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

Marco Morana
OWASP Lead/ TISO Citigroup

Application Security For E-Government

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!
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?

Incidents by Breach Type - All Time

- StolenLaptop - 21%
- Hack - 16%
- Web - 13%
- FraudSe - 8%
- StolenComputer - 7%
- Disposal_Document - 5%
- SnailMail - 4%
- Unknown - 4%
- Email - 4%
- LostMedia - 3%
- StolenDocument - 3%
- LostTape - 2%
- LostDocument - 2%
- LostDrive - 1%
- StolenTape - 1%
- StolenMedia - 1%
- StolenDrive - 1%
- LostLaptop - 1%
- Virus - 1%
- Disposal_Tape - 0%
- MissingLaptop - 0%
- Disposal_Drive - 0%
- LostComputer - 0%
- Disposal_Computer - 0%
Which Vulnerabilities Are Most Exploited? (WHID)

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?

I BROKE INTO THE WEB SITE TO HELP YOU!

THANKS, BUT IT’S BETTER IF WE DON’T KNOW THE SITE IS INSECURE.
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
- Automated Vulnerability Scanning
- Manual Code Review
- 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

Process Metrics
- Is code validated against security coding standards?
- Is design 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 were developed with the SDL
- % of security issues identified by lifecycle phase
- % of issues whose risk has been accepted
- % of issues being fixed
- Average time to correct vulnerabilities
- Business impact of critical security incidents
Software Security Engineering & Risk Management

Development Process

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
- System Tests
- White Box Testing
- Black Box Testing
- Secure Configuration
- Secure Deployment

Event Probability vs. Impact
- Mitigate or Reduce Risk
- Avoid the Risk
- Accept Risk
- Share or Transfer Risk

Other Disciplines
- High-Level Risk Assessments
- Technical Risk Assessment
- Incident Management
- Patch Management
Systemic Solution To Unsecure 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**
  - 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**
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?

Most of my vulnerabilities are coding and design issues

But are mostly found during pen test in UAT

The cost of fixing them in UAT is 10 X during coding (unit tests)

Source: Applied Software Measurement, Capers Jones, 1996
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
   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

- 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

- 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

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

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

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

Maturity Level

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

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

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

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

CMM 5: Optimizing
Continuous process improvement. Focus on Investment in process automation
Software Security Maturity Models: SAMM, BSI MM

The Software Security Framework (SSF)

<table>
<thead>
<tr>
<th>Governance</th>
<th>Intelligence</th>
<th>SSDL Touchpoints</th>
<th>Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strategy and Metrics</td>
<td>Attack Models</td>
<td>Architecture Analysis</td>
<td>Penetration Testing</td>
</tr>
<tr>
<td>Compliance and Policy</td>
<td>Security Features and Design</td>
<td>Code Review</td>
<td>Software Environment</td>
</tr>
<tr>
<td>Training</td>
<td>Standards and Requirements</td>
<td>Security Testing</td>
<td>Configuration Management and Vulnerability Management</td>
</tr>
</tbody>
</table>
## 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.

<table>
<thead>
<tr>
<th>Objective</th>
<th>Activity</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR1.1 know which bugs matter to you</td>
<td>create top N bugs list (real data preferred) (T: training)</td>
<td>1</td>
</tr>
<tr>
<td>CR1.2 review high-risk applications opportunistically</td>
<td>have SSG perform ad hoc review</td>
<td></td>
</tr>
<tr>
<td>CR1.3 spread software security around without any process</td>
<td>establish coding labs or of course do code review</td>
<td></td>
</tr>
<tr>
<td>CR2.1 drive efficiency/consistency with automation</td>
<td>use automated tools along with manual review</td>
<td>2</td>
</tr>
<tr>
<td>CR2.2 drive behavior objectively</td>
<td>enforce coding standards</td>
<td></td>
</tr>
<tr>
<td>CR2.3 find bugs earlier</td>
<td>make code review mandatory for all projects</td>
<td></td>
</tr>
<tr>
<td>CR2.4 know which bugs matter (for training)</td>
<td>use centralized reporting (close knowledge loop, drive training) (T: strategy/metrics)</td>
<td></td>
</tr>
<tr>
<td>CR2.5 make most efficient use of tools</td>
<td>assign tool mentors</td>
<td></td>
</tr>
<tr>
<td>CR3.1 drive efficiency/reduce false positives</td>
<td>use automated tools with tailored rules</td>
<td>3</td>
</tr>
<tr>
<td>CR3.2 combine assessment techniques</td>
<td>build a factory</td>
<td></td>
</tr>
<tr>
<td>CR3.3 handle new bug classes in an already scanned codebase</td>
<td>build capability for eradicating specific bugs from entire codebase</td>
<td></td>
</tr>
</tbody>
</table>
Software Security Maturity Curve

