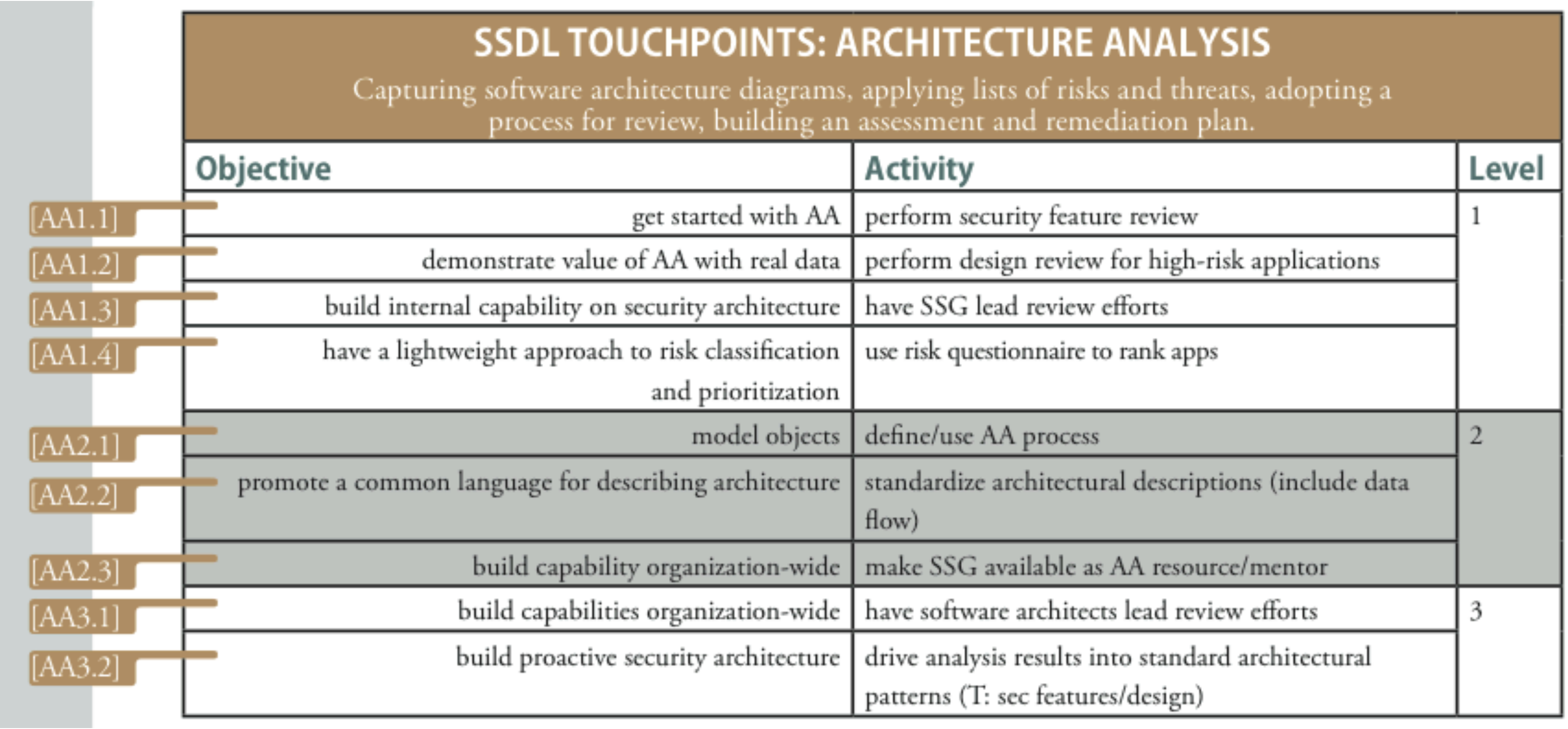


The Software Security Framework (SSF)

Governance	Intelligence	SSDL 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

- Twelve practices
- An “archeology grid”
- See informIT article at http://bsimm2.com

architecture analysis practice *skeleton*



SSDL TOUCHPOINTS: ARCHITECTURE ANALYSIS

Capturing software architecture diagrams, applying lists of risks and threats, adopting a process for review, building an assessment and remediation plan.

