



How to start a software security initiative within your organization: a maturity based and metrics driven approach

Marco Morana
OWASP Lead/ TISO Citigroup

OWASP

Application Security For
E-Government



MEF

Ministero dell'Economia e delle Finanze

Copyright © 2009 - The OWASP Foundation
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License.

The OWASP Foundation
<http://www.owasp.org>

Presentation Agenda

- Rationale For Building Secure Software
 - ❑ Compliance, cyber-threats, defect management costs, analysts
- Avenues to the Software Security Initiative
 - ❑ Step 1: From Info-Sec to App-Sec and
 - ❑ Step 2: From App-Sec to Software-Sec
 - ❑ Step 3: From tactical activities to strategic plans
- Software Security Initiative Roadmap
 - ❑ Software security maturity, S-SDLCs, Metrics & Measurements
- Questions & Answers

The Rationale for Building Secure Software

■ Some good reasons:

- ▶ **Compliance with technology security standards requires either secure code reviews or deployment of WAF** (e.g. PCI-DSS section 6.6)
- ▶ **Data breaches exploit vulnerabilities in applications with root causes in unsecure software**
- ▶ **Secure code reviews increase the level of software security assurance**
- ▶ **Cheaper to fix bugs then patching**
- ▶ **Fixing security bugs eliminates most of application security issues**

Factors Pointing To Fixing Insecure Software

Go Fix Security Bugs!



```
End Sub  
Private Sub tbToolBar_ButtonClicked  
On Error Resume Next  
timTimer.Enabled = True  
Select Case Button.Key  
Case "Back"  
    brwWebBrowser.GoBack  
Case "Forward"  
    brwWebBrowser.GoForward  
Case "Refresh"  
    brwWebBrowser.Refresh  
Case "Home"  
    brwWebBrowser.Home  
End Select  
End Sub
```

VISA



What PCI-DSS Compliance say?

■ [PCI-DSS] 6 Develop and Maintain Secure Systems and Applications

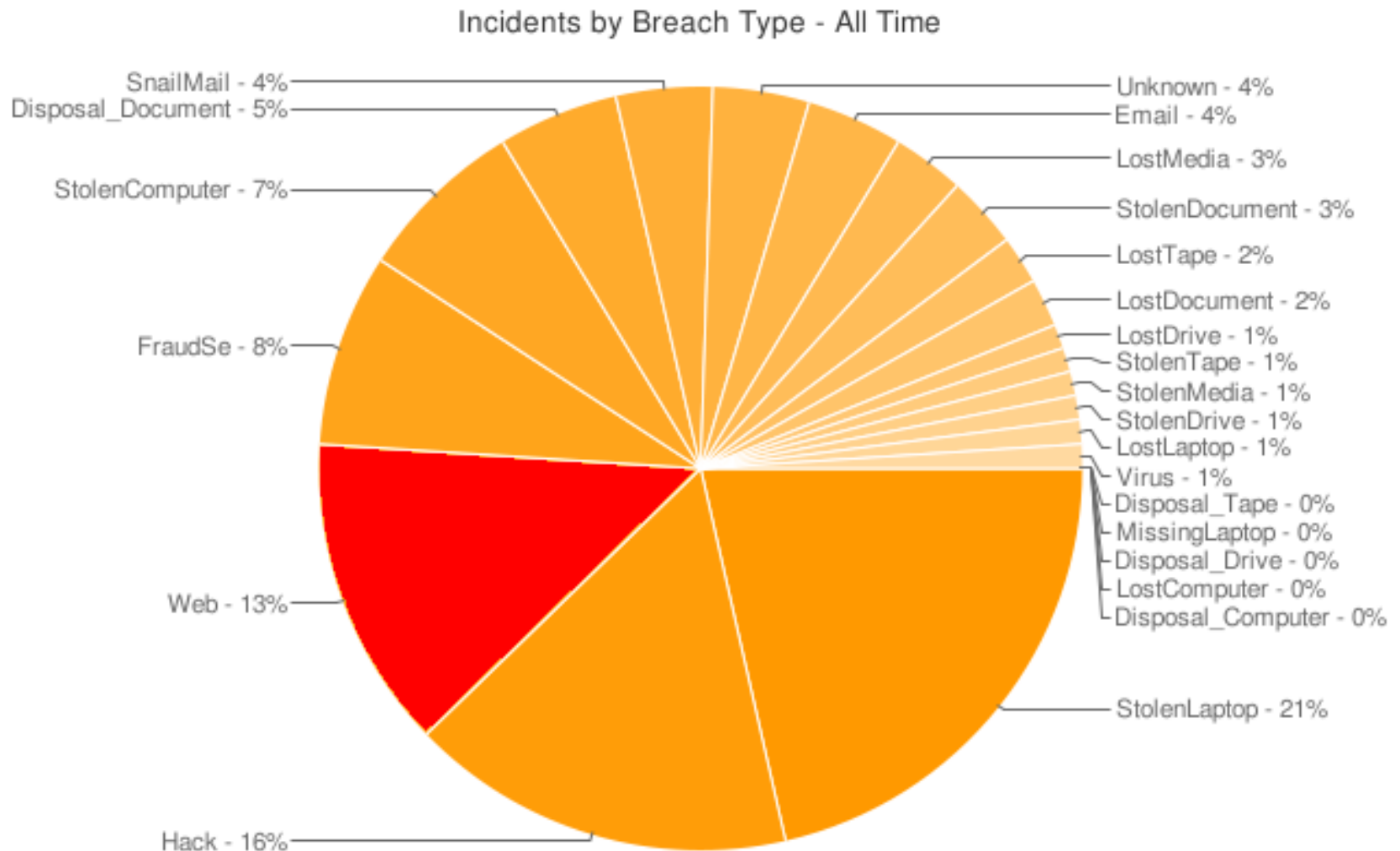
- ▶ All vulnerabilities must be corrected.
- ▶ The application must be re-evaluated after the corrections.
- ▶ The application firewall must detect and prevent web based attacks such as cross site scripting and SQL injection.

