Security in knowledge

BUG PARADES, ZOMBIES, AND THE BSIMM:

A DECADE OF SOFTWARE SECURITY

Dr. Gary McGraw (@cigitalgem)

CTO, Cigital

RSACONFERENCE
EUROPE 2013

Session ID: ADS-T07

Session Classification: Advanced

IN THE BEGINNING

Software industry blooms in the 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

Security in knowledge

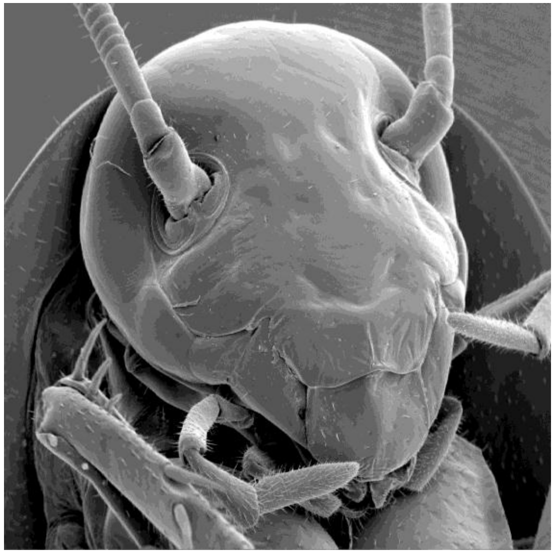
#RSAC

RSA® CONFERENCE EUROPE 2013

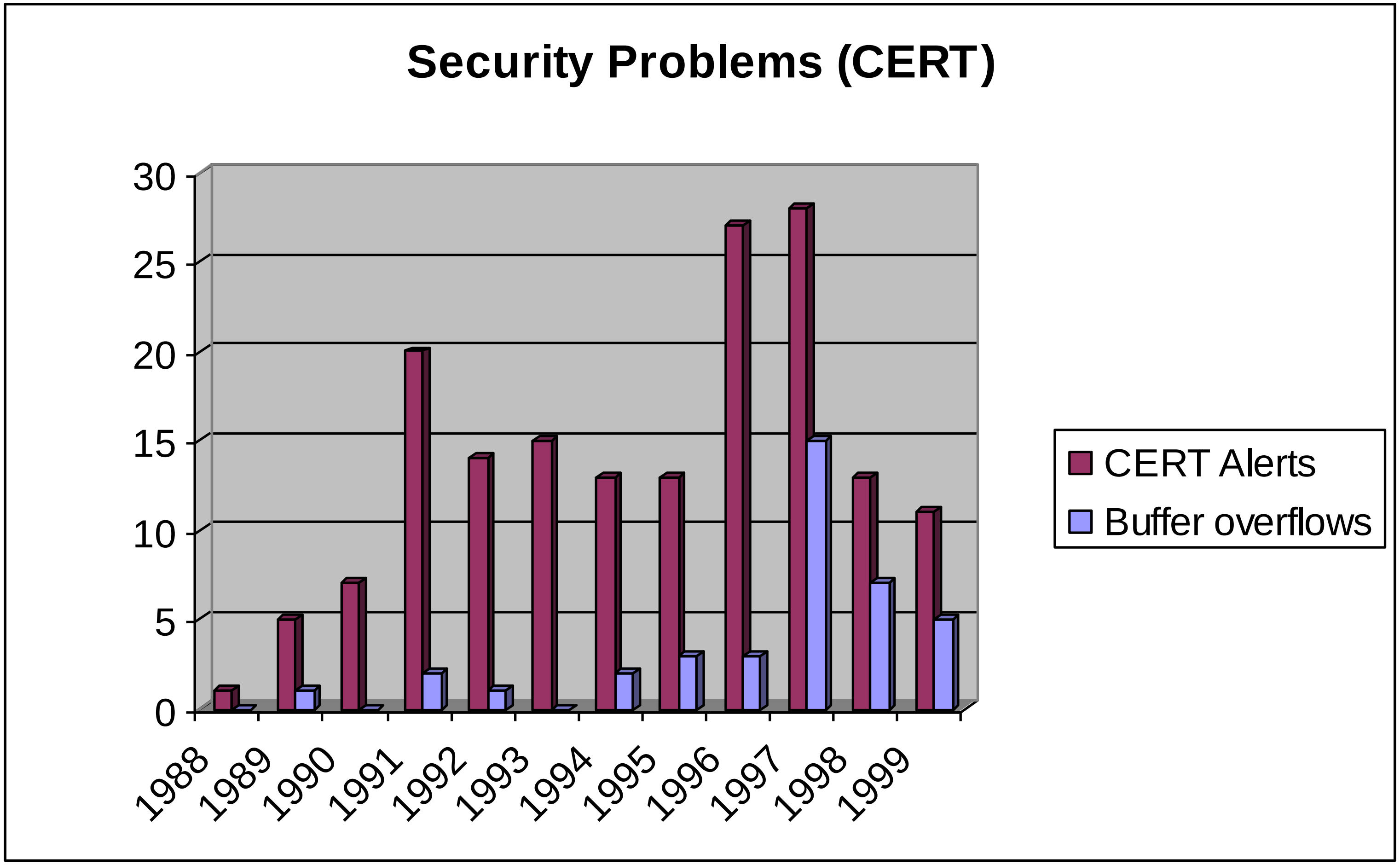
Bug: The dreaded buffer overflow

- 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



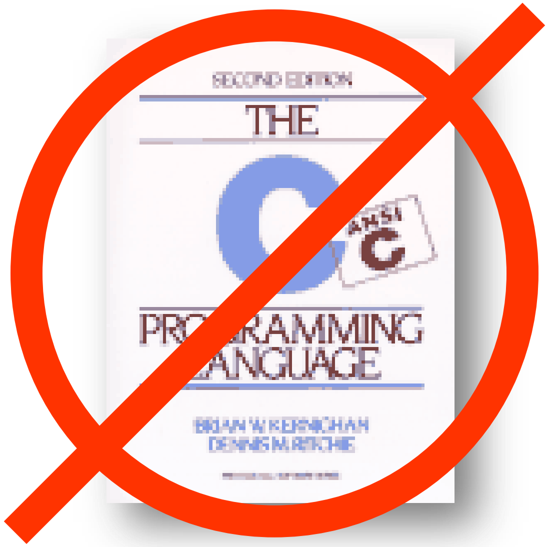
Eleven years of CERT data



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

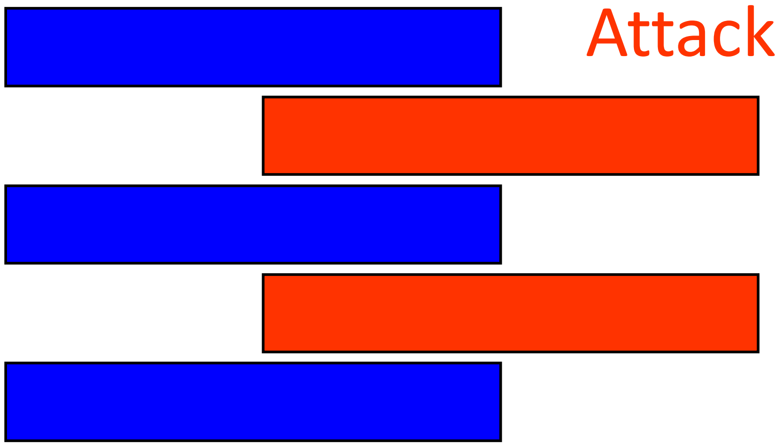
- **Be wary:**

  bcopy,fgets,memcpy,snprintf, strccpy,strcadd,strncpy,vsnprintf

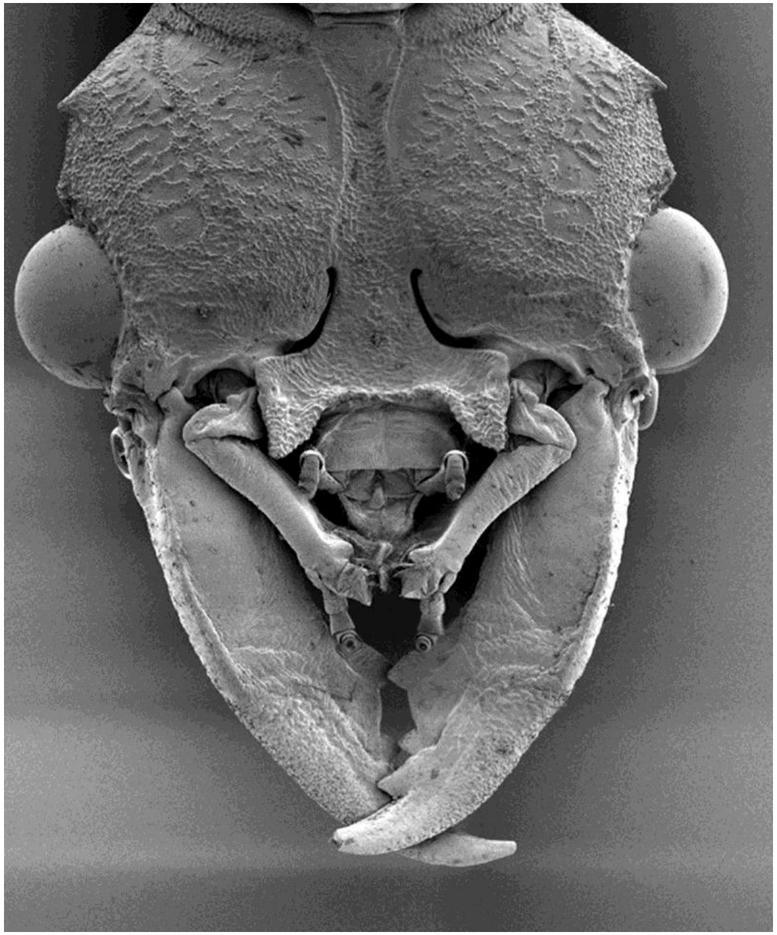
Big 1999 idea: Why not make a tool to find these for you??!