	Objective	Activity	Level
[AA1.1]	get started with AA	perform security feature review	1
[AA1.2]	demonstrate value of AA with real data	perform design review for high-risk applications	
[AA1.3]	build internal capability on security architecture	have SSG lead review efforts	
[AA1.4]	have a lightweight approach to risk classification and prioritization	use risk questionnaire to rank apps	
[AA2.1]	model objects	define/use AA process	2
[AA2.2]	promote a common language for describing architecture	standardize architectural descriptions (include data flow)	
[AA2.3]	build capability organization-wide	make SSG available as AA resource/mentor	
[AA3.1]	build capabilities organization-wide	have software architects lead review efforts	3
[AA3.2]	build proactive security architecture	drive analysis results into standard architectural patterns (T: sec features/design)	

example activity

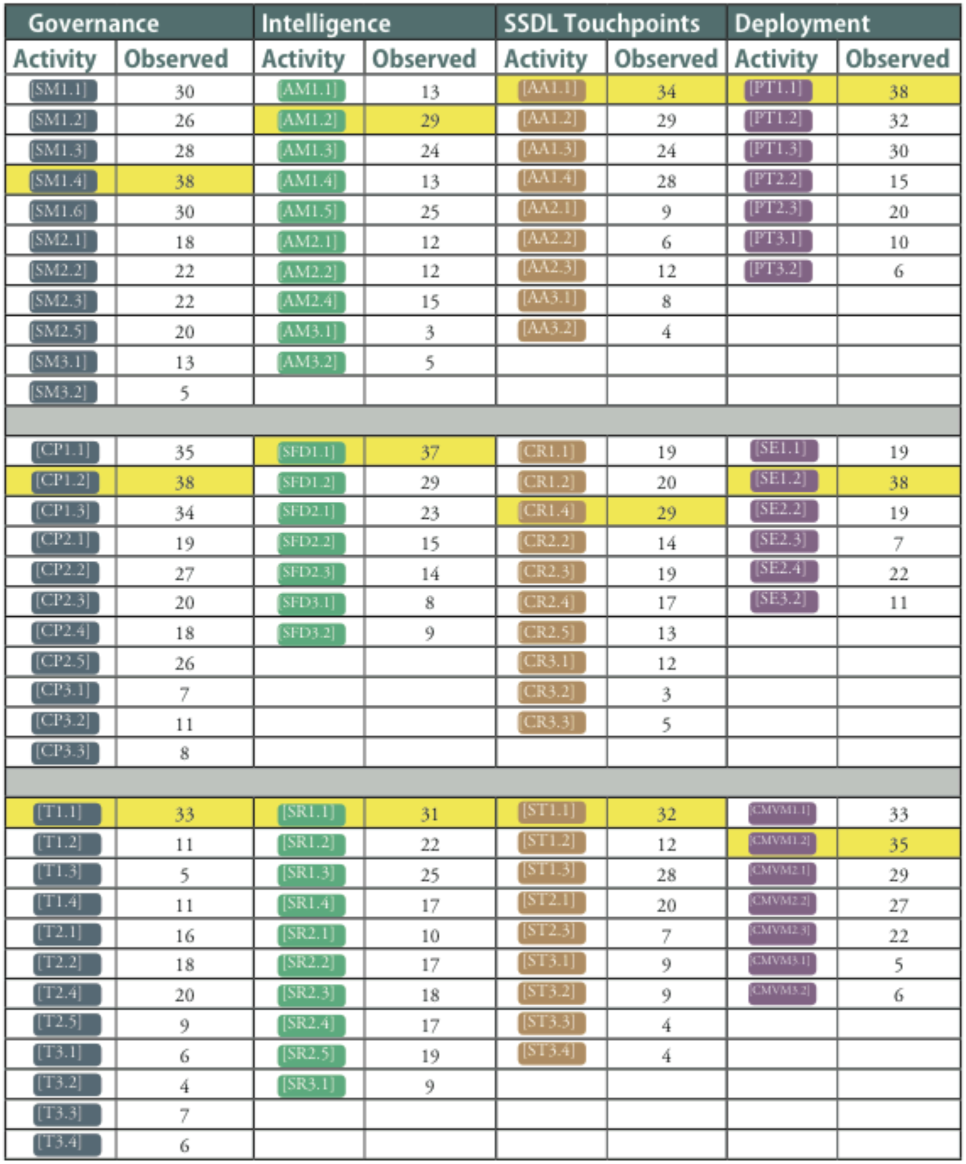
[AA1.2] **Perform design review for high-risk applications.** The organization learns about the benefits of architecture analysis by seeing real results for a few high-risk, high-profile applications. If the SSG is not yet equipped to perform an in-depth architecture analysis, it uses consultants to do this work. Ad hoc review paradigms that rely heavily on expertise may be used here, though in the long run they do not scale.