■ [PCI-DSS] 11 Regularly Test Security Systems and Processes

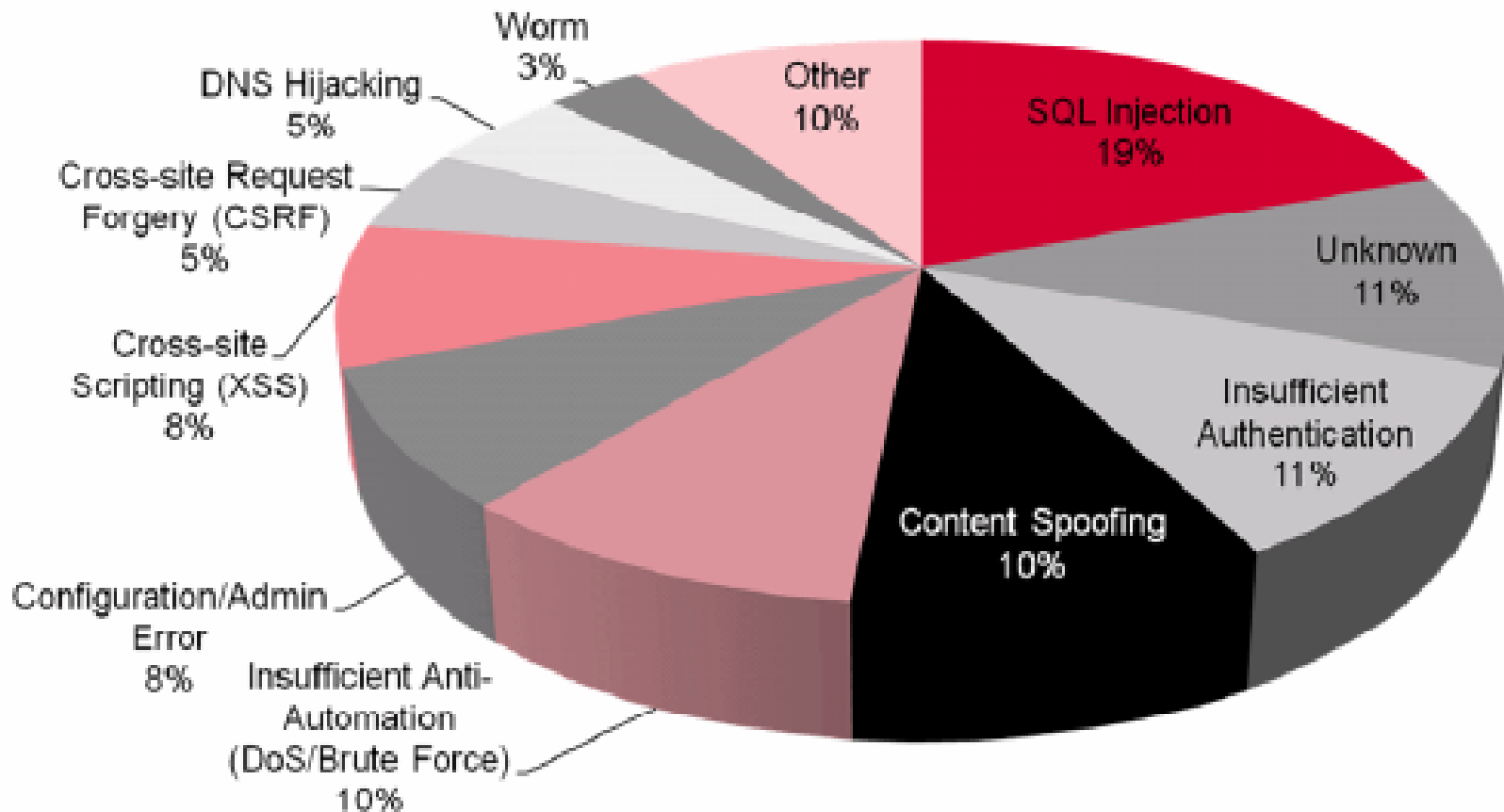
■ [PCI-DSS] 11.3.2 External application layer penetration test.

- ▶ For web applications, the tests should include, at a minimum, the following vulnerabilities: OWASP T10

What Data Breaches Stats Say ?



Which Vulnerabilities Are Most Exploited? (WHID)



SOURCE: Breach Security The WHID 2009, August 2009

What the “experts” say ?

- **“75% of security breaches happen at the application” - *Gartner***
- **“Over 70 percent of security vulnerabilities exist at the application layer, not the network layer” – *Gartner***
- **“If only 50 percent of software vulnerabilities were removed prior to production ... costs would be reduced by 75 percent” - *Gartner***
- **92 % of reported vulnerabilities are in applications not in networks - *NIST***
- **The cost of fixing a bug in the field is \$30,000 vs. \$5,000 during coding - *NIST***

What do you say ? What is Your Company Culture?



What we covered so far..

- Rationale For Building Secure Software
 - ✓ Compliance, cyber-threats, defect management costs, analysts
- Avenues to the Software Security Initiative
 - Step1: From Info-Sec to App-Sec and
 - Step 2: From App-Sec to Software-Sec
 - Step 3: From tactical activities to strategic plans
- Software Security Initiative Roadmap
 - Software security maturity, S-SDLCs, Metrics & Measurements
- Questions & Answers

Step 1: From Information Security To Application Security

- Provision Applications Security In Compliance With Information Security Standards, Processes and IS Risk Management
 - ▶ Protection of Confidentiality, Integrity and Availability leads to enforcement of application security controls (e.g. encryption, auditing and logging, authentication, authorization)
 - ▶ Validate that are no gaps in implementation of security lead to vulnerability assessments
 - ▶ High and Medium risk vulnerabilities remediated before are released in the production environment lead to risk mitigation, acceptance, transfer

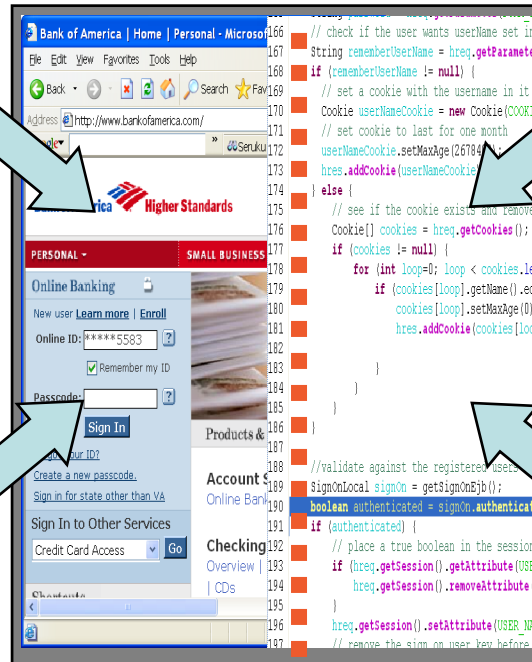
Step 2: From Application Security to Software Security Assessments