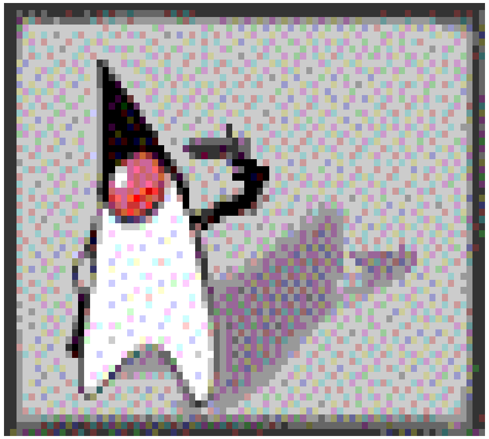
Bug: Race condition

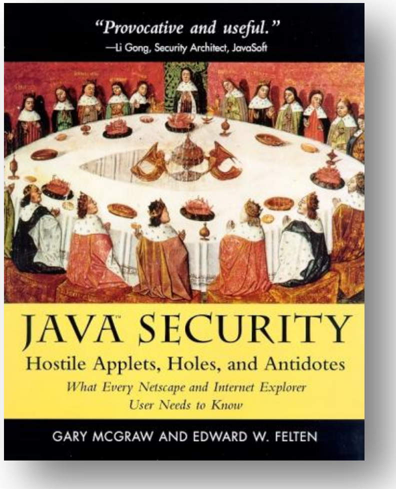
- Time makes all the difference
- Atomic operations that are not atomic



Bug: Java security







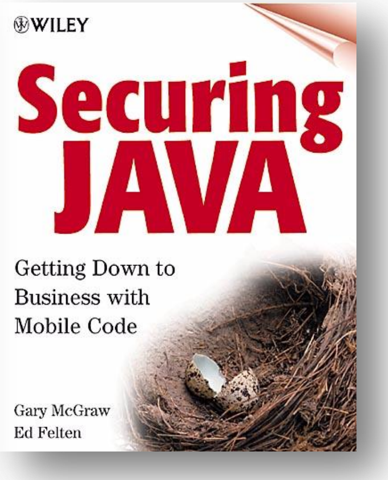
"Provocative and useful."
—Li Gong, Security Architect, JavaSoft

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 of Java attack applets

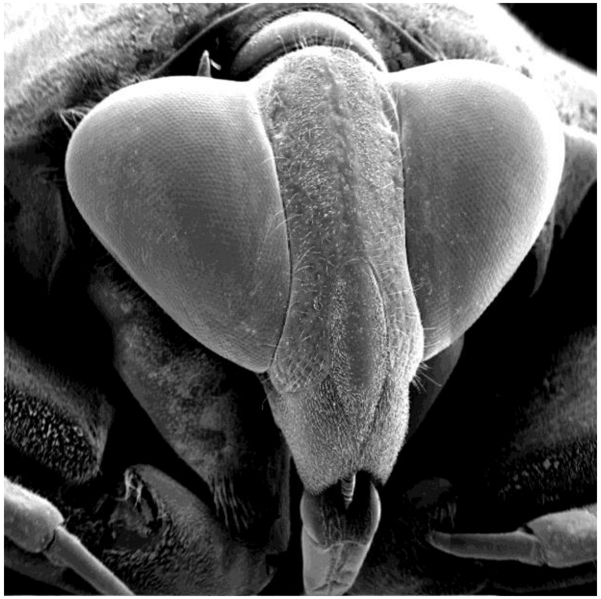
- February 96: DNS flaw in JDK 1.0.1
- March 96: Path name bug
- March 96: Princeton Class Loader bug
- May 96: type casting attack
- June 96: Array type implementation error
- July 96: More type casting problems
- August 96:Flaw in Microsoft's Java VM

All of these bugs have been fixed (but they're back)

- 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

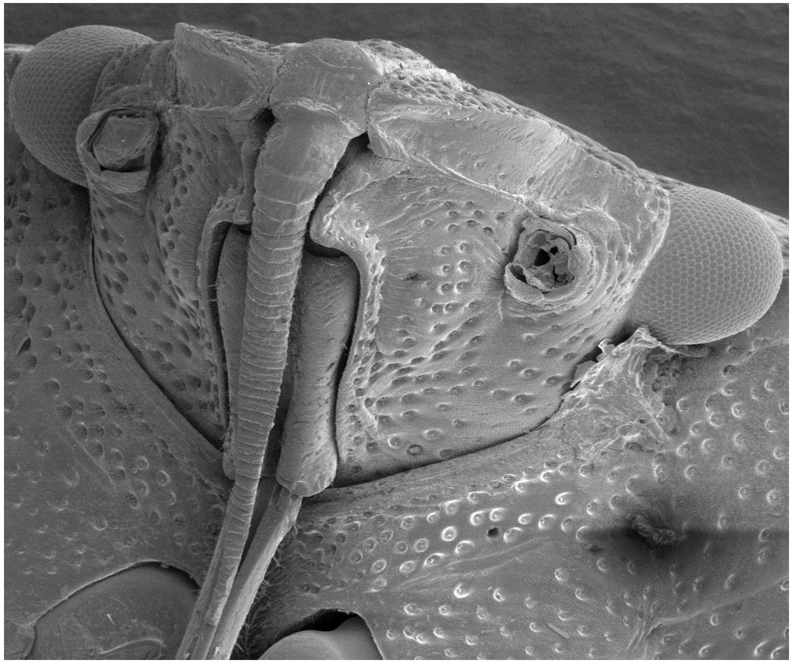
Bug: SQL injection

- 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’