real-world data (42 firms)

- Initiative age
  - Average: 5.5 years
  - Newest: 1
  - Oldest: 16
  - Median: 4
- SSG size
  - Average: 19.2
  - Smallest: 0.5
  - Largest: 100
  - Median: 8
- Satellite size
  - Average: 42.7
  - Smallest: 0
  - Largest: 350
  - Median: 15
- Dev size
  - Average: 5183
  - Smallest: 11
  - Largest: 30,000
  - Median: 1675

Average SSG size: 1.99% of dev group size

BSIMM3 scorecard

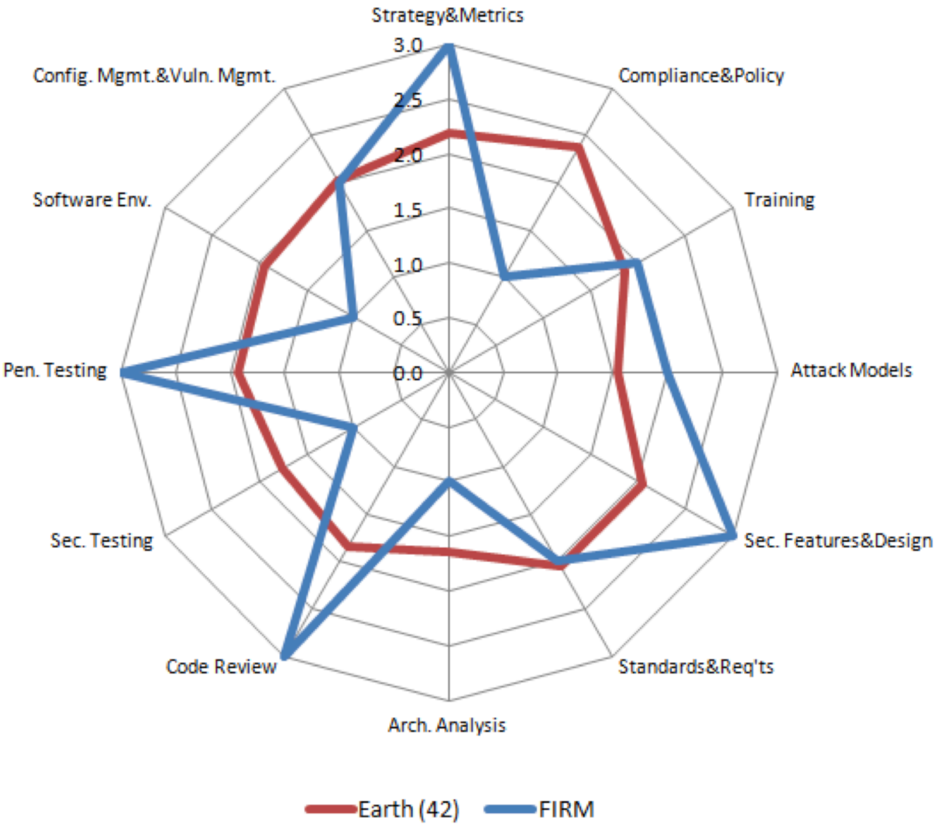


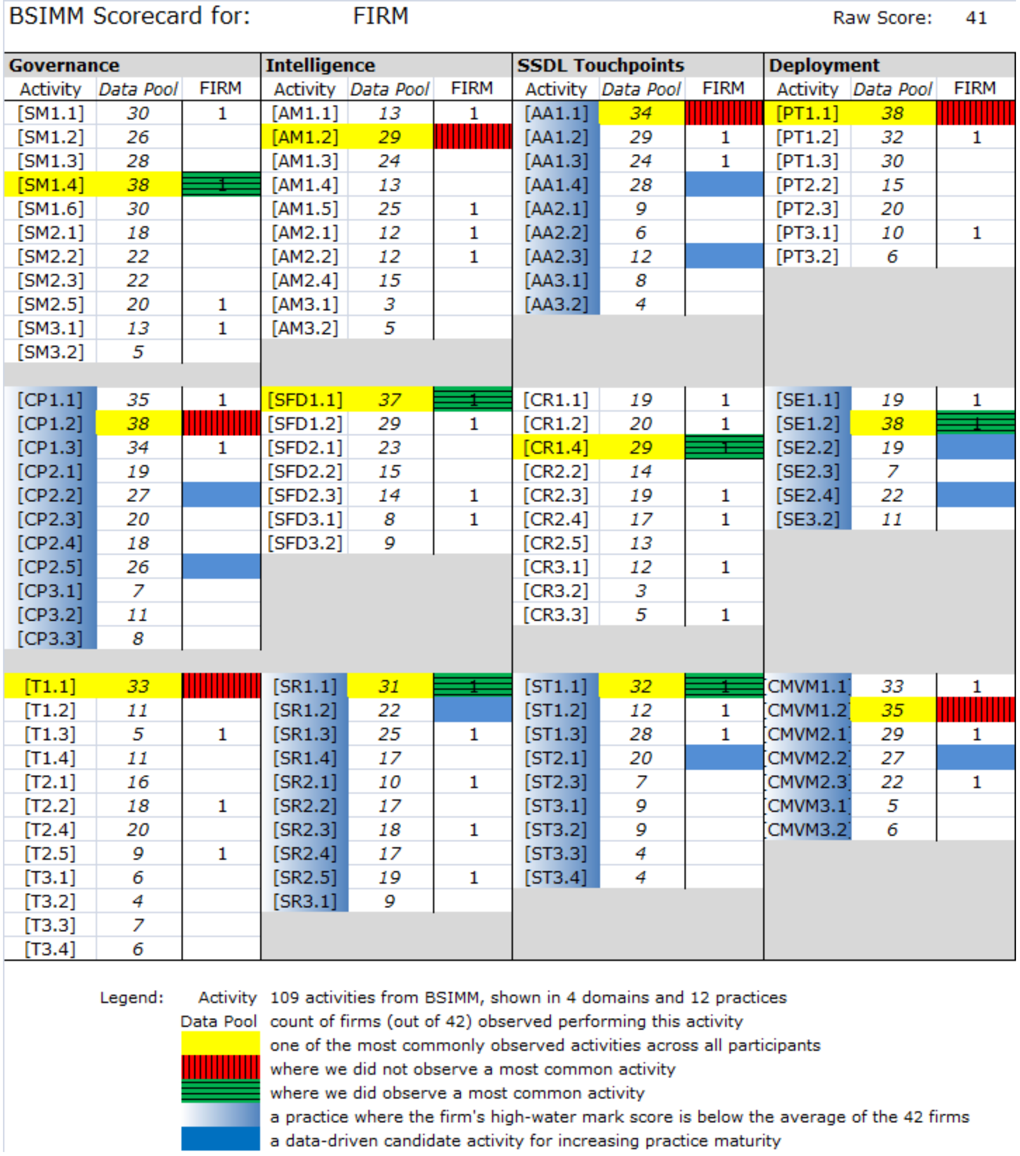
Governance		Intelligence		SSDL Touchpoints		Deployment	
Activity	Observed	Activity	Observed	Activity	Observed	Activity	Observed
[SM1.1]	30	[AM1.1]	13	[AA1.1]	34	[PT1.1]	38
[SM1.2]	26	[AM1.2]	29	[AA1.2]	29	[PT1.2]	32
[SM1.3]	28	[AM1.3]	24	[AA1.3]	24	[PT1.3]	30
[SM1.4]	38	[AM1.4]	13	[AA1.4]	28	[PT2.2]	15
[SM1.6]	30	[AM1.5]	25	[AA2.1]	9	[PT2.3]	20
[SM2.1]	18	[AM2.1]	12	[AA2.2]	6	[PT3.1]	10
[SM2.2]	22	[AM2.2]	12	[AA2.3]	12	[PT3.2]	6