**Manual
Penetration
Testing**

**Manual
Code
Review**

**Automated
Vulnerability
Scanning**

**Automated
Static Code
Analysis**

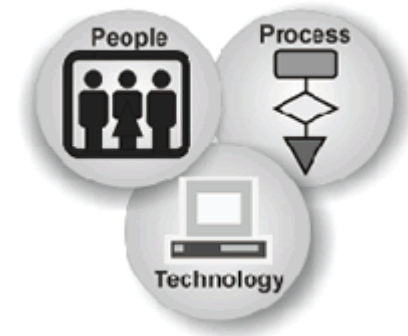
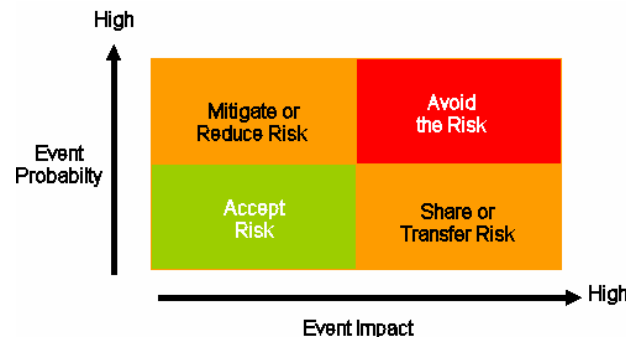
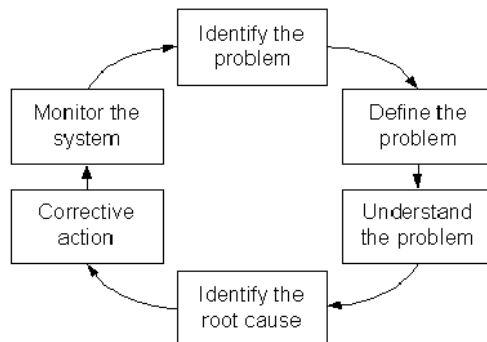


Step 3 : From Tactical To Strategic Activities

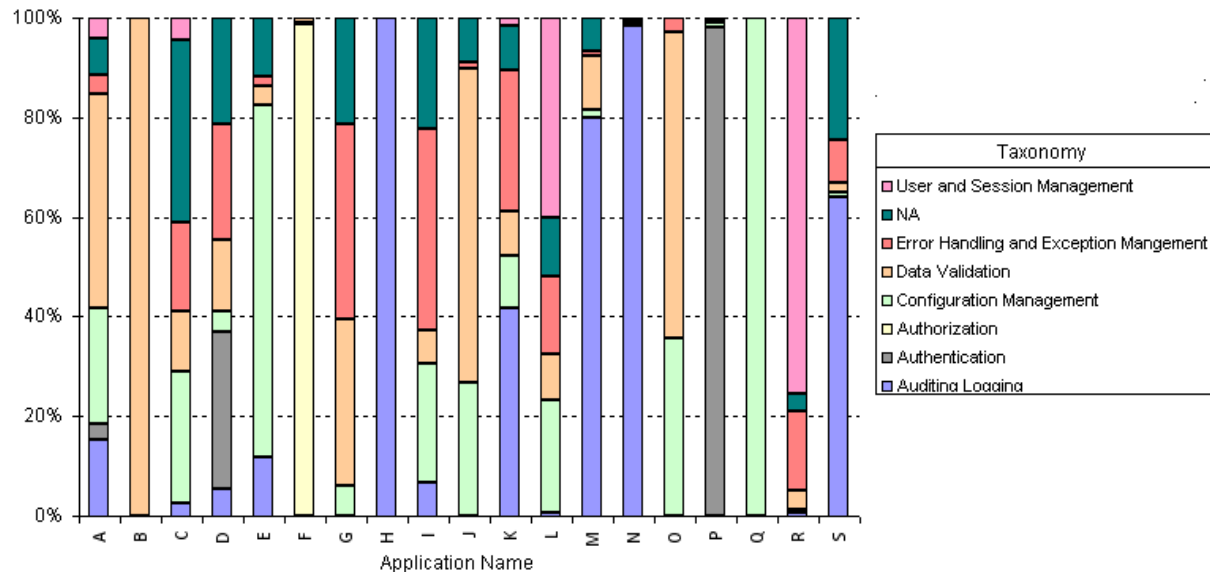
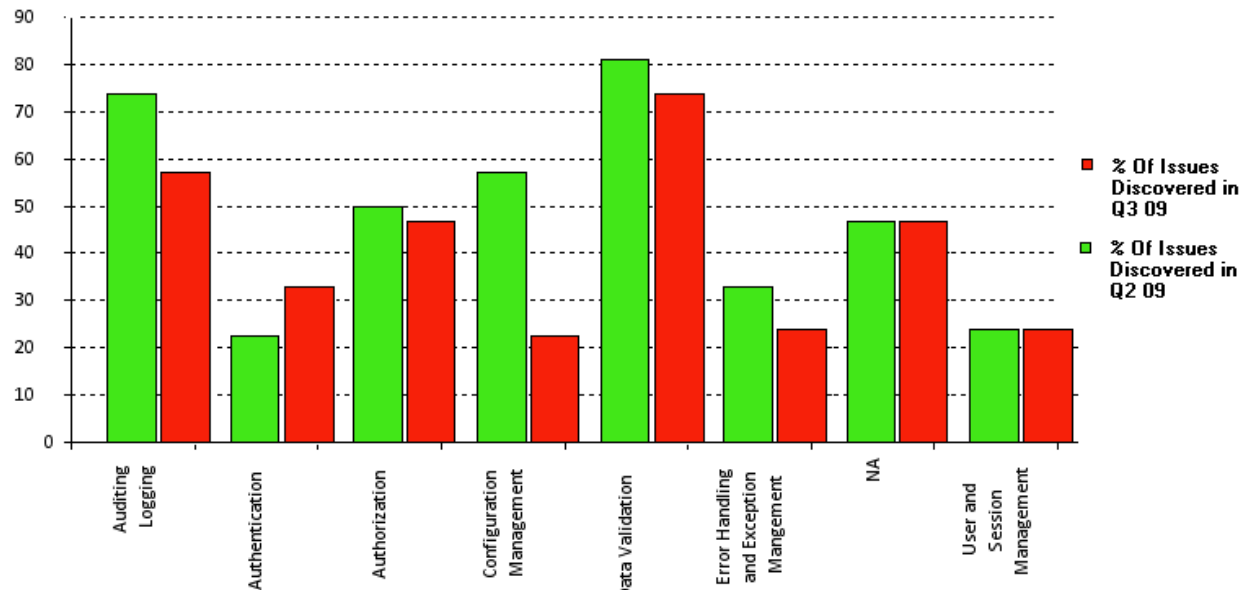
From: Reactive Security, Pen Tests, Catch and Patch



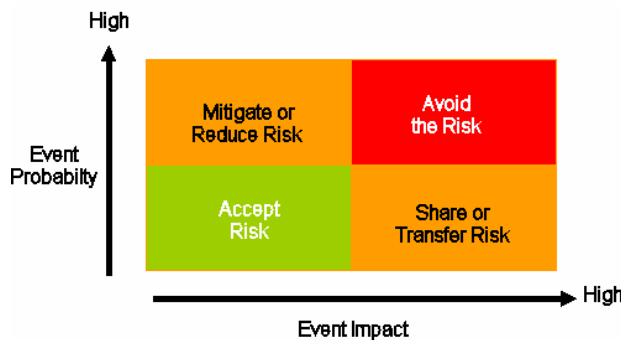
To: Issue Analysis, Risk Analysis, Holistic Security