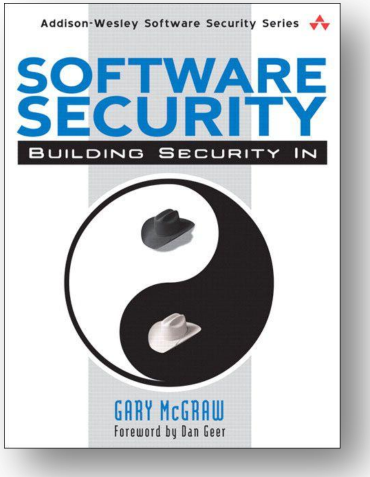
Bug: XSS

- 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



Addison-Wesley Software Security Series

SOFTWARE SECURITY

BUILDING SECURITY IN

GARY McGRAW

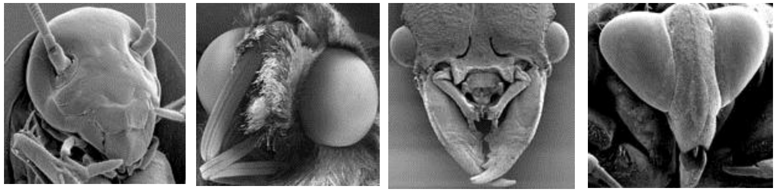
Foreword by Dan Geer

Bug parade FAIL

IMPLEMENTATION BUGS

- 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

50%



ARCHITECTURAL FLAWS

- 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

50%

SOFTWARE SECURITY ZOMBIES

#RSAC

cigital

RSACONFERENCE EUROPE 2013

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

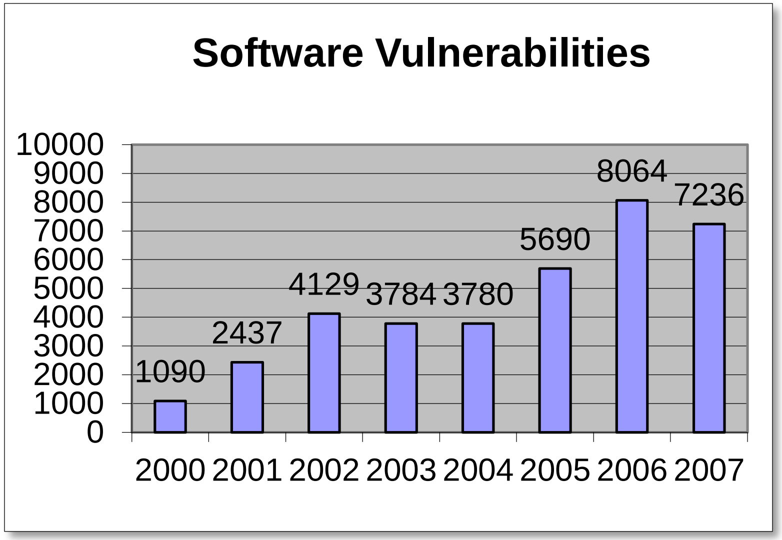
- Network security FAIL
- More code more bugs
- SDLC integration
- Bugs and flaws
- Badness-ometers

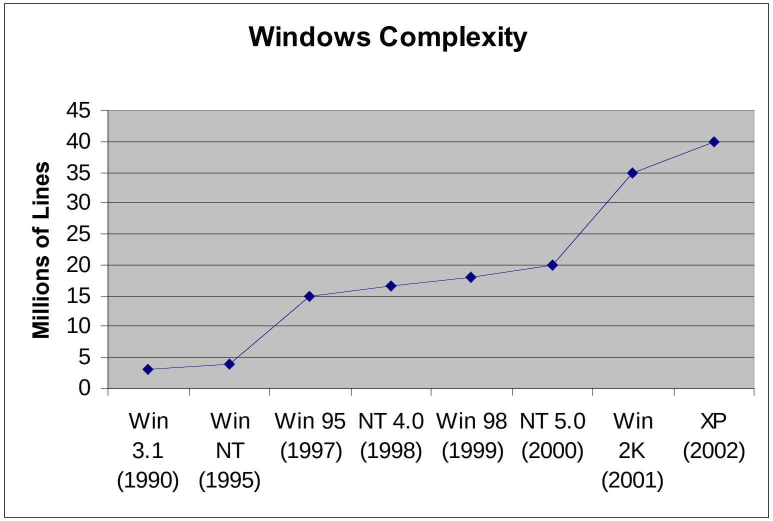
Zombie: old school security is reactive

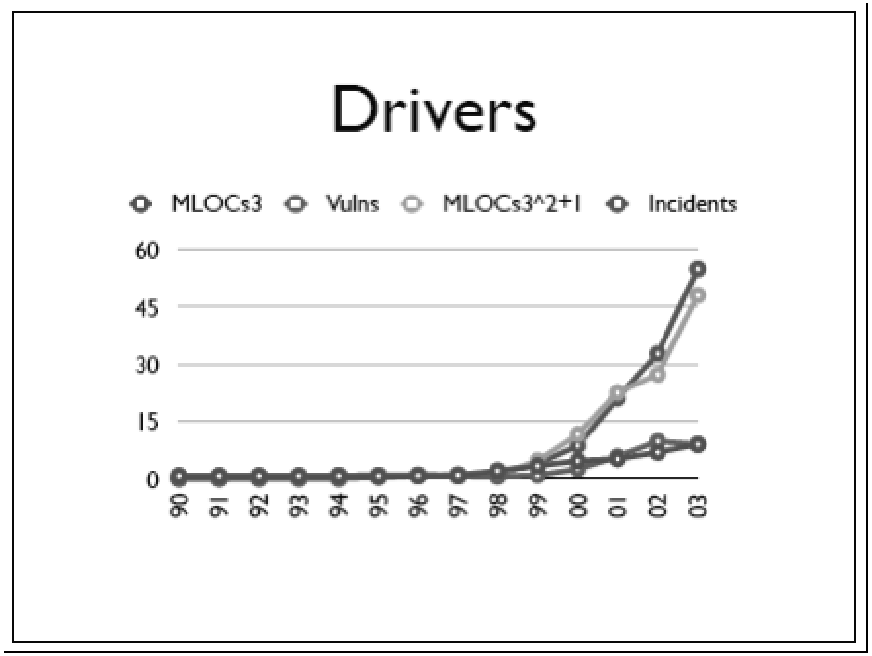
- Defend the "perimeter" with a firewall
  - To keep stuff out
- Promulgate "penetrate and patch"
- "Review" products when they're complete
  - Throw it over the wall testing
  - Too much weight on penetration testing
- Over-rely on security functions
  - "We use SSL"

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

Zombie: more code, more bugs

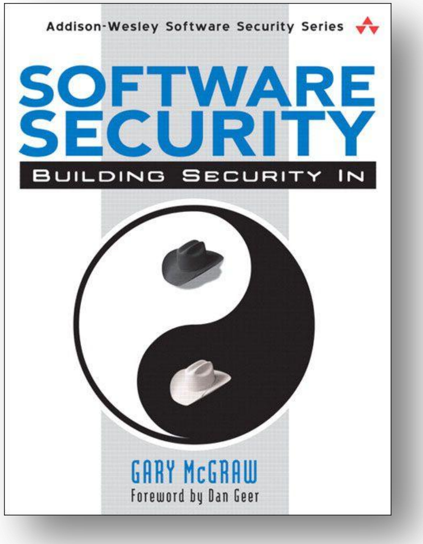

Software Vulnerabilities

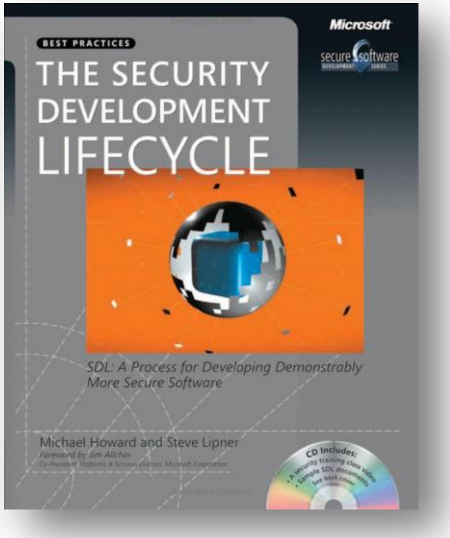

Windows Complexity


Drivers

Zombie: SDLC integration

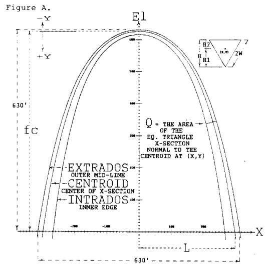
- Integrating best practices into large organizations
- Microsoft’s SDL
- Cigital’s touchpoints
- OWASP CLASP/SAMM



