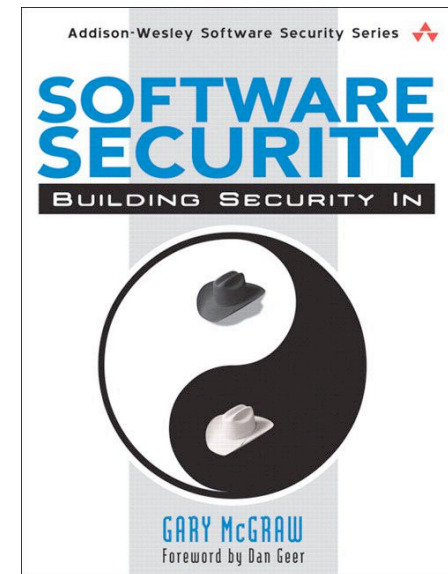




Software Security Touchpoint: Architectural Risk Analysis

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

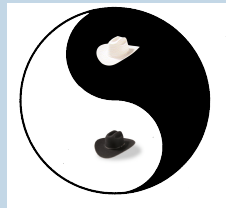




Cigital

- 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

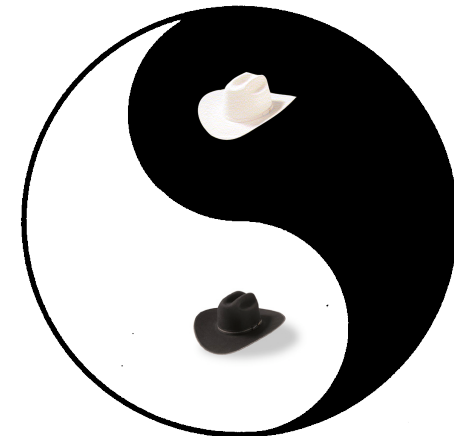
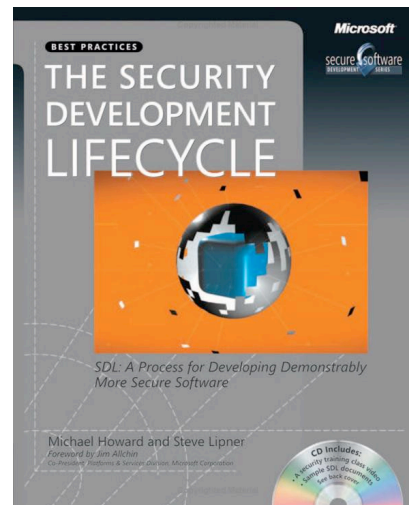




ARA in Context: State of the Practice

A shift from philosophy to HOW TO

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



What works: BSIMM

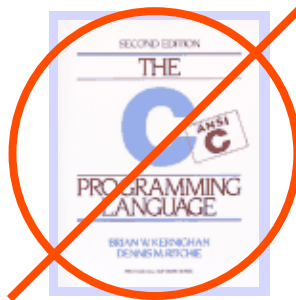


- Building Security In Maturity Model
- Real data from real initiatives

Two kinds of security defects

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
- Cross-site scripting
- SQL injection



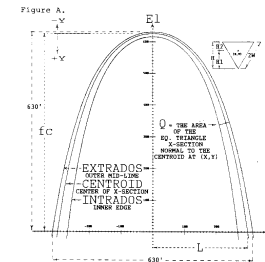
ARCHITECTURAL FLAWS

- Misuse of cryptography
- Compartmentalization problems in design
- Privileged block protection on 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

The bugs/flaws continuum



gets ()



