Building & Measuring Security in Web Applications

Fabio Cerullo
Cycubix Limited
30 May 2012 - Belfast
Brief Bio

- CEO & Founder Cycubix Limited
- 10+ years security experience in Technology, Manufacturing, Financial & Government sectors.
- MSc in Computer Engineering.
- Certified Information Security Professional (CISSP)
- Certified Secure Software Lifecycle Professional (CSSLP)
- OWASP Global Education Committee
- OWASP Ireland Chapter Leader
- OWASP AppSec Ireland Conference Chair
Agenda

- Introduction to Application Security
- Building Secure Software
- A phased approach to AppSec
- Conclusions
- Q & A
Introduction to Application Security

What is Application INSecurity?
A principle-based approach for application security involves:
- Knowing your threats (People)
- Incorporating security in your SDLC (Process)
- Securing the network, host and APPLICATION (Technology)
Introduction to Application Security

Risk Theory

RISK is a path from Threat Agent to Business Impact
Every vulnerability originates from:

**Missing Control**
- Lack of Input Validation
- Failure to perform Access Control

**Broken Control**
- Improper Session Handling
- Fail Open

**Ignored Control**
- Failure to implement Encryption
- Forgot to use Output Encoding

ESAPI COULD HELP YOU HERE

NOBODY COULD HELP YOU HERE
Introduction to Application Security

TOP 10 WEB APPLICATION SECURITY RISKS

A1: Injection
A2: Cross Site Scripting (XSS)
A3: Broken Authentication and Session Management
A4: Insecure Direct Object References
A5: Cross Site Request Forgery (CSRF)
A6: Security Misconfiguration
A7: Failure to Restrict URL Access
A8: Unvalidated Redirects and Forwards
A9: Insecure Cryptographic Storage
A10: Insufficient Transport Layer Protection

http://www.owasp.org/index.php/Top_10

OWASP
The Open Web Application Security Project
http://www.owasp.org
A1. Injection

DEFINITION
Injection flaws occur when an application sends untrusted data to an interpreter.

EXAMPLE: SONY MUSIC JAPAN (MAY 2011)

A SQL error implies a vulnerable application to SQL Injection

http://www.sonymusic.co.jp/bv/cro-magnons/track.php?item=7419 union all
select 1,concat(user,0x3a,pass,0x3a,email) from users // what we get here is
user:pass:email from table users. (0x3a is hex value for colon)

IMPACT: SONY PICTURES (JUNE 2011)
Over 1,000,000 users' passwords, email addresses, home addresses, dates of birth,
as well as administrator login passwords.

HOW TO PREVENT IT?
• Avoid the usage of the interpreter by implementing stored procedures or parameterized queries.
• Escape special characters using API’s like OWASP ESAPI.
• Perform positive or whitelist input validation with proper canonicalization.
A2. Cross Site Scripting (XSS)

**DEFINITION**
XSS flaws occur when an application includes user supplied data in a page sent to the browser without properly validating or escaping that content. There are three known types of XSS flaws: 1) Stored, 2) Reflected, and 3) DOM based XSS

**EXAMPLE**
- Google XSS in spreadsheets.google.com allowed session hijacking in all domains.

**HOW TO PREVENT IT?**
- Escape all untrusted data
- **Positive input validation** using API’s like OWASP ESAPI.
- HTTPOnly Cookie Flag
A3. Broken Authentication & Session Management

DEFINITION
Leaks or flaws in the authentication or session management functions (e.g., exposed accounts, passwords, session IDs) due to customized functions.

EXAMPLE
Session Fixation found in Mandiri (biggest Indonesian bank)
1) Attacker crafts email with following link:
   https://ib.bankmandiri.co.id/retail/Login.do?action=form&JSESSIONID=JHAb6Q3Q1BG
   E5uCwNMfTDU1yxfxV9vhMODrP0krLdbem8FvqPA7l!56845468
   - correct domain
   - valid URL
   - using https
2) Victim clicks link validating the SESSIONID
3) Attacker has access to Victim Online Banking.

HOW TO PREVENT IT?
- Meet all the authentication and session management requirements defined in OWASP’s Application Security Verification Standard (ASVS) areas V2 (Authentication) and V3 (Session Management).
- Have a simple interface for developers. Consider the ESAPI Authenticator and User APIs as good examples to emulate, use, or build upon.
Security in the SDLC

SAMM

OWASP

Controls

SDLC

Controls

OWASP

Policy              Awareness          Training

Plan & Design       Build             Test             Implement

architectural risk analysis          Security code review          security testing          WAF/XML firewalls

ASVS                  T10               Webscarab        WebGoat          Swingset

Code Crawler         ZAP               ESAPI            W3AF             ESAPI        WAF

Code Reviewing       Testing           ESAPI            WAF
ASVS can be used to establish a **level of confidence** in the security of Web applications.