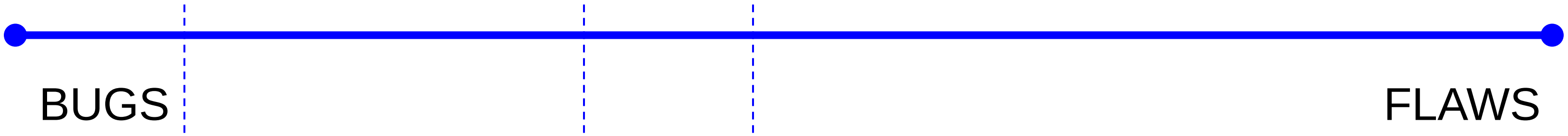

Microsoft
secure software
BEST PRACTICES
THE SECURITY DEVELOPMENT LIFECYCLE
SDL: A Process for Developing Demonstrably More Secure Software
Michael Howard and Steve Lipner
Foreword by Jim Allchin
CD Includes:

Zombie: bugs AND flaws

gets()



attacker in the middle



- Architectural risk analysis
- Customized static rules (Fidelity)
- Commercial SCA tools: Fortify, Ounce Labs, Coverity

Zombie: badness-ometer

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

Security in knowledge

#RSAC

cigital

RSA CONFERENCE EUROPE 2013

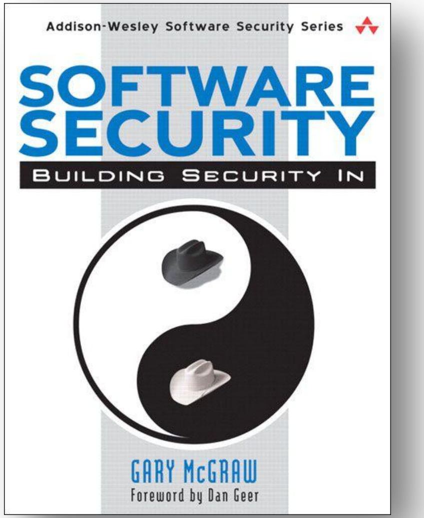
The rise of the software security group

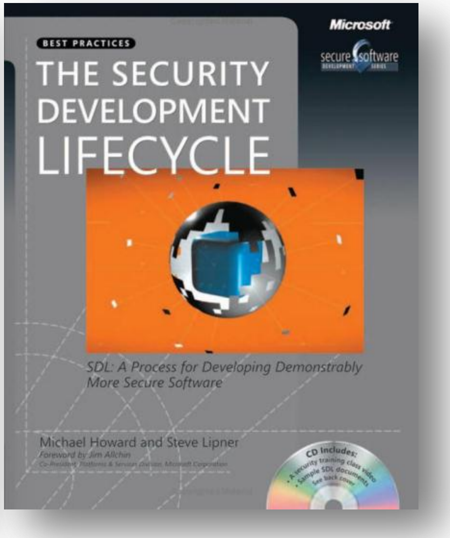
- Cigital SSG turned fifteen in 2012
- Microsoft adopts the Secure Development Lifecycle
- Most firms have a group devoted to software security

- 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

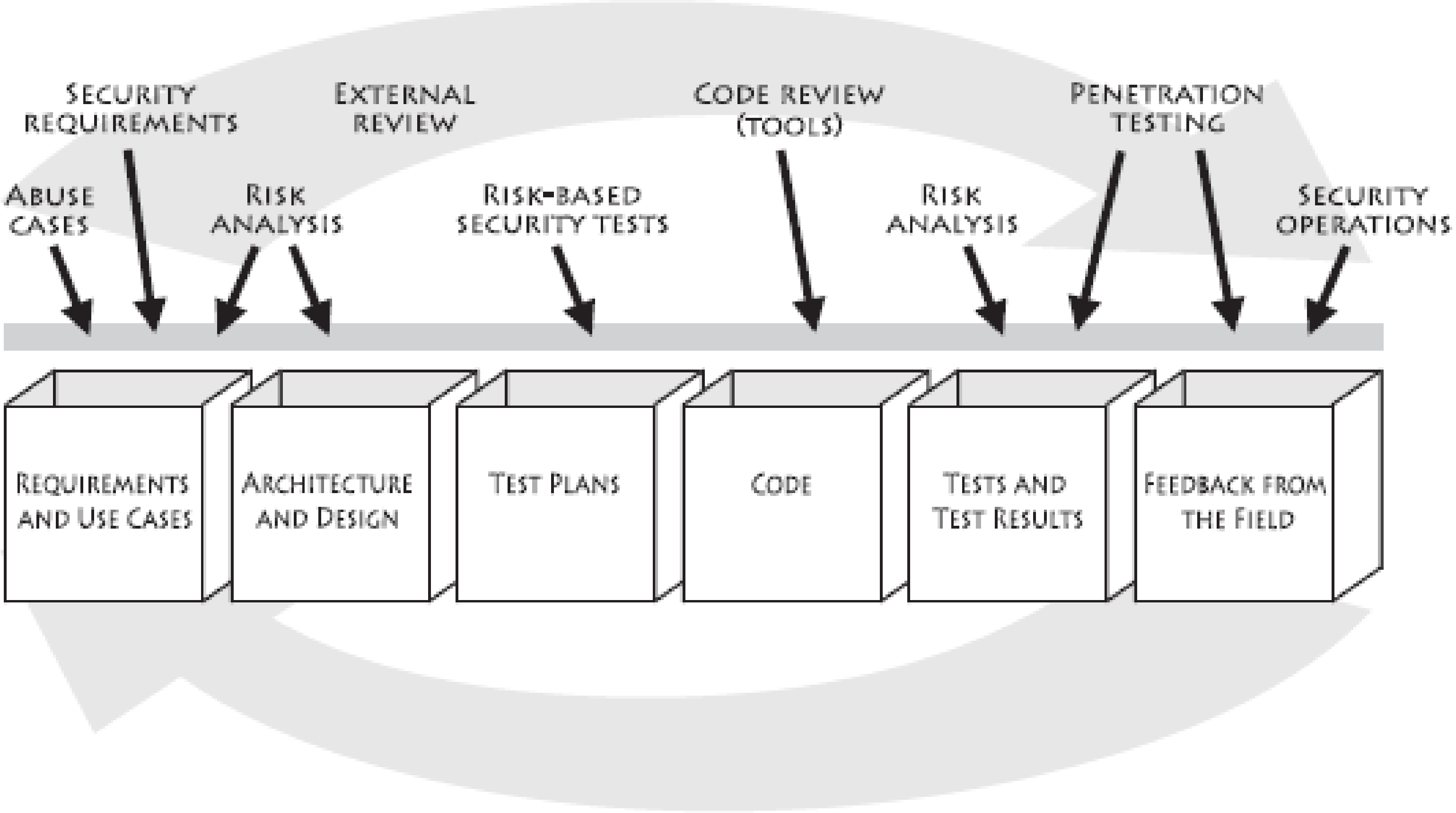
2006: shift from philosophy to HOW TO

- Integrating best practices into large organizations’ SDLC (that is, an SSDL)
  - Microsoft’s SDL
  - Cigital’s Touchpoints
  - OWASP CLASP




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
Co-President, Platforms & Services Division, Microsoft Corporation
CD Includes:
• A security training class video
• Sample SDL documents
See back cover

Software security touchpoints



BSIMM V

Building Security In Maturity Model

Security in knowledge

#RSAC

cigital

RSA CONFERENCE
EUROPE 2013

BSIMM: software security measurement

- □ Real data from (67) real initiatives
- □ 161 measurements
- □ 21 (4) over time
- □ McGraw, Migues, & West