Examples of Vulnerability Management Metrics



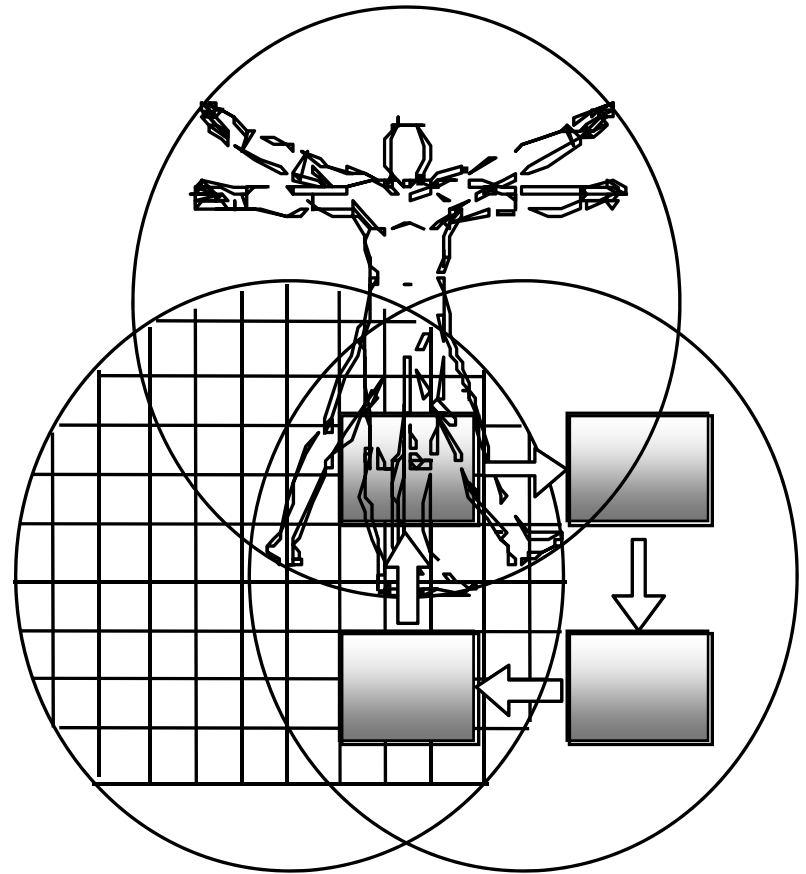
Software Security Engineering & Risk Management



SDLC Phases	Requirements		Design	Development	Testing	Deployment and Operations	
Secure Software Best Practices	Preliminary Software Risk Analysis	Security Requirements Engineering	Security Risk-Driven Design	Secure Code Implementation	Security Tests	Security Configuration & Deployment	Secure Operations
Ongoing S-SDLC Activities Metrics and Measurements, Training, and Awareness							
S-SDLC Activities	Define Use & Misuse Cases	Define Security Requirements	Secure Architecture & Design Patterns Threat Modeling Security Test Planning Security Architecture Review	Peer Code Review Automated Static and Dynamic Code Review Security Unit Tests	Functional Test Risk Driven Tests Systems Tests White Box Testing Black Box Testing	Secure Configuration Secure Deployment	
Other Disciplines	High-Level Risk Assessments		Technical Risk Assessment				Incident Management Patch Management

Systemic Solution To Insecure Software: People, Process and Technology

- Train developers with software security
- Implement secure coding standards and design patterns
- Deploy software security assessment and management tools



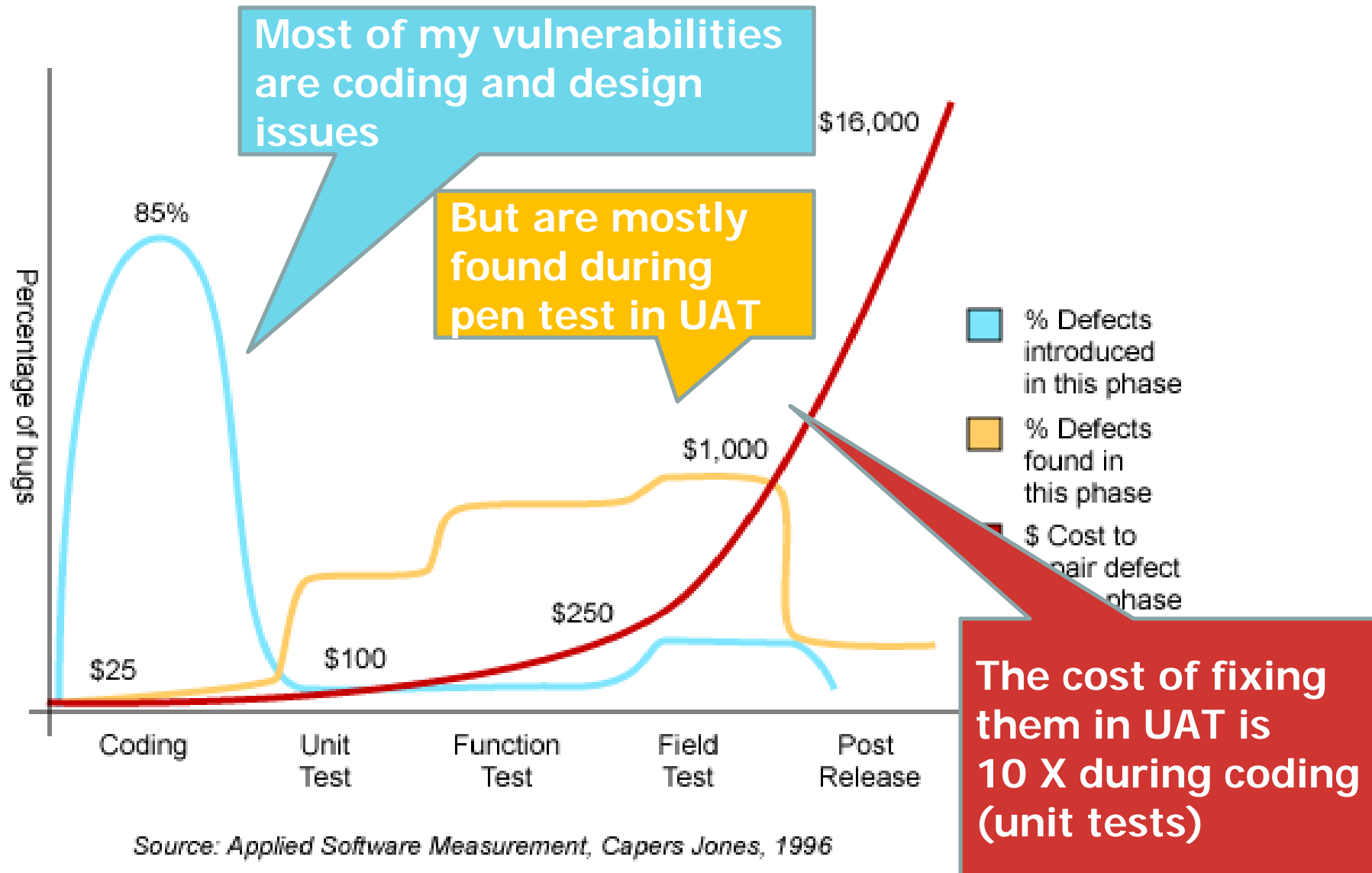
What we covered so far..