attacker in the middle

BUGS

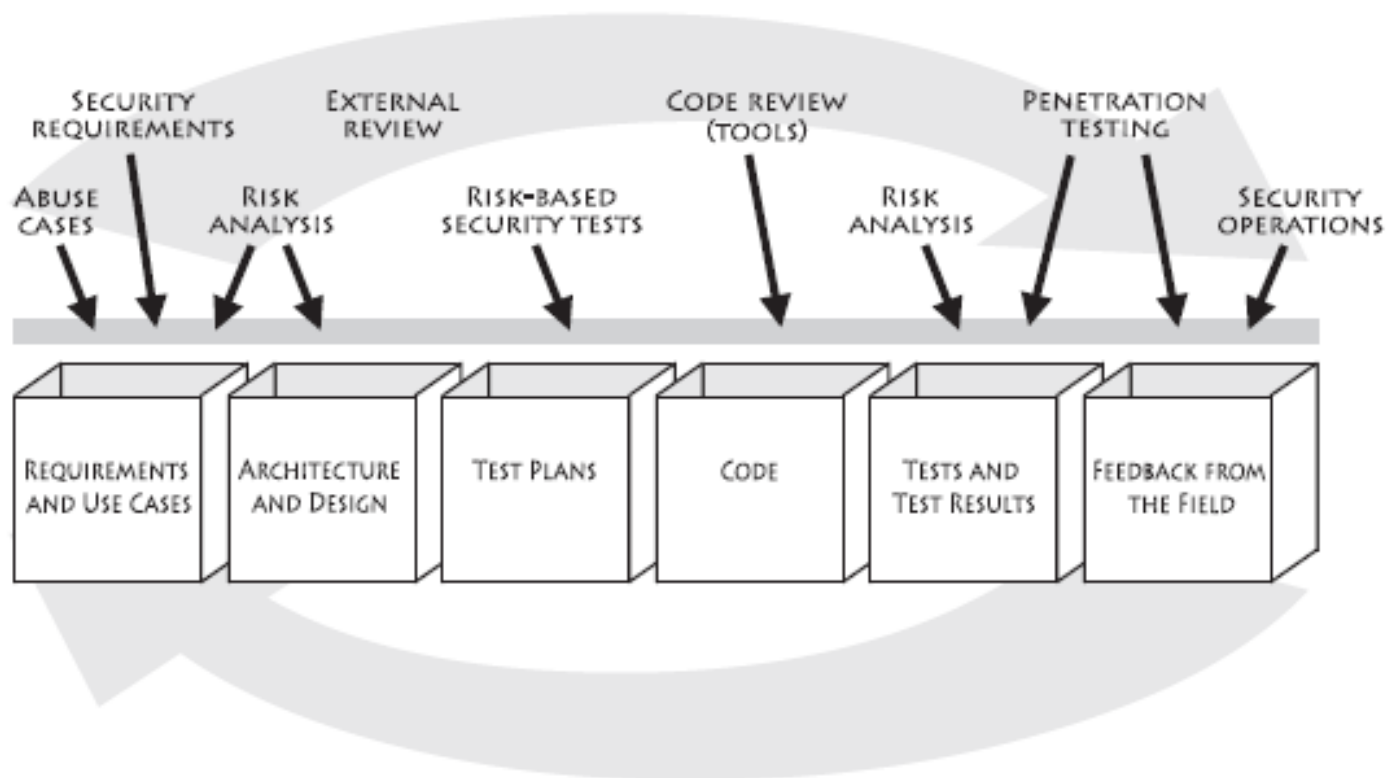
FLAWS

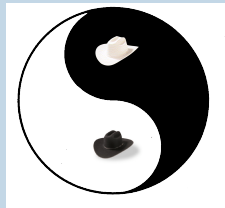
- Open source tools: ITS4, RATS, grep()
- Commercial SCA tools: Fortify, Ounce Labs, Coverity
- Customized static rules (Fidelity)
- Architectural risk analysis



cigital

Software security touchpoints





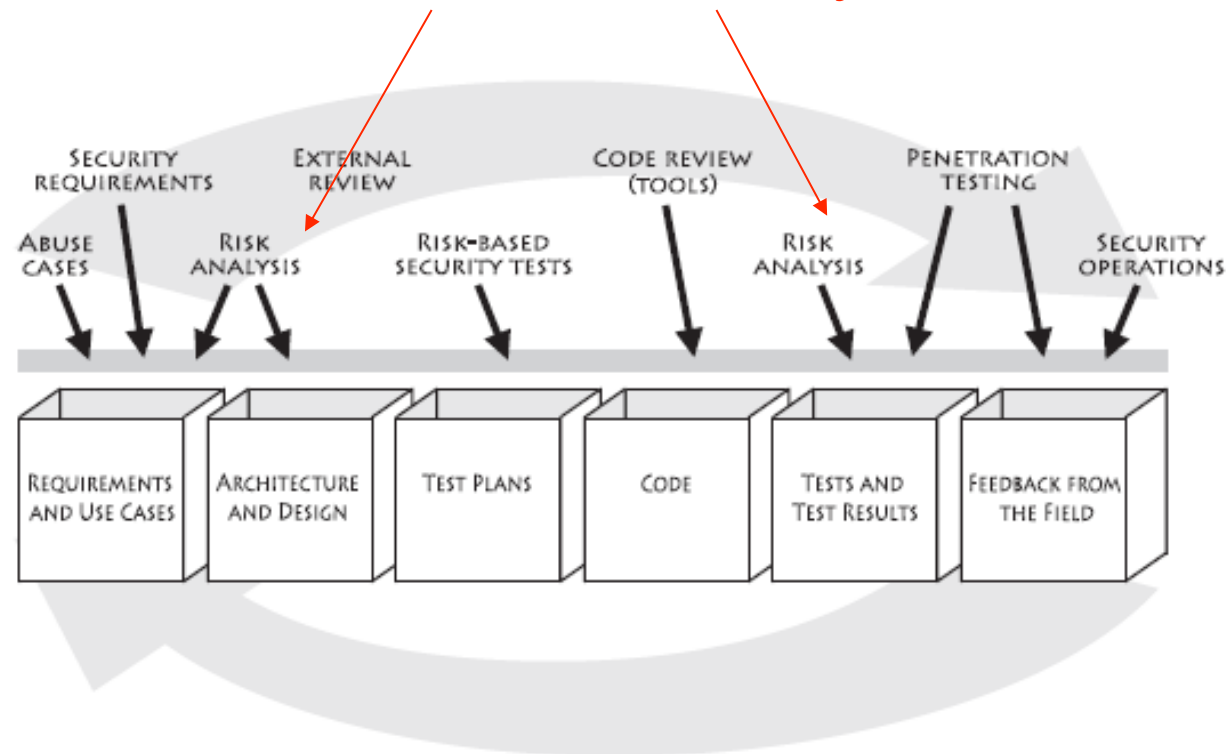
Architectural Risk Analysis

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

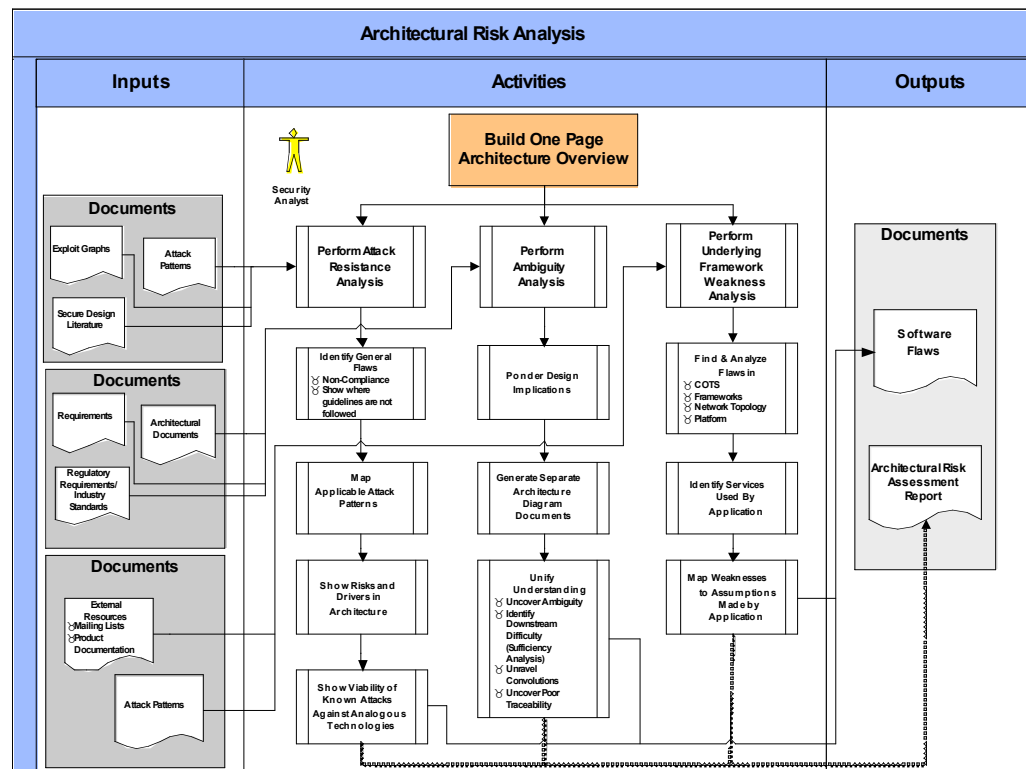
- <http://www.informit.com/articles/article.aspx?p=1315431>