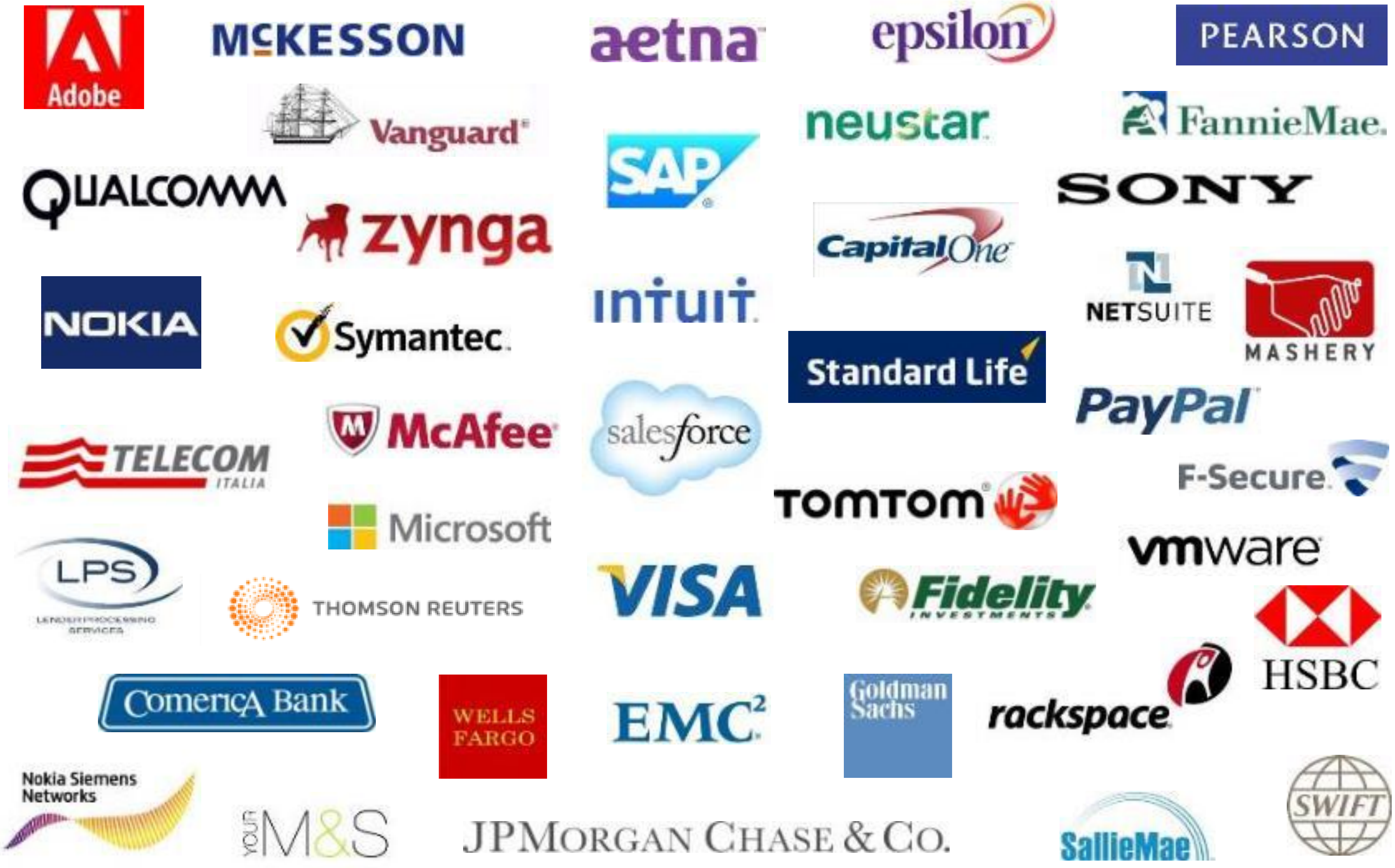
cigital

FORTIFY
An HP Company

PlexLogic

Minded
security

67 firms in the BSIMM community



Plus 22 firms that remain anonymous

BSIMM by the numbers

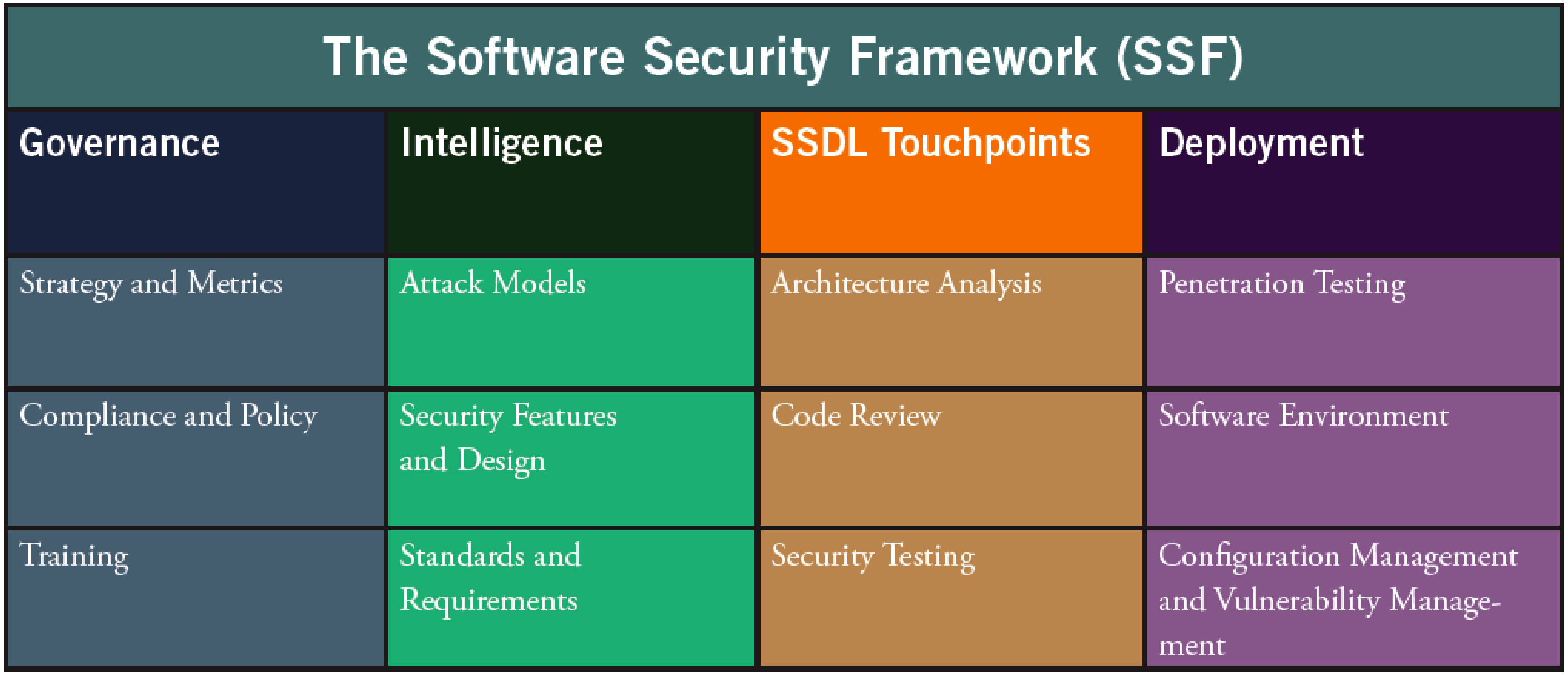
	BSIMM1	BSIMM2	BSIMM3	BSIMM4	BSIMM-V
Firms	9	30	42	51	67
Measurements	9	49	81	95	161
2nd Measurements	0	0	11	13	21
3rd Measurements	0	0	0	1	4
SSG Members	370	635	786	974	976
Satellite Members	710	1150	1750	2039	1954
Developers	67,950	141,175	185,316	218,286	272,358
Applications	3970	28,243	41,157	58,739	69,039
Avg SSG Age	5.32	4.49	4.32	4.13	4.28
SSG Avg of Avgs	1.13 / 100	1.02 / 100	1.99 / 100	1.95 / 100	1.4 / 100
Financials	4	12	17	19	26
ISVs	4	7	15	19	25
High Tech	2	7	10	13	14