[SM2.3]	22	[AM2.4]	15	[AA3.1]	8		
[SM2.5]	20	[AM3.1]	3	[AA3.2]	4		
[SM3.1]	13	[AM3.2]	5				
[SM3.2]	5						
[CP1.1]	35	[SFD1.1]	37	[CR1.1]	19	[SE1.1]	19
[CP1.2]	38	[SFD1.2]	29	[CR1.2]	20	[SE1.2]	38
[CP1.3]	34	[SFD2.1]	23	[CR1.4]	29	[SE2.2]	19
[CP2.1]	19	[SFD2.2]	15	[CR2.2]	14	[SE2.3]	7
[CP2.2]	27	[SFD2.3]	14	[CR2.3]	19	[SE2.4]	22
[CP2.3]	20	[SFD3.1]	8	[CR2.4]	17	[SE3.2]	11
[CP2.4]	18	[SFD3.2]	9	[CR2.5]	13		
[CP2.5]	26			[CR3.1]	12		
[CP3.1]	7			[CR3.2]	3		
[CP3.2]	11			[CR3.3]	5		
[CP3.3]	8						
[T1.1]	33	[SR1.1]	31	[ST1.1]	32	[CMVM1.1]	33
[T1.2]	11	[SR1.2]	22	[ST1.2]	12	[CMVM1.2]	35
[T1.3]	5	[SR1.3]	25	[ST1.3]	28	[CMVM2.1]	29
[T1.4]	11	[SR1.4]	17	[ST2.1]	20	[CMVM2.2]	27
[T2.1]	16	[SR2.1]	10	[ST2.3]	7	[CMVM2.3]	22
[T2.2]	18	[SR2.2]	17	[ST3.1]	9	[CMVM3.1]	5
[T2.4]	20	[SR2.3]	18	[ST3.2]	9	[CMVM3.2]	6
[T2.5]	9	[SR2.4]	17	[ST3.3]	4		
[T3.1]	6	[SR2.5]	19	[ST3.4]	4		
[T3.2]	4	[SR3.1]	9				
[T3.3]	7						
[T3.4]	6						