Architectural Risk Analysis

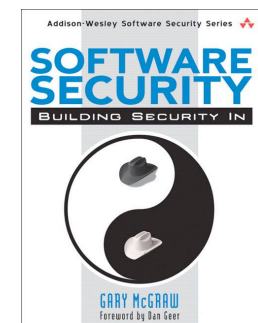


For more information, see
<http://www.cigital.com/services/security/>

Touchpoint: Architectural risk analysis

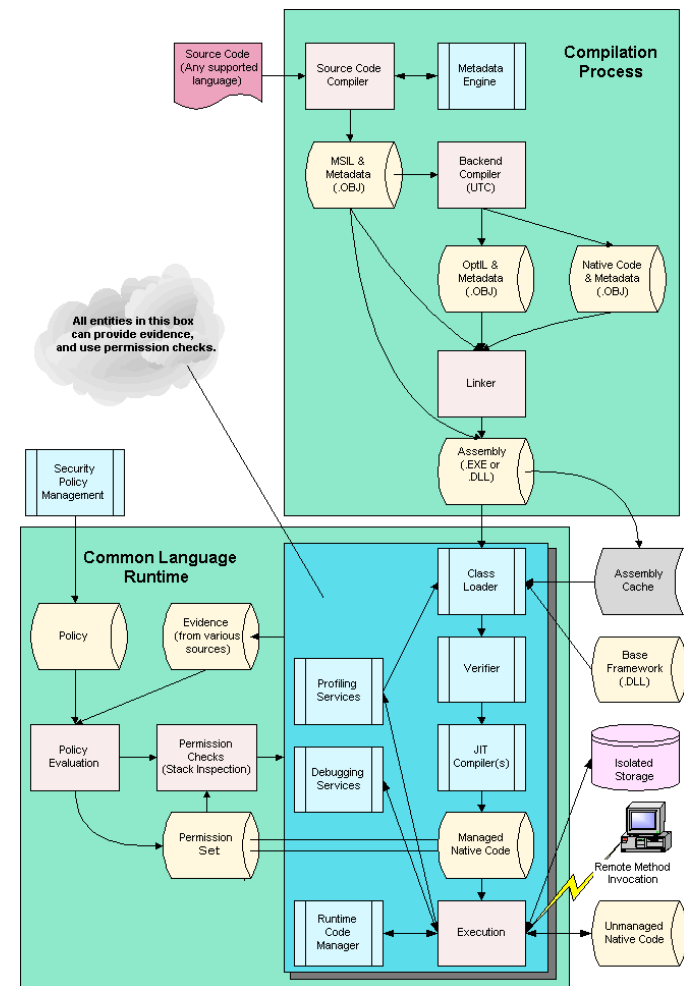


- Start by building a one-page overview of your system
- Then apply the three-step process
 - Attack resistance
 - Ambiguity analysis
 - Weakness analysis

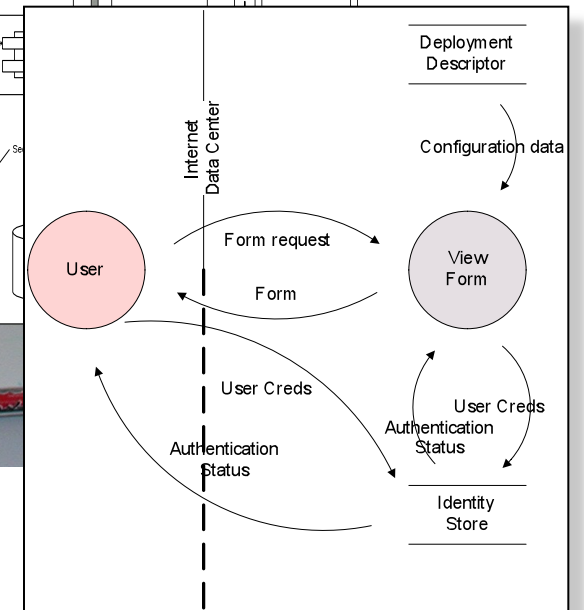
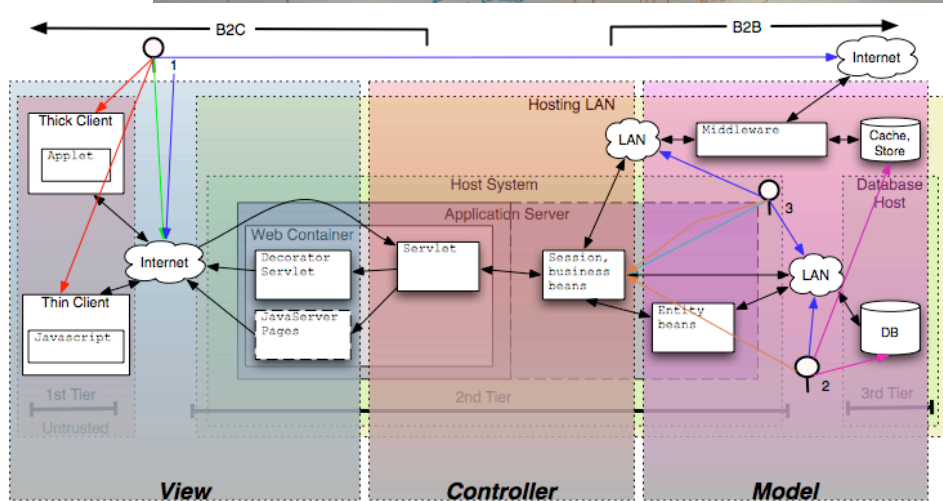
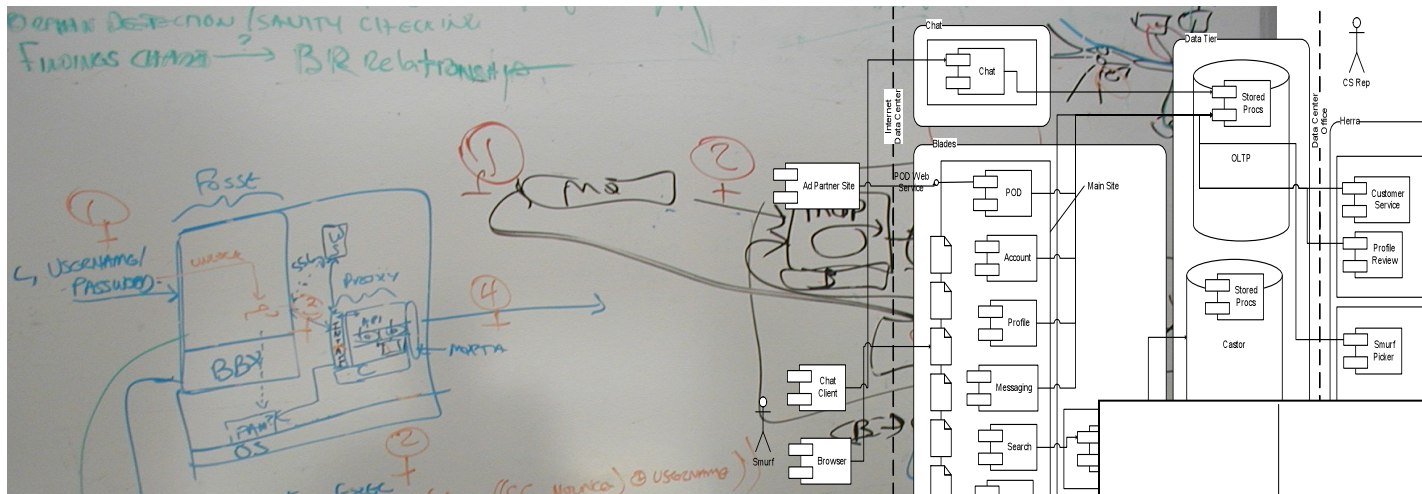