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
- BSIMM describes and measures multiple prescriptive approaches

A software security framework

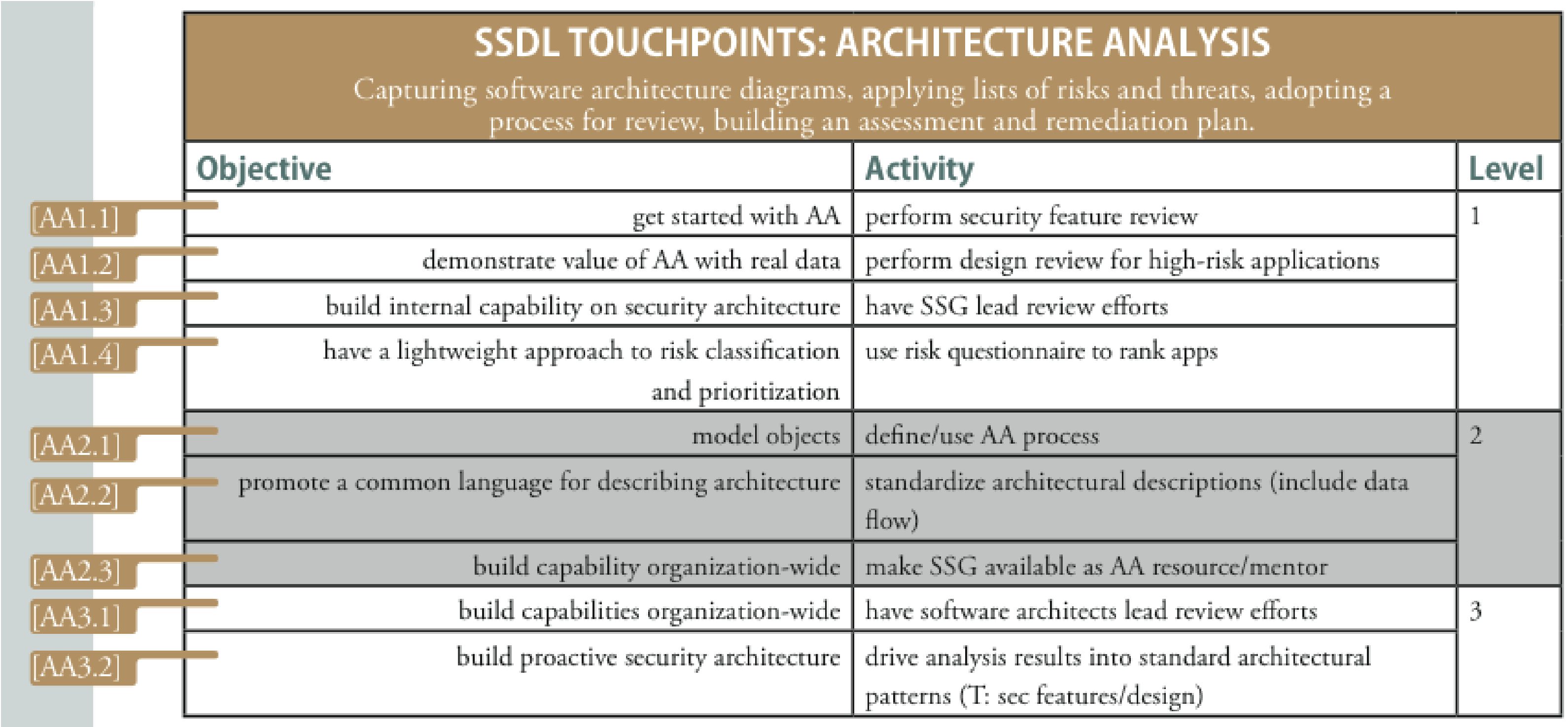
- Four domains
- Twelve practices



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

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 software security group (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.

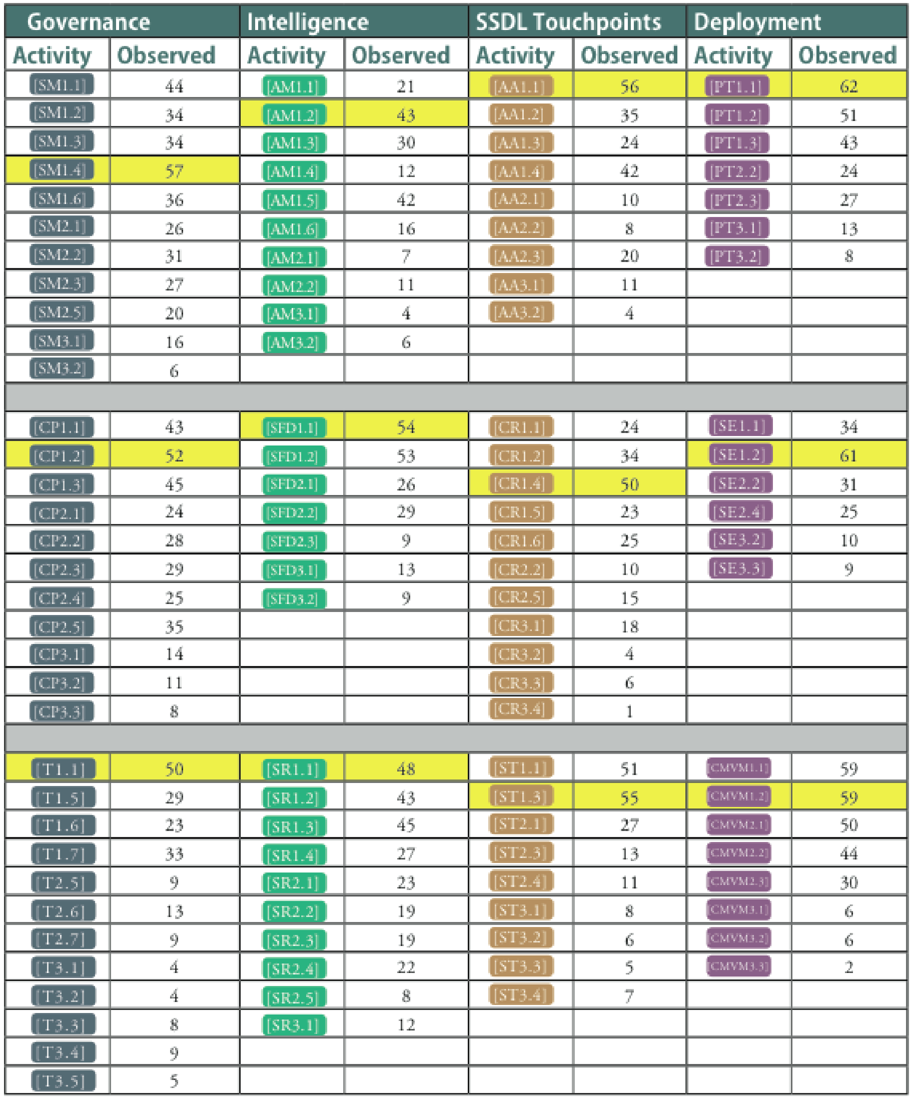
Real-world data (67 firms)

- Initiative age
  - Average: 6 years
  - Newest: 0.4
  - Oldest: 18.1
  - Median: 5.3
- SSG size
  - Average: 14.78
  - Smallest: 1
  - Largest: 100
  - Median: 7

- Satellite size
  - Average: 29.6
  - Smallest: 0
  - Largest: 400
  - Median: 4
- Dev size
  - Average: 4190
  - Smallest: 11
  - Largest: 30,000
  - Median: 1600