- **CMM Level 1 Initial (Ad Hoc)**
- **CMM Level 2 Repeatable (Reactive Processes)**
- **CMM Level 3 Defined (Proactive)**
- **CMM Level 4 Managed (Product Driven)**
- **CMM Level 5 Optimizing (Service Driven)**

Tasks:
- **Ethical Hacking**
- **Secure Code Reviews on existing Applications**
- **Vulnerability Assessments**
- **Source Code Analysis**
- **Secure Coding Standards Before Product Release**

- **Software Security Risks Identified and Managed At Different Checkpoints During the SDLC**
- **Improve Coverage of Software Security Risk Assessments, Identify Gaps and Opportunities**

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

[Diagram showing the combination of hiring the right people and security knowledge results in a successfully hired individual]
Security-enhanced lifecycle process (S-SDLC) models: MS-SDL, Cigital TP and CLASP
Application Threat Modeling

Injection flaws  
CSRF,  
Weak Session Mgmt,  
Weak business rule and authorization  
Malicious file execution  
Insecure Object reference

Insecure Transit  
URL Parameter Tampering

XSS, XFS,  
SQL Injection,  
Weak AUTHN  
AUTHZ Flaws  
Forceful browsing  
Information Disclosure  
Via errors & Files

Broken Authentication/ AuthZ  
Lack of Synch Session Management

I. Phishing,  
II. Privacy Violations,  
III. Financial Loss  
IV. Identity Theft  
V. System Compromise,  
Data Alteration,  
Destruction  
VI. Reputation loss

Broken Authentication,  
Connection with DB PWD in clear

Insecure Storage,  
poor or non-existent cryptographic controls

OWASP
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, misconfiguration.

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/authorization management
  - Cross site scripting attacks
  - Buffer overflow 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

<table>
<thead>
<tr>
<th></th>
<th>Hot</th>
<th>Warning</th>
<th>Info</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1 Unvalidated Input</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>A2 Broken Access Control</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A3 Broken Authentication and Session Management</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A4 Cross Site Scripting (XSS) Flaws</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>A5 Buffer Overflows</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A6 Injection Flaws</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>A7 Improper Error Handling</td>
<td>0</td>
<td>1</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>A8 Insecure Storage</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A9 Application Denial of Service</td>
<td>0</td>
<td>17</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>A10 Insecure Configuration Management</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>6</td>
</tr>
</tbody>
</table>

Total: 7 41 10 50

Security Metrics Goals The Good and The Bad

- **Good:** if goals 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**
  - 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**
QUESTIONS
ANSWERS
Thanks for listening, further references

- Applied Software Measurement: Assuring Productivity and Quality

- PCI - Data Security Standard (PCI DSS)

- A CISO's Guide to Application Security
Further references con’t

- Gartner 2004 Press Release

- Software Assurance Maturity Model
  - [http://www.opensamm.org/](http://www.opensamm.org/)

- The Software Security Framework (SSF)
  - [http://www.bsi-mm.com/ssf/](http://www.bsi-mm.com/ssf/)

- SEI Capability Maturity Model Integration CMMI
  - [http://www.sei.cmu.edu/cmmi/](http://www.sei.cmu.edu/cmmi/)

- The Microsoft Security Development LifeCycle
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

■ Internet Crime Compliant Center
Further references con’t

■ OWASP Education Module Embed within SDLC
  ‣ http://www.owasp.org/index.php/Education_Module_Embd_within_SDLC

■ Producing Secure Software With Software Security Enhanced Processes

■ Security Flaws Identification and Technical Risk Analysis Through Threat Modeling