Touchpoint: Architectural risk analysis

- Step one: get an architecture
- Forrest level view
 - Up out of the code
- Widespread use of common components helps (but also has security impact!)
 - Spring
 - Hibernate
 - Log4J
 - OpenSSL
- Design patterns also help

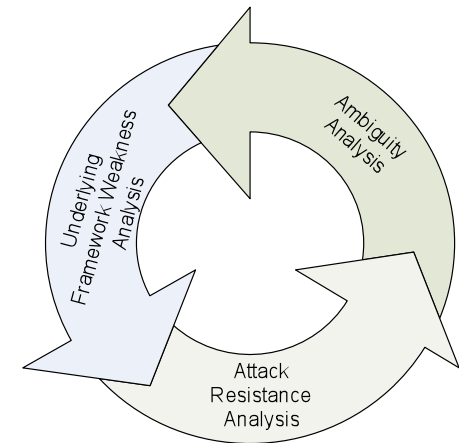


Design diagrams need security too



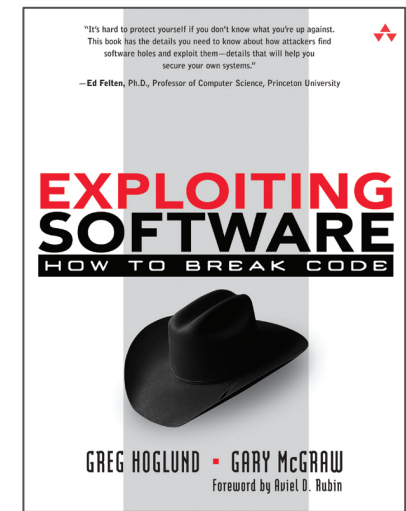
Three steps to ARA

- Attack Resistance (use a CHECKLIST)
 - Apply a list of known attacks (like STRIDE)
 - Calculate risk-based impact
- Ambiguity Analysis (multiple PERSPECTIVES)
 - Find attacks based on how the system works
 - Expose invalid assumptions
- Weakness Analysis (DEPENDENCIES)
 - Think through dependencies: toolkits and frameworks
 - In, Over, Under, Outside



Attack resistance: build an attack checklist

- Understand known attacks
 - Designers – what controls are needed to prevent common attacks?
 - Attackers – what to try again
- Example: Microsoft SDL's STRIDE model
 - Spoofing, tampering, repudiation, info disclosure, denial of service, elevation of privilege
- Start with common taxonomies
 - 7 Pernicious Kingdoms; McGraw
 - 19 Deadly Sins; Howard, LeBlanc, Viega
 - 48 Attack Patterns; McGraw/Hoglund
 - Common Weakness Enumeration
 - <http://cve.mitre.org/cwe>