Average SSG size: 1.4% of dev group size

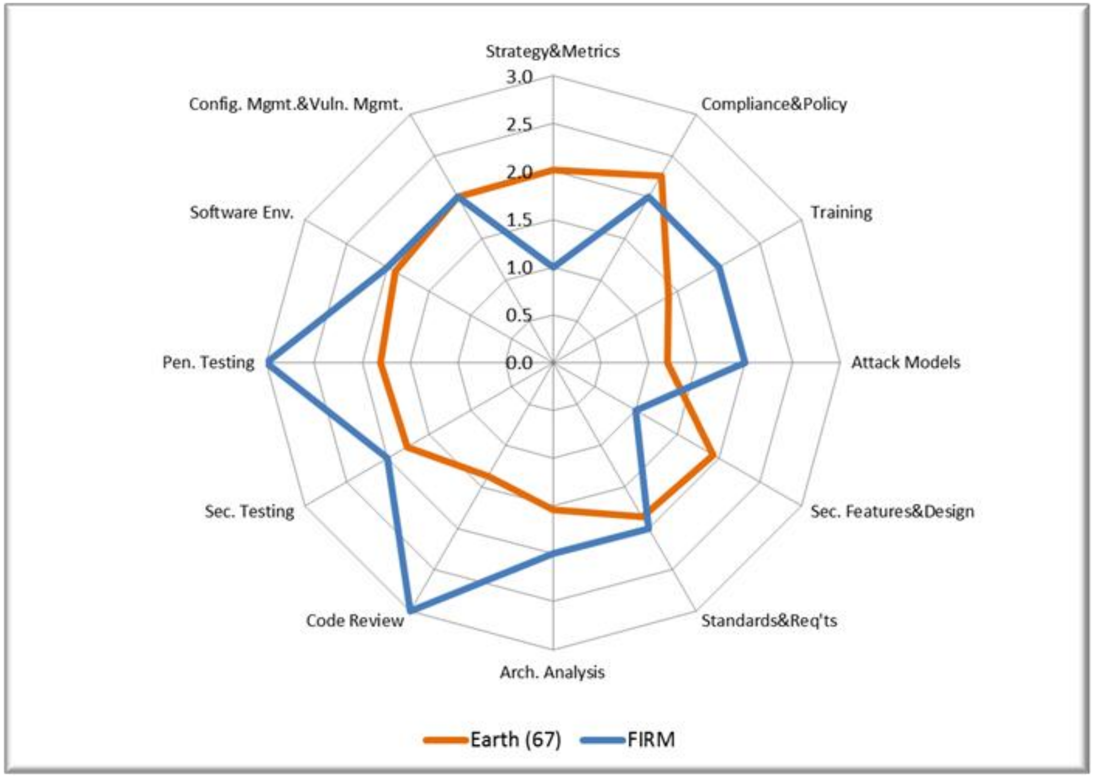
BSIMM-V scorecard



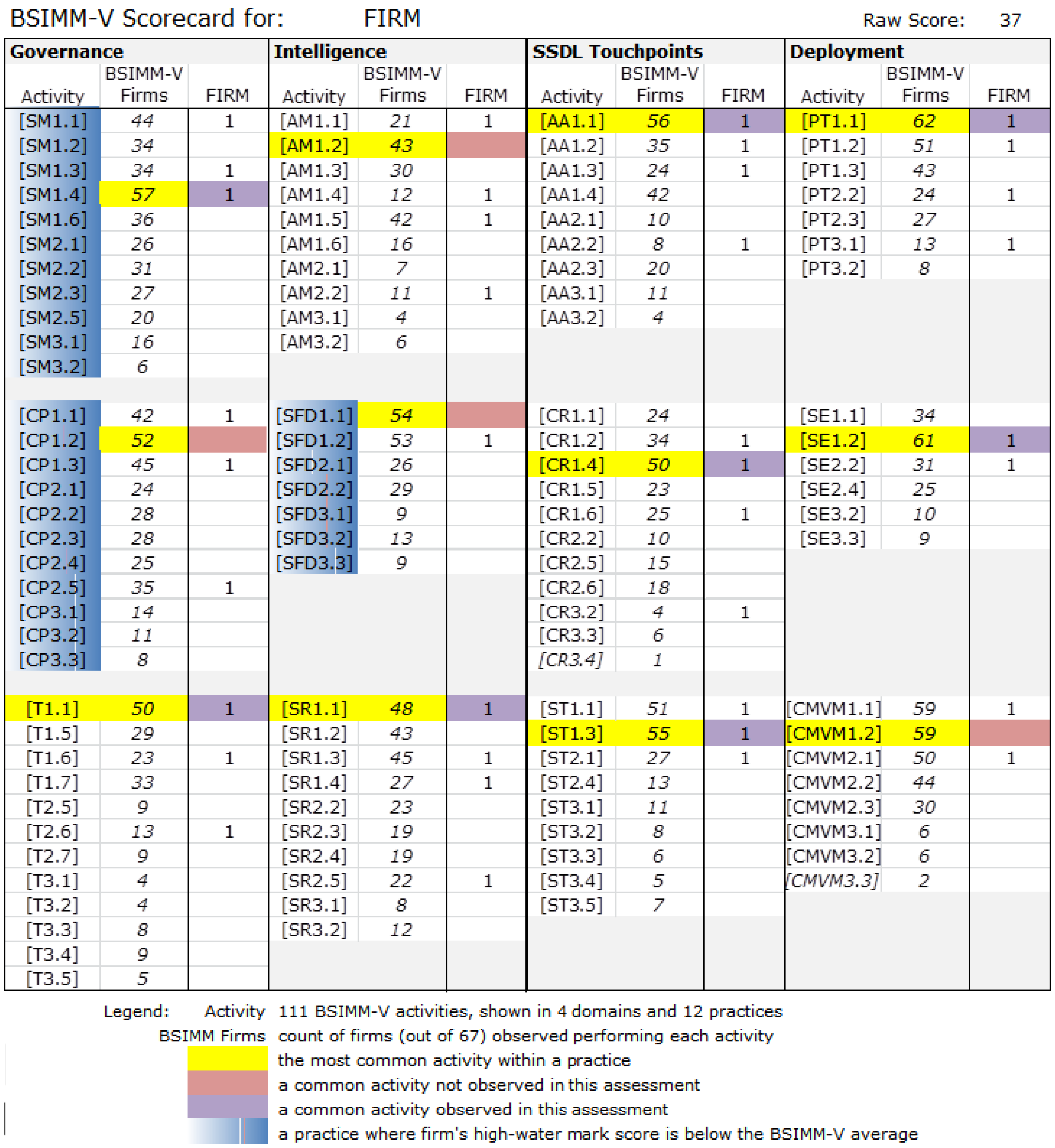
Governance		Intelligence		SSDL Touchpoints		Deployment	
Activity	Observed	Activity	Observed	Activity	Observed	Activity	Observed
[SM1.1]	44	[AM1.1]	21	[AA1.1]	56	[PT1.1]	62
[SM1.2]	34	[AM1.2]	43	[AA1.2]	35	[PT1.2]	51
[SM1.3]	34	[AM1.3]	30	[AA1.3]	24	[PT1.3]	43
[SM1.4]	57	[AM1.4]	12	[AA1.4]	42	[PT2.2]	24
[SM1.6]	36	[AM1.5]	42	[AA2.1]	10	[PT2.3]	27
[SM2.1]	26	[AM1.6]	16	[AA2.2]	8	[PT3.1]	13
[SM2.2]	31	[AM2.1]	7	[AA2.3]	20	[PT3.2]	8
[SM2.3]	27	[AM2.2]	11	[AA3.1]	11		
[SM2.5]	20	[AM3.1]	4	[AA3.2]	4		
[SM3.1]	16	[AM3.2]	6				
[SM3.2]	6						
[CP1.1]	43	[SFD1.1]	54	[CR1.1]	24	[SE1.1]	34
[CP1.2]	52	[SFD1.2]	53	[CR1.2]	34	[SE1.2]	61
[CP1.3]	45	[SFD2.1]	26	[CR1.4]	50	[SE2.2]	31
[CP2.1]	24	[SFD2.2]	29	[CR1.5]	23	[SE2.4]	25
[CP2.2]	28	[SFD2.3]	9	[CR1.6]	25	[SE3.2]	10
[CP2.3]	29	[SFD3.1]	13	[CR2.2]	10	[SE3.3]	9
[CP2.4]	25	[SFD3.2]	9	[CR2.5]	15		
[CP2.5]	35			[CR3.1]	18		
[CP3.1]	14			[CR3.2]	4		
[CP3.2]	11			[CR3.3]	6		
[CP3.3]	8			[CR3.4]	1		
[T1.1]	50	[SR1.1]	48	[ST1.1]	51	[CMVM1.1]	59
[T1.5]	29	[SR1.2]	43	[ST1.3]	55	[CMVM1.2]	59
[T1.6]	23	[SR1.3]	45	[ST2.1]	27	[CMVM2.1]	50
[T1.7]	33	[SR1.4]	27	[ST2.3]	13	[CMVM2.2]	44
[T2.5]	9	[SR2.1]	23	[ST2.4]	11	[CMVM2.3]	30
[T2.6]	13	[SR2.2]	19	[ST3.1]	8	[CMVM3.1]	6
[T2.7]	9	[SR2.3]	19	[ST3.2]	6	[CMVM3.2]	6
[T3.1]	4	[SR2.4]	22	[ST3.3]	5	[CMVM3.3]	2
[T3.2]	4	[SR2.5]	8	[ST3.4]	7		
[T3.3]	8	[SR3.1]	12				
[T3.4]	9						
[T3.5]	5						