- 109 Activities
- 3 levels
- Top 12 activities
  - 69% cutoff
  - 29 of 42 firms
- Comparing scorecards between releases is interesting

BSIMM3 as a measuring stick

- Compare a firm with peers using the high water mark view
- Descriptive (not prescriptive)
- Incredible insight for planning





BSIMM Scorecard for: FIRM

Raw Score: 41

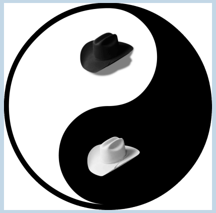
Governance			Intelligence			SSDL Touchpoints			Deployment		
Activity	*Data Pool*	FIRM	Activity	*Data Pool*	FIRM	Activity	*Data Pool*	FIRM	Activity	*Data Pool*	FIRM
[SM1.1]	*30*	1	[AM1.1]	*13*	1	[AA1.1]	*34*	🟥	[PT1.1]	*38*	🟥
[SM1.2]	*26*		[AM1.2]	*29*	🟥	[AA1.2]	*29*	1	[PT1.2]	*32*	1
[SM1.3]	*28*		[AM1.3]	*24*		[AA1.3]	*24*	1	[PT1.3]	*30*	
[SM1.4]	*38*	🟩 1	[AM1.4]	*13*		[AA1.4]	*28*	🟦	[PT2.2]	*15*	
[SM1.6]	*30*		[AM1.5]	*25*	1	[AA2.1]	*9*		[PT2.3]	*20*	
[SM2.1]	*18*		[AM2.1]	*12*	1	[AA2.2]	*6*		[PT3.1]	*10*	1
[SM2.2]	*22*		[AM2.2]	*12*	1	[AA2.3]	*12*	🟦	[PT3.2]	*6*	
[SM2.3]	*22*		[AM2.4]	*15*		[AA3.1]	*8*				
[SM2.5]	*20*	1	[AM3.1]	*3*		[AA3.2]	*4*				
[SM3.1]	*13*	1	[AM3.2]	*5*							
[SM3.2]	*5*										
[CP1.1]	*35*	1	[SFD1.1]	*37*	🟩 1	[CR1.1]	*19*	1	[SE1.1]	*19*	1
[CP1.2]	*38*	🟥	[SFD1.2]	*29*	1	[CR1.2]	*20*	1	[SE1.2]	*38*	🟩 1
[CP1.3]	*34*	1	[SFD2.1]	*23*		[CR1.4]	*29*	🟩 1	[SE2.2]	*19*	🟦
[CP2.1]	*19*		[SFD2.2]	*15*		[CR2.2]	*14*		[SE2.3]	*7*	
[CP2.2]	*27*	🟦	[SFD2.3]	*14*	1	[CR2.3]	*19*	1	[SE2.4]	*22*	🟦
[CP2.3]	*20*		[SFD3.1]	*8*	1	[CR2.4]	*17*	1	[SE3.2]	*11*	
[CP2.4]	*18*		[SFD3.2]	*9*		[CR2.5]	*13*				
[CP2.5]	*26*	🟦				[CR3.1]	*12*	1			
[CP3.1]	*7*					[CR3.2]	*3*				
[CP3.2]	*11*					[CR3.3]	*5*	1			
[CP3.3]	*8*										
[T1.1]	*33*	🟥	[SR1.1]	*31*	🟩 1	[ST1.1]	*32*	🟩 1	CMVM1.1	*33*	1
[T1.2]	*11*		[SR1.2]	*22*	🟦	[ST1.2]	*12*	1	CMVM1.2	*35*	🟥
[T1.3]	*5*	1	[SR1.3]	*25*	1	[ST1.3]	*28*	1	CMVM2.1	*29*	1