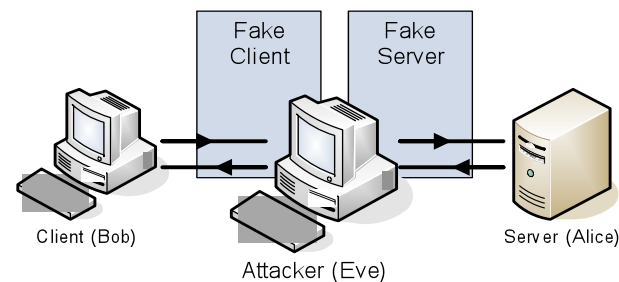


Attack resistance: common design elements

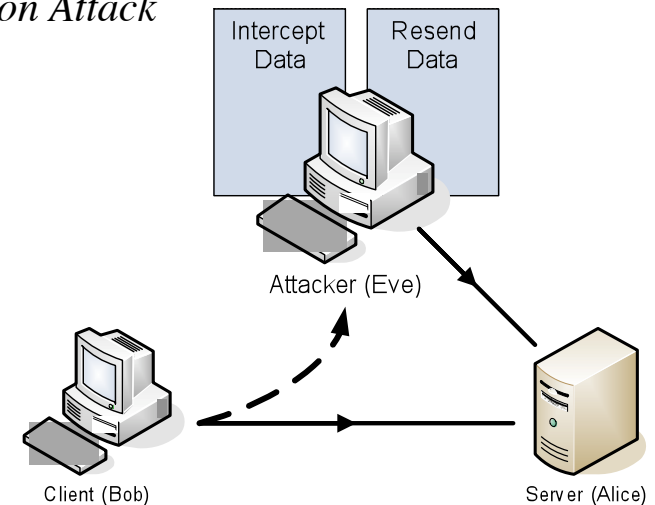
- Flag design elements that are historically vulnerable to attack
- Enterprise applications share many of the same design elements
 - Distributed architecture
 - Dynamic code generation and interpretation
 - APIs across stateless protocols
 - Rich Internet Applications
 - Service-oriented Architecture

Example: distributed architecture risks

- Distributed systems are susceptible to network-based attacks
 - Eavesdropping
 - Tampering
 - Spoofing
 - Hijacking
 - Observing
- Relevant Attack Patterns
 - Interposition attacks
 - Network sniffing
 - Replay attacks



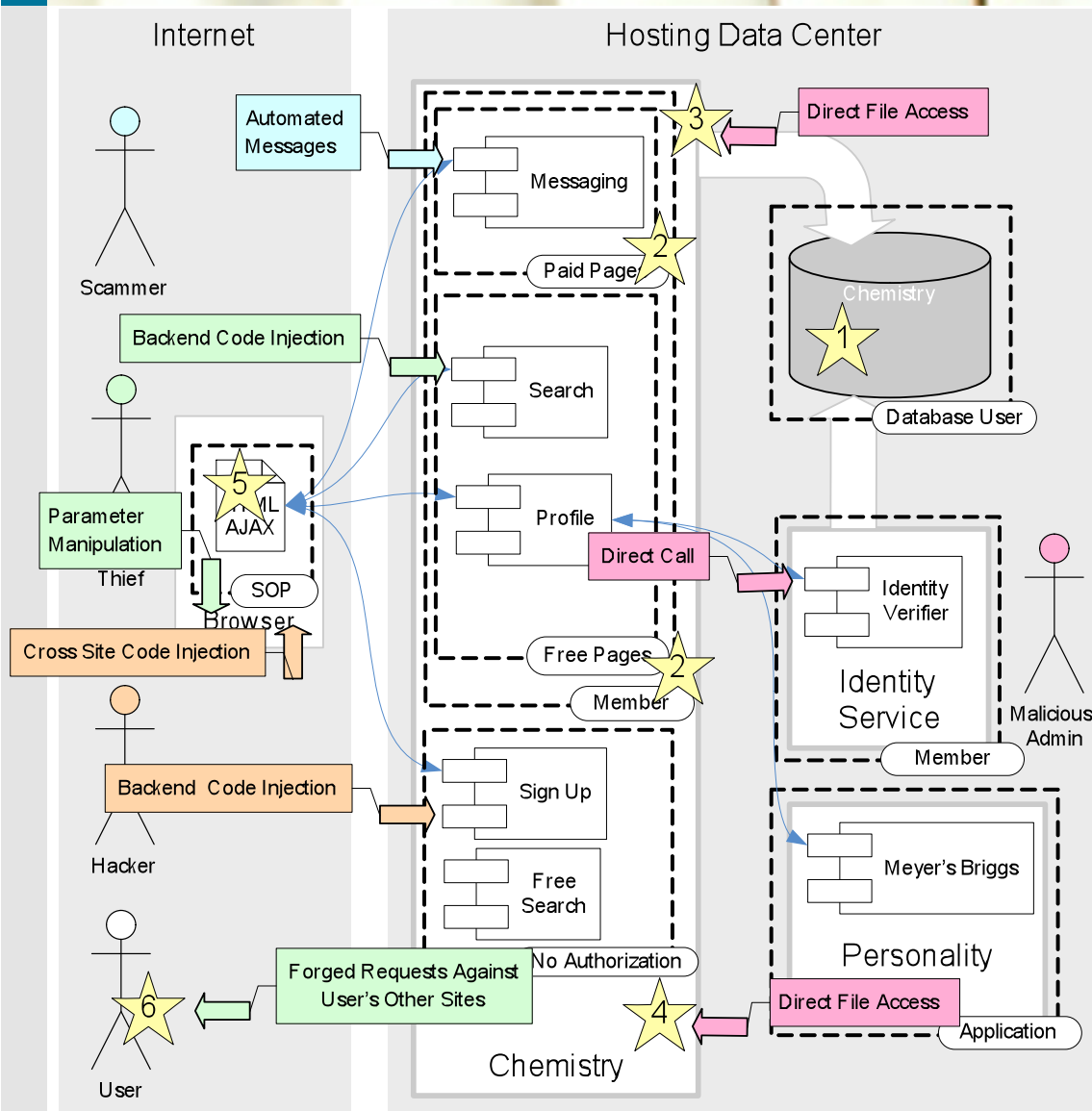
Interposition Attack



Replay Attack

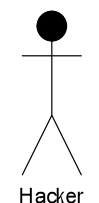
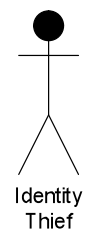
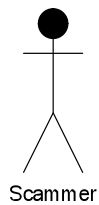
Ambiguity analysis: model your stuff

- Modeling techniques help expose an application's area of potential vulnerability
- Multiple points of view (and sets of experience) help
- **Trust Modeling** identifies the boundaries for security policy for function and data
- **Data Sensitivity Modeling** identifies privacy and trust issues for application data
- **Threat Modeling** identifies the attacker's perspective and areas of weakness