- Rationale For Building Secure Software
 - ✓ Compliance, cyber-threats, defect management costs, analysts
- Avenues to the Software Security Initiative
 - ✓ Step1: From Info-Sec to App-Sec and
 - ✓ Step 2: From App-Sec to Software-Sec
 - ✓ Step 3: From tactical activities to strategic plans
- **Software Security Initiative Roadmap**
 - **Software security maturity, S-SDLCs, Metrics & Measurements**
- Questions & Answers

Making the Case for Software Security

- **Make executives aware of how software security effort compares to everyone else's**
 - ▶ Assess capabilities and make them visible
 - ▶ Point to goals and activity needs to reach them
 - ▶ Provide the context for software security activities
- **Use available metrics to articulate software security needs/opportunities**
 - ▶ Analyze vulnerability assessment processes and data
 - ▶ Point to software security root causes
 - ▶ Identify historical vulnerability gaps and trends
 - ▶ Prepare a plan for software security improvements

What Your Defect Management Metrics Say?



Prepare a Roadmap For Software Security

- 1. Assess the maturity** of the organization software security development processes, people and tools
- 2. Define the software security process:** security enhanced SDLCs, frameworks and checkpoints
- 3. Start software security engineering push**
 1. Security Requirements
 2. Secure Design and Threat Modeling
 3. Secure Coding Guidelines and Security Code Review
 4. Security Testing
 5. Secure Deployment
- 4. Collect defect and vulnerability metrics**
- 5. Optimize and improve software security processes**

Software Security Mapped to CMM

■ Initial to Repeatable: From CMM Level 1 to Level 2

- ▶ Penetrate and patch ad-hoc approach
- ▶ Some applications undergo penetration tests before production release and every year after release

■ Defined to Managed: From CMM Level 2 to Level 3

- ▶ Vulnerability assessments are tracked and managed
- ▶ Source code is reviewed for security
- ▶ Penetration tests validate issues dealt with earlier in the SDLC with source code analysis

■ Managed to Optimizing: From CCM Level 4 to Level 5

- ▶ Software risks assessed in each phase of the SDLC
- ▶ Risk metrics and measurements are used for improving security engineering and risk management processes

Capability Maturity Models (CMM)

Visibility Into The Software Security Process

Maturity Level

- New opportunities for improve software security with adoption of new tools/process are identified during design, development and tests
- Inefficient security activities are identified and replaced

CMM 5: Optimizing
Continuous process improvement. Focus on Investment in process automation

- Security is able to measure progress in detecting and fixing vulnerabilities earlier in the SDLC
- Software risks are managed during the SDLC
- Engineering is able to management costs for vulnerabilities

CMM 4: Managed
Quantitative process control. Focus on product and process quality

- Vulnerability Assessment process defined for all applications
- Source code analysis process for critical applications
- Vulnerability metrics used for risk assessment

CMM 3: Defined
Qualitative process control. Focus on process and organization, proactive process management

- Vulnerability scans for High risk applications
- Security Coding Standards documented
- Source Code Analysis tools tested

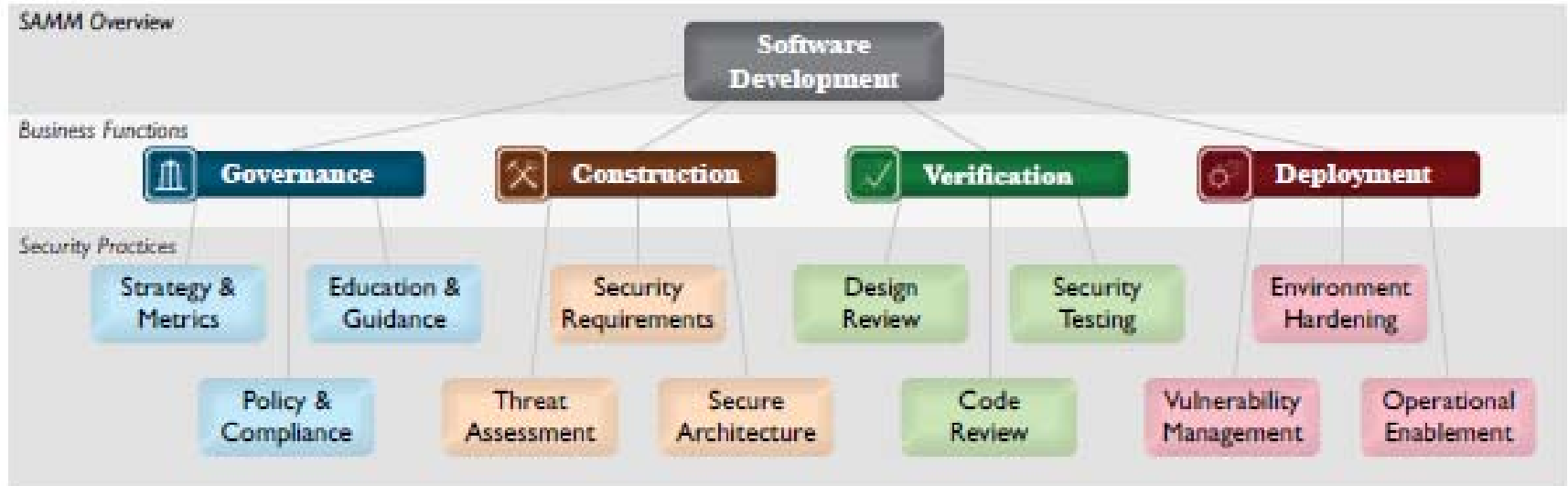
CMM 2: Repeatable
Intuitive process control, Focus on project administration, mostly reactive process management

- Security Issues addressed by patching existing applications
- Hire security consulting company to perform ethical hacking/pen testing

CMM 1: Initial
Ad hoc process, Focus on Individual Competence, Unpredictable Cost, Schedule and Quality