BSIMM-V as a measuring stick

- □ Compare a firm with peers using the high water mark view
- □ Compare business units
- □ Chart an SSI over time



BSIMM-V scorecard with FAKE firm data



BSIMM-V Scorecard for: FIRM

Raw Score: 37

Governance			Intelligence			SSDL Touchpoints			Deployment		
Activity	BSIMM-V Firms	FIRM	Activity	BSIMM-V Firms	FIRM	Activity	BSIMM-V Firms	FIRM	Activity	BSIMM-V Firms	FIRM
[SM1.1]	44	1	[AM1.1]	21	1	[AA1.1]	56	1	[PT1.1]	62	1
[SM1.2]	34		[AM1.2]	43	🟥	[AA1.2]	35	1	[PT1.2]	51	1
[SM1.3]	34	1	[AM1.3]	30		[AA1.3]	24	1	[PT1.3]	43	
[SM1.4]	57	1	[AM1.4]	12	1	[AA1.4]	42		[PT2.2]	24	1
[SM1.6]	36		[AM1.5]	42	1	[AA2.1]	10		[PT2.3]	27	
[SM2.1]	26		[AM1.6]	16		[AA2.2]	8	1	[PT3.1]	13	1
[SM2.2]	31		[AM2.1]	7		[AA2.3]	20		[PT3.2]	8	
[SM2.3]	27		[AM2.2]	11	1	[AA3.1]	11				
[SM2.5]	20		[AM3.1]	4		[AA3.2]	4				
[SM3.1]	16		[AM3.2]	6							
[SM3.2]	6										
[CP1.1]	42	1	[SFD1.1]	54	🟥	[CR1.1]	24		[SE1.1]	34	
[CP1.2]	52	🟥	[SFD1.2]	53	1	[CR1.2]	34	1	[SE1.2]	61	1
[CP1.3]	45	1	[SFD2.1]	26		[CR1.4]	50	1	[SE2.2]	31	1
[CP2.1]	24		[SFD2.2]	29		[CR1.5]	23		[SE2.4]	25	
[CP2.2]	28		[SFD3.1]	9		[CR1.6]	25	1	[SE3.2]	10	
[CP2.3]	28		[SFD3.2]	13		[CR2.2]	10		[SE3.3]	9	
[CP2.4]	25		[SFD3.3]	9		[CR2.5]	15				
[CP2.5]	35	1				[CR2.6]	18				
[CP3.1]	14					[CR3.2]	4	1			
[CP3.2]	11					[CR3.3]	6				
[CP3.3]	8					*[CR3.4]*	1				
[T1.1]	50	1	[SR1.1]	48	1	[ST1.1]	51	1	[CMVM1.1]	59	1
[T1.5]	29		[SR1.2]	43		[ST1.3]	55	1	[CMVM1.2]	59	🟥
[T1.6]	23	1	[SR1.3]	45	1	[ST2.1]	27	1	[CMVM2.1]	50	1
[T1.7]	33		[SR1.4]	27	1	[ST2.4]	13		[CMVM2.2]	44	
[T2.5]	9		[SR2.2]	23		[ST3.1]	11		[CMVM2.3]	30	
[T2.6]	13	1	[SR2.3]	19		[ST3.2]	8		[CMVM3.1]	6	
[T2.7]	9		[SR2.4]	19		[ST3.3]	6		[CMVM3.2]	6	
[T3.1]	4		[SR2.5]	22	1	[ST3.4]	5		*[CMVM3.3]*	2	
[T3.2]	4		[SR3.1]	8		[ST3.5]	7				
[T3.3]	8		[SR3.2]	12							
[T3.4]	9										
[T3.5]	5										

Legend:		
	Activity	111 BSIMM-V activities, shown in 4 domains and 12 practices
	BSIMM Firms	count of firms (out of 67) observed performing each activity
	🟨	the most common activity within a practice
	🟥	a common activity not observed in this assessment
	🟪	a common activity observed in this assessment
	🟦	a practice where firm's high-water mark score is below the BSIMM-V average

- ❑ Top 12 activities
  - ❑ purple = good?
  - ❑ red = bad?
- ❑ "Blue shift" practices to emphasize

cigital

BSIMM-V to BSIMM6

- ❑ BSIMM-V released October 2013 under creative commons
  - ❑ http://bsimm.com
  - ❑ Italian, German, and Spanish translations available
- ❑ BSIMM is a yardstick
  - ❑ Use it to see where you stand
  - ❑ Use it to figure out what your peers do
- ❑ BSIMM-V→BSIMM6
  - ❑ BSIMM is growing

WHERE TO LEARN MORE

SearchSecurity + Silver Bullet

TechTarget

SearchSecurity

www.searchsecurity.com

No-nonsense monthly security column by Gary McGraw

www.cigital.com/~gem/writing

www.cigital.com/justiceleague

In-depth thought leadership blog from the Cigital Principals

- Scott Matsumoto
- Gary McGraw
- Sammy Migues
- John Steven
- Paco Hope

www.cigital.com/silverbullet

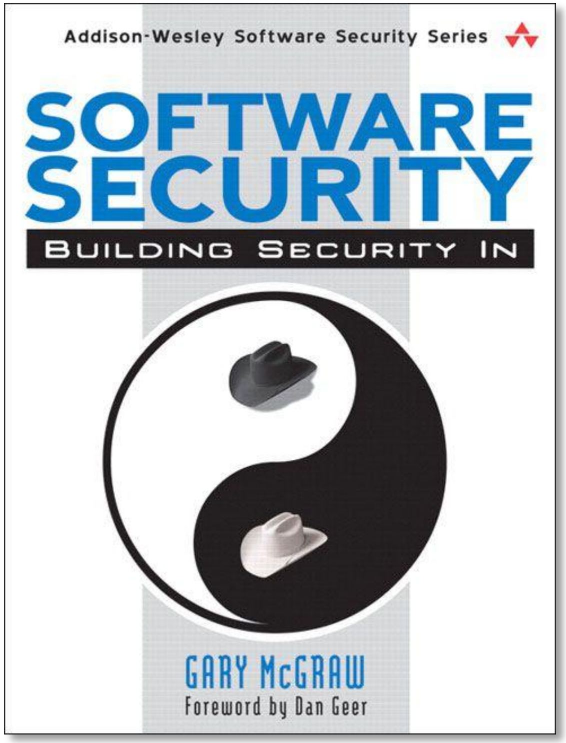
Build security in

http://bsimm.com

THANK YOU

Read 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

Addison Wesley

Thank you!

Dr. Gary McGraw

CTO, Cigital

@cigitalgem

gem@cigital.com

http://www.cigital.com/~gem