[T1.4]	*11*		[SR1.4]	*17*		[ST2.1]	*20*	🟦	CMVM2.2	*27*	🟦
[T2.1]	*16*		[SR2.1]	*10*	1	[ST2.3]	*7*		CMVM2.3	*22*	1
[T2.2]	*18*	1	[SR2.2]	*17*		[ST3.1]	*9*		CMVM3.1	*5*	
[T2.4]	*20*		[SR2.3]	*18*	1	[ST3.2]	*9*		CMVM3.2	*6*	
[T2.5]	*9*	1	[SR2.4]	*17*		[ST3.3]	*4*				
[T3.1]	*6*		[SR2.5]	*19*	1	[ST3.4]	*4*				
[T3.2]	*4*		[SR3.1]	*9*							
[T3.3]	*7*										
[T3.4]	*6*										

Legend:		
	Activity	109 activities from BSIMM, shown in 4 domains and 12 practices
	Data Pool	count of firms (out of 42) observed performing this activity
	🟨	one of the most commonly observed activities across all participants
	🟥	where we did not observe a most common activity
	🟩	where we did observe a most common activity
	▨	a practice where the firm's high-water mark score is below the average of the 42 firms
	🟦	a data-driven candidate activity for increasing practice maturity

- Top 12 activities
  - green = good?
  - red = bad?
- "Blue shift" practices to emphasize
  - activities you should maybe think about in blue

BSIMM3 to BSIMM4

- BSIMM3 released September 2011 under creative commons
  - http://bsimm.com
  - Italian and German translations
  - BSIMM is a yardstick
  - Use it to see where you stand
  - Use it to figure out what your peers do
- BSIMM3→BSIMM4
  - BSIMM is growing
  - Target of 50 firms/100 measures



where to learn more

SearchSecurity & justice league

SearchSecurity

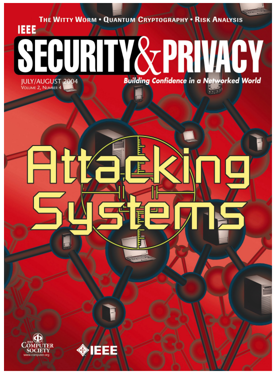
- www.searchsecurity.com
- No-nonsense monthly security column by Gary McGraw debuts in April
- www.cigital.com/~gem/writing

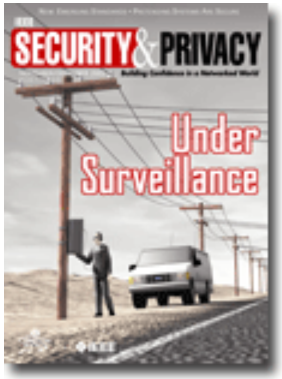
- www.cigital.com/justiceleague
- In-depth thought leadership blog from the Cigital Principals
  - Scott Matsumoto
  - Gary McGraw
  - Sammy Migues
  - Craig Miller
  - John Steven