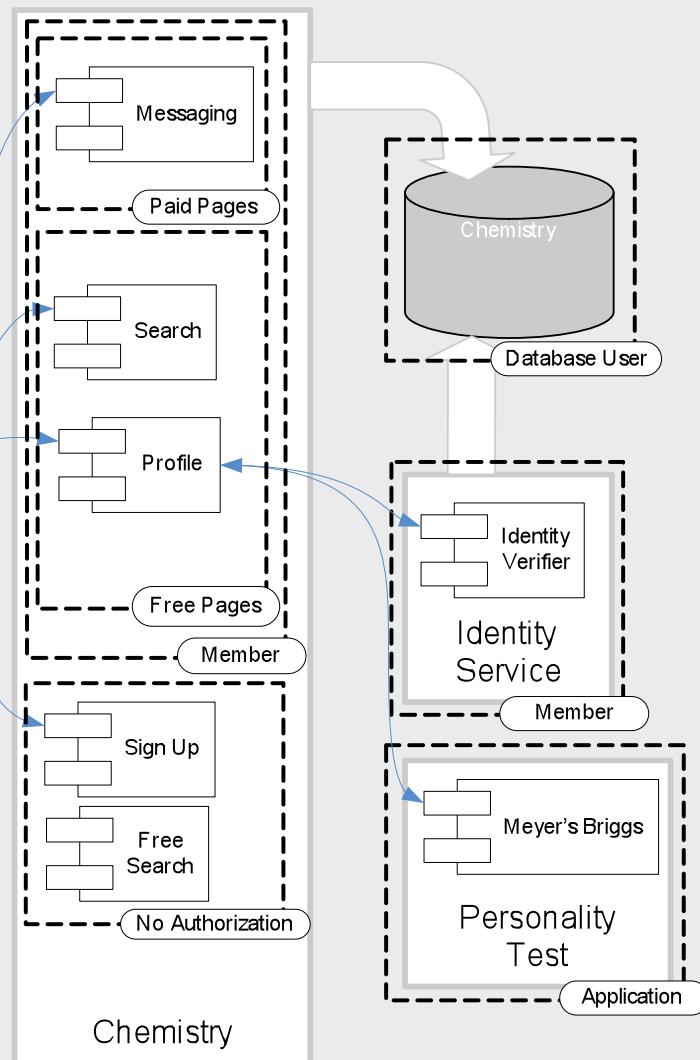
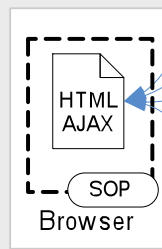
Ex: Threat modeling

- Threat: agents of malicious intent
- Asset: function and data the threat desires
- Point of Attack: Design element requiring hardening and/or the method of attack