Software Security Maturity Models: SAMM, BSIMM



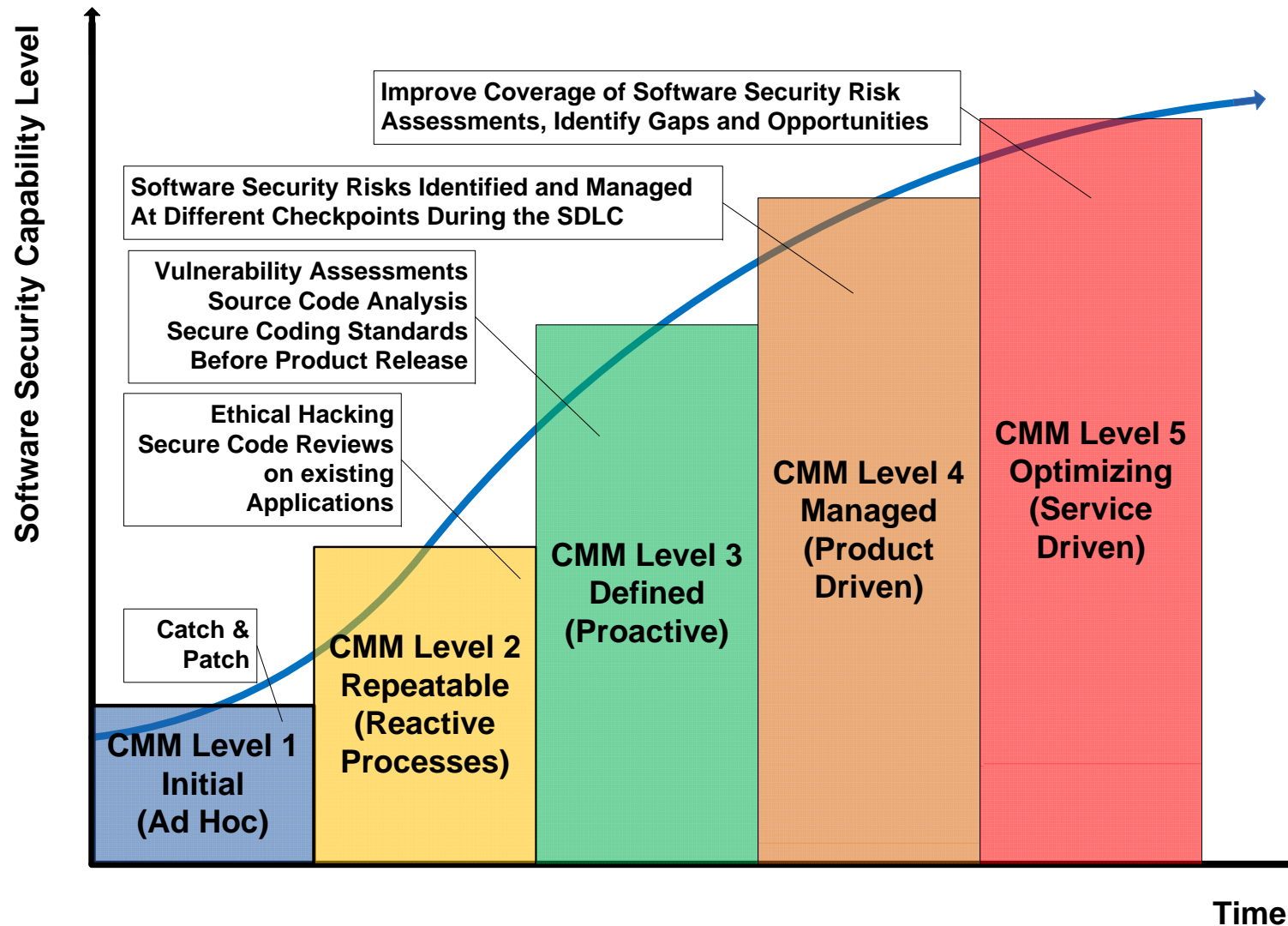
The Software Security Framework (SSF)			
Governance	Intelligence	SSDL Touchpoints	Deployment
Strategy and Metrics	Attack Models	Architecture Analysis	Penetration Testing
Compliance and Policy	Security Features and Design	Code Review	Software Environment
Training	Standards and Requirements	Security Testing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

Code Review Activities And Capability Levels: BSIMM

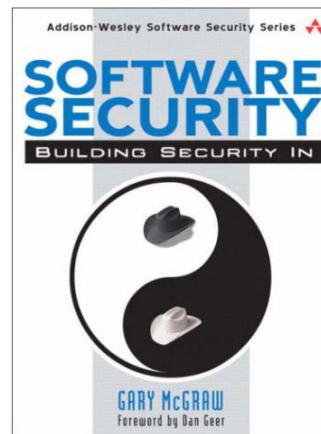
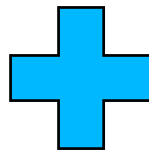
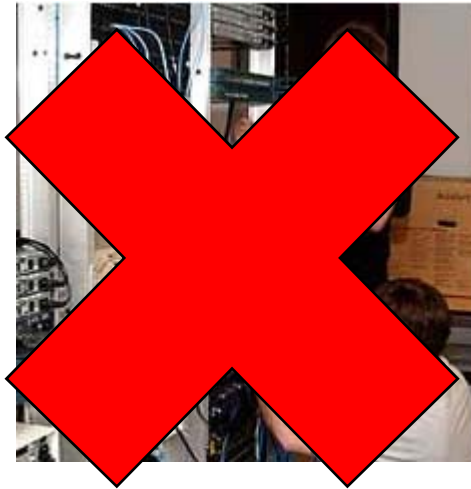
SSDL TOUCHPOINTS: CODE REVIEW			
Use of code review tools, development of customized rules, profiles for tool use by different roles, manual analysis, ranking/measuring results.			
	Objective	Activity	Level
CR1.1	know which bugs matter to you	create top N bugs list (real data preferred) (T: training)	1
CR1.2	review high-risk applications opportunistically	have SSG perform ad hoc review	
CR1.3	spread software security around without any process	establish coding labs or of review	
CR2.1	drive efficiency/consistency with automation	use automated tools along with manual review	2
CR2.2	drive behavior objectively	enforce coding standards	
CR2.3	find bugs earlier	make code review mandatory for all projects	
CR2.4	know which bugs matter (for training)	use centralized reporting (close knowledge loop, drive training) (T: strategy/metrics)	
CR2.5	make most efficient use of tools	assign tool mentors	
CR3.1	drive efficiency/reduce false positives	use automated tools with tailored rules	3
CR3.2	combine assessment techniques	build a factory	
CR3.3	handle new bug classes in an already scanned codebase	build capability for eradicating specific bugs from entire codebase	