IEEE security & privacy + silver bullet

- www.cigital.com/silverbullet

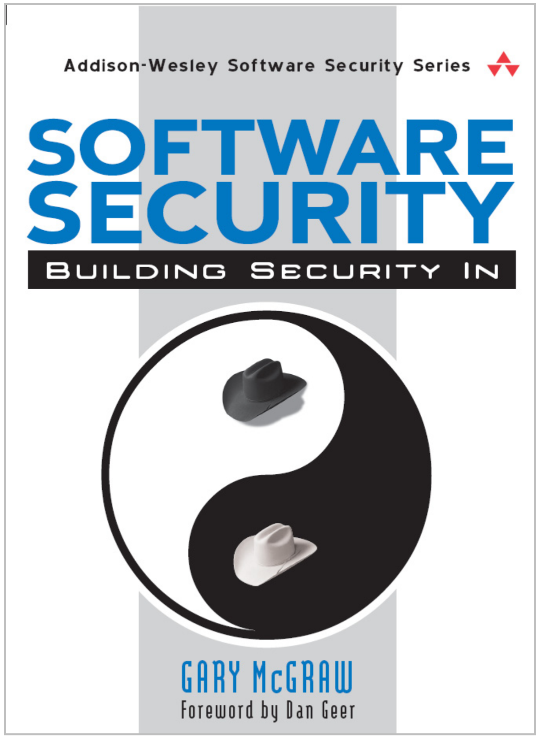
- Building Security In
- Software Security Best Practices column edited by John Steven
- www.computer.org/security/bsisub/





cigital

Software Security: the book



Addison-Wesley Software Security Series

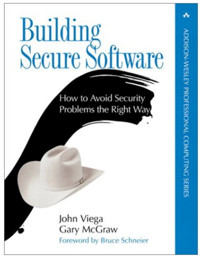
SOFTWARE SECURITY

BUILDING SECURITY IN

GARY McGRAW

Foreword by Dan Geer

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

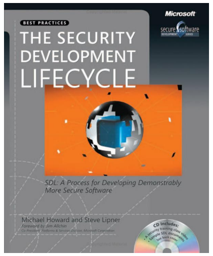


Building Secure Software

How to Avoid Security Problems the Right Way

John Viega
Gary McGraw
Foreword by Bruce Schneier

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES



BEST PRACTICES

THE SECURITY DEVELOPMENT LIFECYCLE

Microsoft

secure software

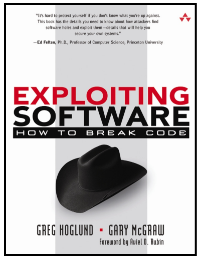
SDL: A Process for Developing Demonstrably More Secure Software

Michael Howard and Steve Lipner

Foreword by Jim Allchin

CD Includes:

WRITING SECURE CODE 2



"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

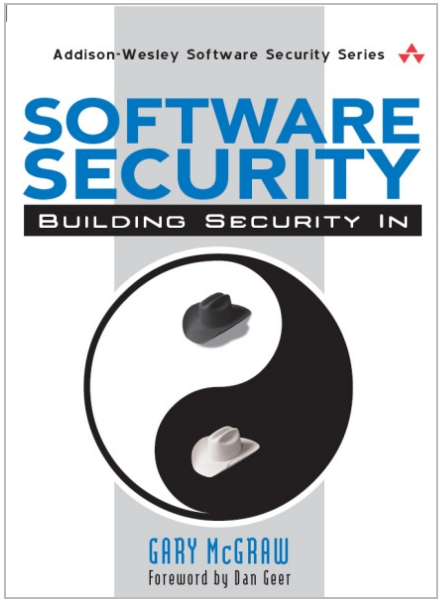
Foreword by Aviel D. Rubin

Addison Wesley

Build Security In

- http://bsimm.com
- WE NEED GREAT PEOPLE
- See the Addison-Wesley Software Security series
- Send e-mail: gem@cigital.com

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



cigital

Addison Wesley