Internet

Hosting Data Center



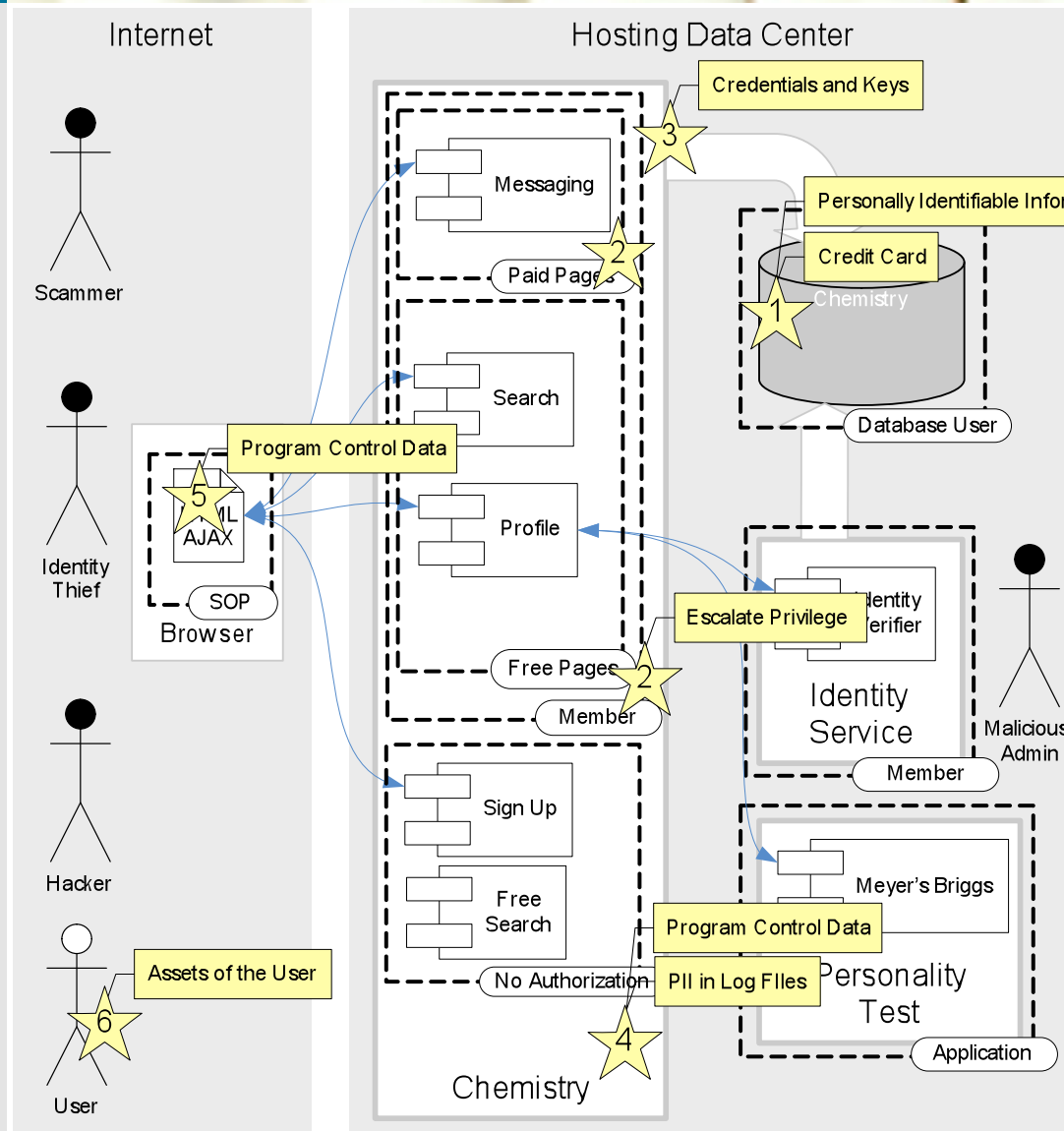
Ex: modeling users

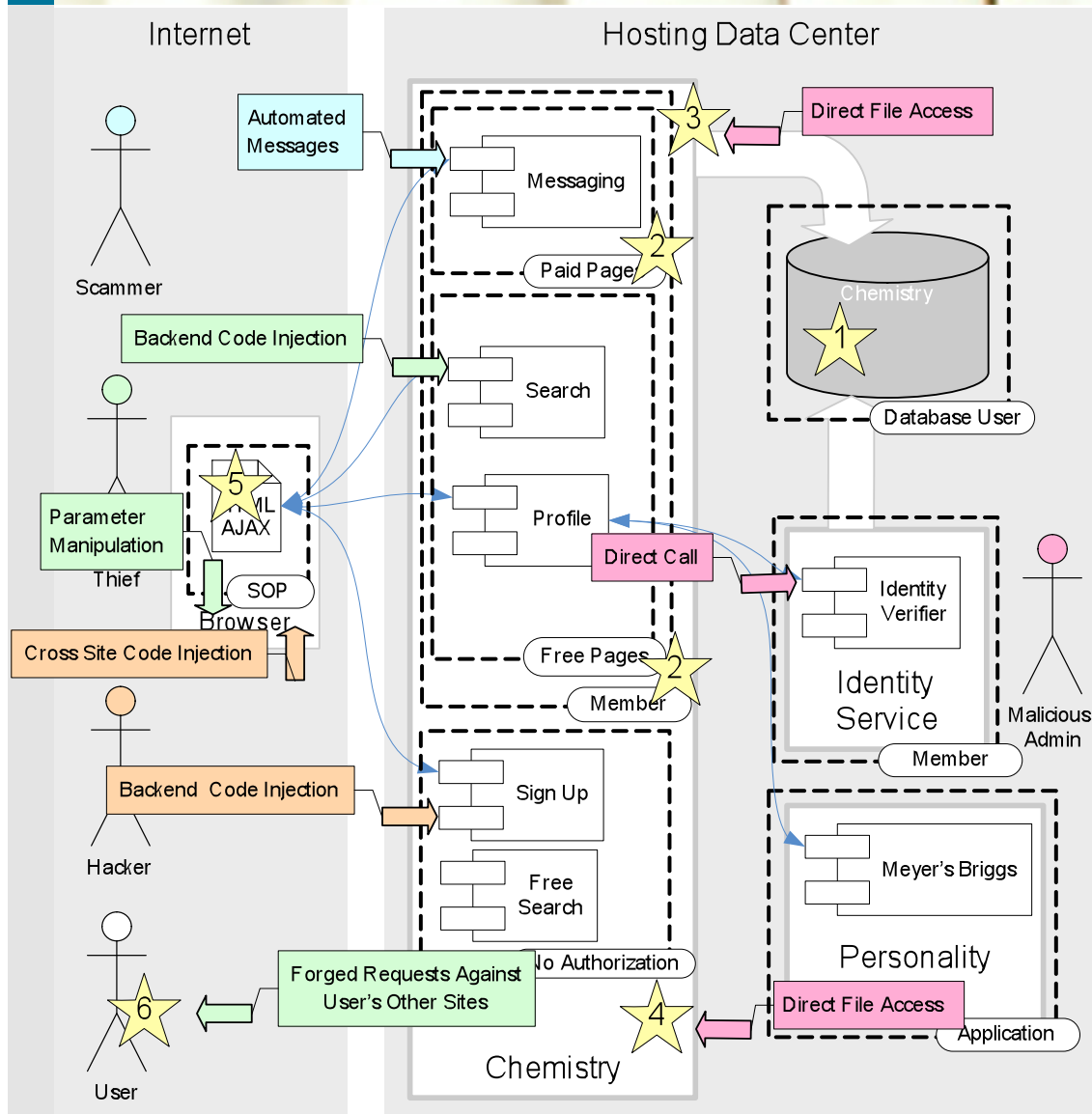
- Threats = malicious users
- Like users, they have capabilities within the system
- Threats have a goal that usually involves subverting a security control or finding a “loophole” in the system

Ex: assets

Application's functions

- Sensitive data
- Data controlling the application's state
- Users and the assets of the other systems the users access



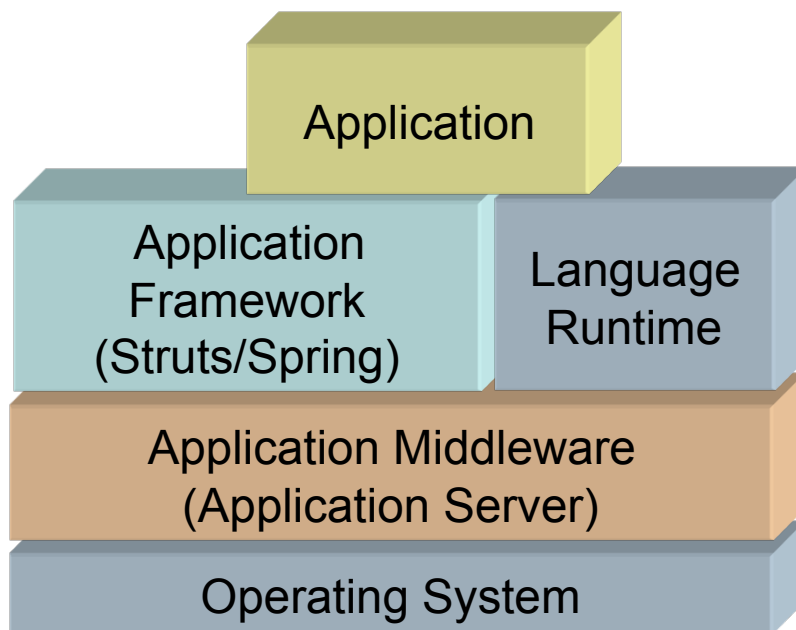


Ex: points of attack

- Associate threat and assets (determine what the attacker can do)
- Ponder nearest, easiest targets first
- Designers: place controls around assets
- Attackers: start with direct attacks and graduate to multi-step

Framework analysis

Software is built upon layers of other software



What kind of flaws exist?