I am here

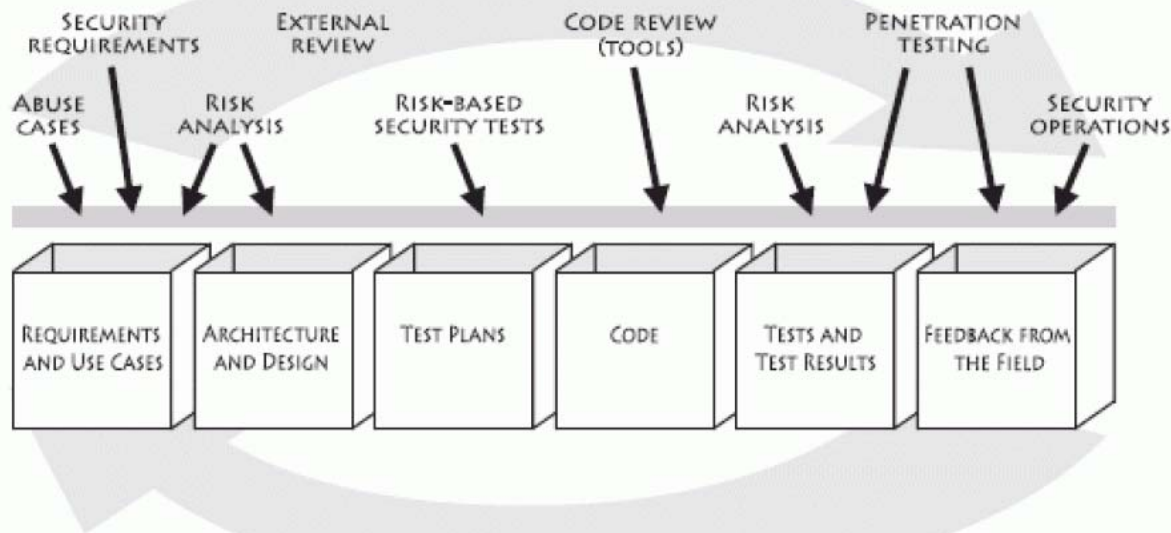
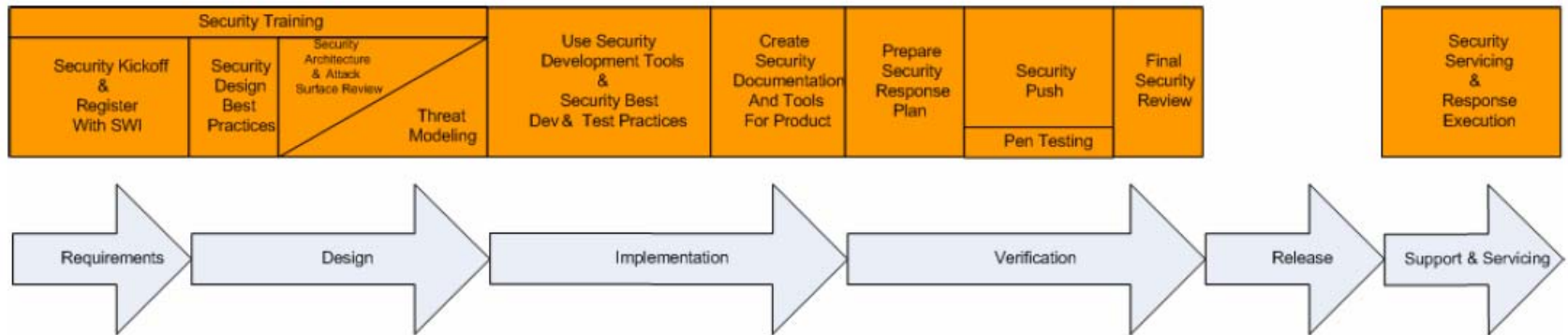
Software Security Maturity Curve



Pre-requisite for Software Security Initiative: Hiring The Right People



Security-enhanced lifecycle process (S-SDLC) models: MS-SDL, Cigital TP and CLASP

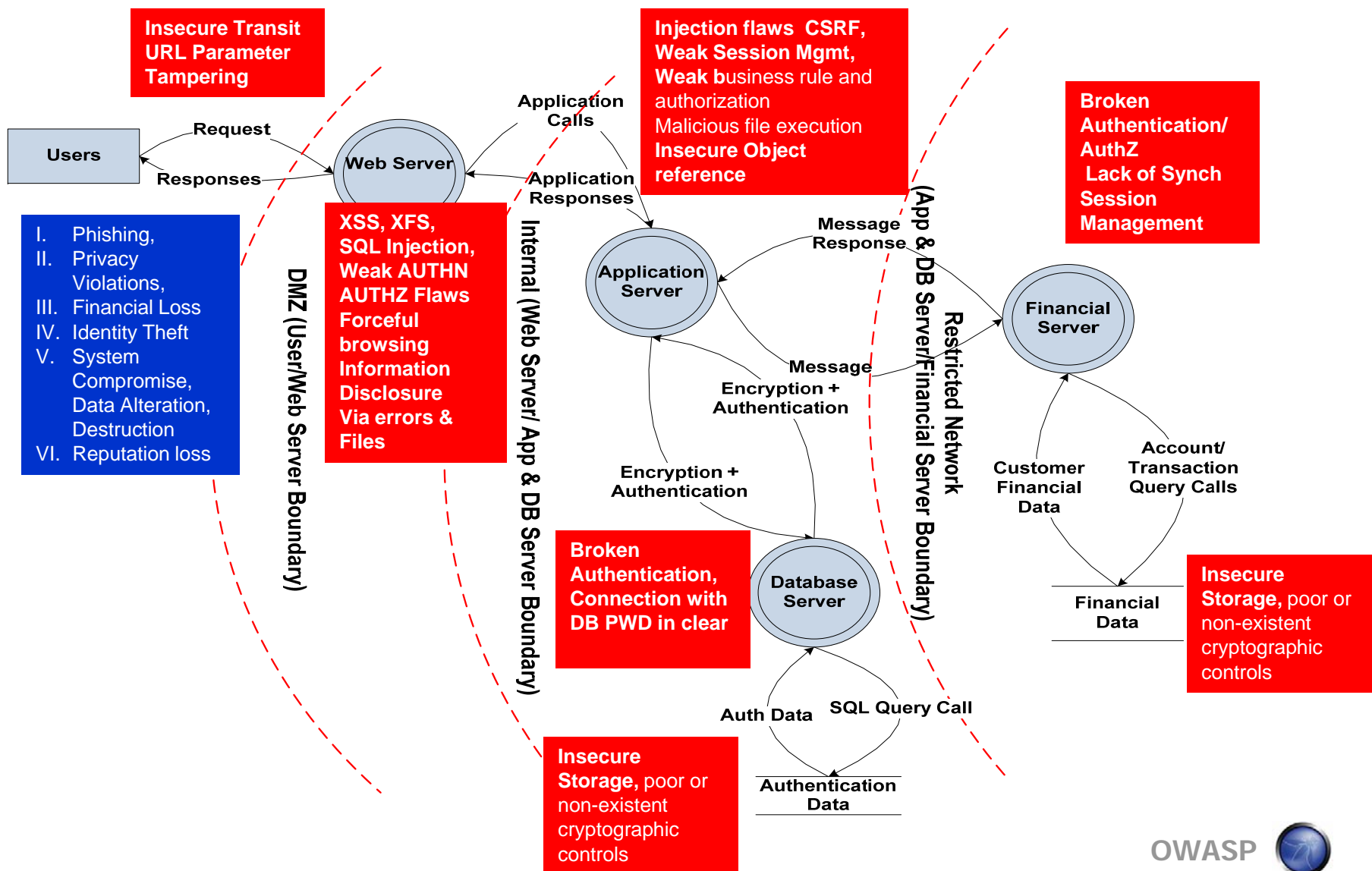


CLASP BEST PRACTICES

- 1) Institute awareness programs
- 2) Perform application assessments
- 3) Capture security requirements
- 4) Implement secure development practices
- 5) Build vulnerability remediation procedures
- 6) Define & monitor metrics
- 7) Publish operational security guidelines