- Known vulnerabilities in open-source or product versions
- Weak security controls provided with the framework
- Framework features that must be disabled or configured to their secure form

Framework analysis: interfaces & contracts

- Place components or application relative to dependencies
 - It is important to see the relationship of an application or component with other callers of shared code and data
- Identify libraries and secure library versions
- Show runtime in diagram where there are security implications:
 - Framework controls
 - VM or other security sandboxes
 - Client-side runtime

Framework security controls

- The application environment provides controls. What are the limitations?
 - Cryptography
 - Example: JCA
 - Authentication and Authorization
 - Example: JAAS
 - Input Validation and Output Encoding
 - .NET validateRequest
 - Sandboxing
 - JavaScript Same Origin Policy

Combine risks and rank

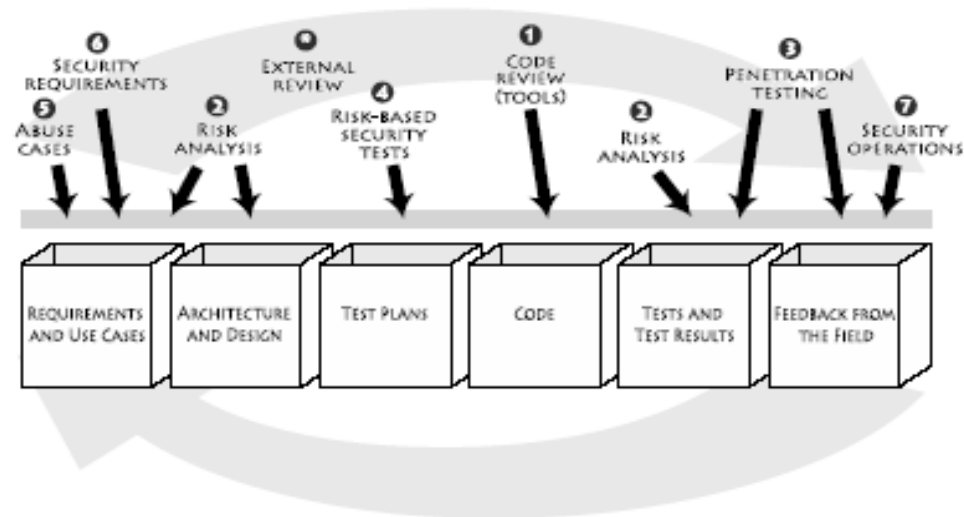
- Take all of your findings and consider business impact
- Rank the findings
- Come up with solutions

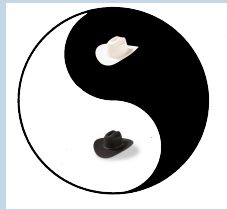
- See chapter 5 of “Software Security”
- <http://www.informit.com/articles/article.asp?p=446451>



Touchpoints adoption

- Code review
 - Widespread
 - Customized tools
 - Training
- ARA
 - Components help
 - Apprenticeship
 - Training
- Pen testing
 - No longer solo
- Security testing
 - Training
- Abuse cases and security requirements
 - Training





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 Miguez
 - Craig Miller
 - John Steven



IEEE Security & Privacy Magazine + 2 Podcasts



The Silver Bullet Security Podcast with Gary McGraw

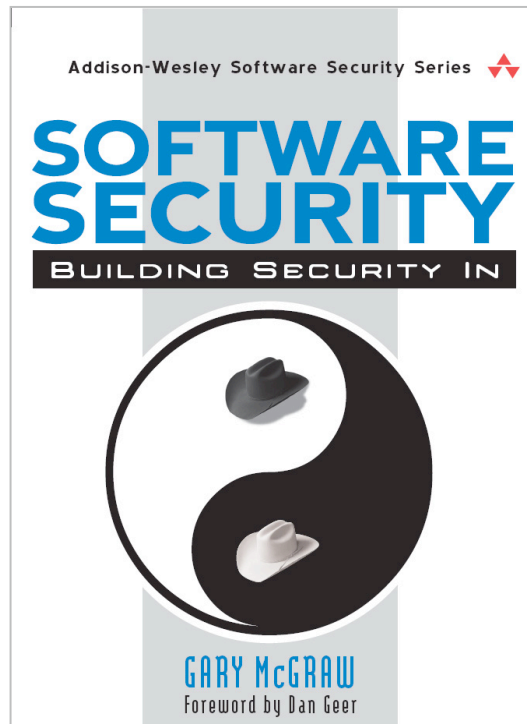


- www.cigital.com/silverbullet
- www.cigital.com/realitycheck

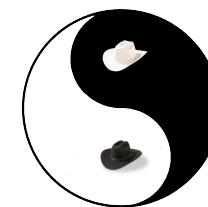
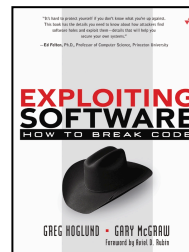
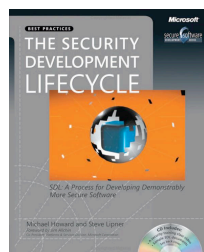
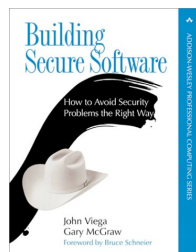
- Building Security In
- Software Security Best Practices column edited by John Steven
- www.computer.org/security/bsisub/



Software Security: the book



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

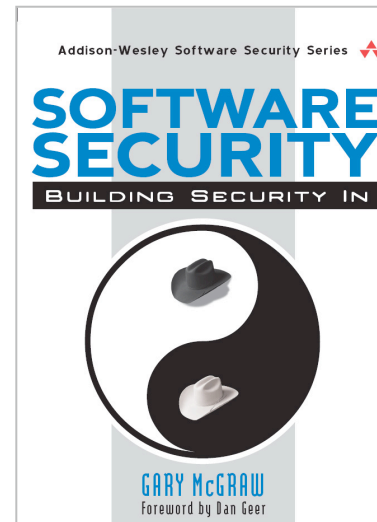




- Cigital's Software Security Group invents and delivers Software Quality Management
- **WE NEED GREAT PEOPLE**
- See the Addison-Wesley Software Security series
- Send e-mail: gem@cigital.com

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

-Bill Gates



For more

cigital