Application Threat Modeling



Application Security Defect Tracking and Metrics

- Define where and how security metrics is collected
- Tracking security defects throughout the SDLC
 - ▶ Report the root causes: requirements, design, code, application
 - ▶ Report the type of the issues, the severity and whether has been fixed or no
- Metrics
 - ▶ What lifecycle stage are most flaws originating in?
 - ▶ What security mechanisms/controls are we having trouble implementing?
 - ▶ What security vulnerabilities are we having trouble fixing?

Vulnerability Root Cause Analysis

- The source of vulnerabilities can be lack of requirements, design flaw, coding error, mis-configuration
- Root causes are identified with different assessments and support focused remediation, risk prioritization and tracking:
 - ▶ Security Design Flaws
 - Introduced because of errors in design
 - Can be identified with threat modeling and manual code reviews
 - ▶ Security Coding Bugs
 - Coding errors that result in vulnerabilities
 - Can be identified with secure code reviews and/or tools

Examples of Application Security Metrics

Process Metrics

- Is a SDL is used are security gates enforced?
- Is code validated against security standards?
- Security posture of a new application before delivery
 - ▶ Security Officers Sign Off?
 - ▶ Test For Security Vulnerabilities Executed?
 - ▶ All high risk issues closed?
 - ▶ Risk assessments completed?
- % of developers trained, using organizational security best practice technology, architecture and processes

Management Metrics

- % of applications rated “business-critical” that have been security tested
- % of projects that where developed with the SDL
- % of security issues identified by lifecycle phase
- % of issues whose risk has been accepted
- % of security issues being fixed
- Average time to correct vulnerabilities
- Business impact of critical security incidents.









Examples of Software Security Metrics: OWASP T10

PCI Requirement 6.5 requires merchants to:

Develop Web software and applications based on secure coding guidelines such as the Open Web Application Security Project guidelines, review custom application code to identify coding vulnerabilities and over prevention of common coding vulnerabilities in software development processes, to include:

- Unvalidated input
- Broken access control
- Broken authentication/session management
- Cross site scripting attacks
- Buffer overflows Injection flaws (e.g. SQL
- Improper error handling
- Insecure storage
- Denial of service
- Insecure configuration management

OWASP 2004 Top 10/PCI Section 6.5 Compliance Category

	Hot	Warning	Info	Total
 A1. Unvalidated Input	0	15	0	15
 A2. Broken Access Control	0	0	0	0
 A3. Broken Authentication and Session Management	0	0	0	0
 A4. Cross-Site Scripting (XSS) Flaws	2	0	0	2
 A5. Buffer Overflows	0	0	0	0
 A6. Injection Flaws	5	0	0	5
 A7. Improper Error Handling	0	1	10	11
 A8. Insecure Storage	0	0	0	0
 A9. Application Denial of Service	0	17	0	17
 A10. Insecure Configuration Management	0	8	0	8
Total	7	41	10	58

<http://www.issa-centralva.org/presentations/SecurityMetrics09122007.pdf>

Security Metrics Goals The Good and The Bad

- **Good:** if goals when are “SMART” that is Specific, Measurable, Attainable, Realistic, Traceable and Appropriate
 - ▶ Example: reducing the overall number of vulnerabilities by 30% by fixing all low hanging fruits with source code analysis during construction
- **Bad:** if the goals justify the means to obtain the goals



Ensure Support For The Initiative Moving Forward

- Tie the metrics to the business cases and support the project stakeholders agendas:
 - ▶ **Developer Leads:** show that developers are getting better to write secure software
 - ▶ **Project Managers:** shows that projects are on schedule and moving on target and testing cycles for vulnerabilities are shorter translating in cost savings
 - ▶ **Information Security Officers:** show that we are getting better on reporting compliance and manage risk reduction

In summary..

- Rationale For Building Secure Software
 - ✓ Compliance, cyber-threats, defect management costs, analysts
- Avenues to the Software Security Initiative
 - ✓ Step1: From Info-Sec to App-Sec and
 - ✓ Step 2: From App-Sec to Software-Sec
 - ✓ Step 3: From tactical activities to strategic plans
- Software Security Initiative Roadmap
 - ✓ Software security maturity, S-SDLCs, Metrics & Measurements
- Questions & Answers

QUESTIONS ANSWERS



Thanks for listening, further references

■ Applied Software Measurement: Assuring Productivity and Quality

- ▶ <http://www.amazon.com/Applied-Software-Measurement-Assuring-Productivity/dp/0070328269>

■ PCI-Data Security Standard (PCI DSS)

- ▶ https://www.pcisecuritystandards.org/security_standards/pci_dss.shtml

■ A CISO's Guide to Application Security

- ▶ http://www.nysforum.org/committees/security/051409_pdfs/A%20CISO'S%20Guide%20to%20Application%20Security.pdf

Further references con't

■ Gartner 2004 Press Release

- ▶ http://www.gartner.com/press_releases/asset_106327_11.html

■ Software Assurance Maturity Model

- ▶ <http://www.opensamm.org/>

■ The Software Security Framework (SSF)

- ▶ <http://www.bsi-mm.com/ssf/>

■ SEI Capability Maturity Model Integration CMMi

- ▶ <http://www.sei.cmu.edu/cmml/>

■ The Microsoft Security Development LifeCycle

- ▶ <http://msdn.microsoft.com/en-us/security/cc448177.aspx>

Further references con't

■ The Seven Touchpoints of Software Security

- ▶ <http://www.buildsecurityin.com/concepts/touchpoints/>

■ OWASP CLASP

- ▶ http://www.owasp.org/index.php/Category:OWASP_CLASP_Project

■ ITARC Software Security Assurance

- ▶ <http://iac.dtic.mil/iatac/download/security.pdf>

■ Internet Crime Compliant Center

- ▶ <http://www.ic3.gov/default.aspx>

Further references con't

- OWASP Education Module Embed within SDLC
 - ▶ http://www.owasp.org/index.php/Education_Module_Embed_within_SDL_C
- Producing Secure Software With Software Security Enhanced Processes
 - ▶ <http://www.net-security.org/dl/insecure/INSECURE-Mag-16.pdf>
- Security Flaws Identification and Technical Risk Analysis Through Threat Modeling
 - ▶ <http://www.net-security.org/dl/insecure/INSECURE-Mag-17.pdf>