- Authentication
- Session Management
- Access Control
- Input Validation
- Output Encoding
- Cryptography
- Error Handling & Logging
- Data Protection
- HTTP Security
Secure Design – App Development Standards

Standards built based on ASVS

- Security Architecture Documentation
- Authentication
- Session Management
- Access Control
- Input/Output validation
- Cryptography
- Error Handling & Logging
- Data Protection
- HTTP Security
- Security Configuration
They should be developed with the following objectives in mind:

- **Use as a metric** – Provide application developers and application owners with a yardstick with which to assess the degree of trust that can be placed in their Web applications.

- **Use as guidance** – Provide guidance to security control developers as to what to build into security controls in order to satisfy application security requirements.

- **Use during procurement** – Provide a basis for specifying application security verification requirements in contracts.
What is the main problem with majority security controls/frameworks?
Secure Coding – The Problem

Frameworks NOT Intuitive, Integrated nor Dev Friendly.
OWASP Enterprise Security API Toolkits helps software developers guard against security-related design and implementation flaws.

Collection of classes that encapsulate the key security operations most applications need.

There are Java EE, .Net, Javascript, Classic ASP ColdFusion/CFML, PHP and Python language versions.

ESAPI for JAVA version includes a Web Application Firewall (WAF).

All language versions of ESAPI are licensed under the BSD license.

You can use or modify ESAPI however you want, even include it in commercial products.

BSD License: Free to Share – copy, distribute and transit the work.

Free to Remix – to adapt the work.
OWASP ESAPI (Enterprise Security API) aims to provide developers with all the security controls they need:

- Standardized
- Centralized
- Organized
- Integrated
- Tested
Security Areas covered by ESAPI

There are more than 120 methods organized in different interfaces
Mapping ESAPI to ASVS (AppDev Stds)

Define your own application risk levels mapped to ASVS for security requirements definition

Here is where you plan how you are going to meet all your selected ASVS security requirements

Build your ESAPI by extending ESAPI controls integrating your standard controls and implementing needed custom controls. Use it to protect your app.

Perform Initial Verification

Here is where you find out if your application has vulnerabilities such as Cross-Site Scripting (XSS), SQL injection, CSRF, etc.

Remediate and Reverify

Fix vulnerabilities

Requirements Definition by Risk Level

App A: Design for a Particular Risk Level

Implementation

Verify against your selected ASVS level

Use ESAPI as part of your Design to meet the ASVS req's

Iterate App Enhancements
Mapping ASVS to ESAPI – An example

- ASVS Session Management

- ESAPI Implementation
  - ESAPI.httpUtilities().changeSessionIdentifier() changes the session id in the login process
  - BTW: prevents session fixation.
Secure Coding - Enterprise Security API

- Web App that demonstrates the features of the ESAPI library.
- Aligned with Application Development Standards.
- Aimed to train developers on ESAPI
  - Each lab presents a vulnerability
  - Developer needs to fix it using ESAPI
  - Labs organized around ASVS
OpenSAMM & SDLC

The resources provided by SAMM will aid in:

**Evaluating an organization’s existing software security practices**

**Building a balanced software security assurance program in well-defined iterations**

**Demonstrating concrete improvements to a security assurance program**

**Defining and measuring security-related activities throughout an organization**
Drivers for a Maturity Model

• An organization behavior changes slowly over time
  – Changes must be iterative while working toward long-term goals
• There is no single recipe that works for all organizations
  – A solution must enable risk-based choices tailor to the organization
• Guidance related to security activities must be prescriptive
  – A solution must provide enough details for non-security-people
• Must be **SIMPLE, WELL DEFINED, and MEASURABLE**
A Maturity Model must:

- **Define** building blocks for an assurance program
  - Delineate all functions within an organization that could be improved
- **Establish** relationships between building blocks
  - Make creating change in iterations an easy task
- **Detail** each building block clearly
  - Clarify the security-relevant parts in a widely applicable way
SAMM Business Functions

- Start with the core activities tied to any organization performing software development
- Named generically, but should resonate with any developer or manager
SAMM Security Practices

• For each of the Business Functions, 3 Security Practices are defined
• The Security Practices cover all areas relevant to software security assurance
• Each one is a ‘siloh’ for improvement
Under each Security Practice

- Each Practice has specific Objectives defining how it can be improved
  - This establishes a notion of a Level at which an organization fulfills a given Practice
- The three Levels for a Practice generally correspond to:
  - (0: Implicit starting point with the Practice unfulfilled)
  - 1: Initial understanding and ad hoc provision of the Practice
  - 2: Increase efficiency and/or effectiveness of the Practice
  - 3: Comprehensive mastery of the Practice at scale
## Example: Education & Guidance Security Practice

### Education & Guidance

<table>
<thead>
<tr>
<th><strong>Objective</strong></th>
<th><strong>Activities</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EG 1</strong></td>
<td>Offer development staff access to resources around the topics of secure programming and deployment</td>
</tr>
<tr>
<td><strong>EG 2</strong></td>
<td>Educate all personnel in the software life-cycle with role-specific guidance on secure development</td>
</tr>
<tr>
<td><strong>EG 3</strong></td>
<td>Mandate comprehensive security training and certify personnel for baseline knowledge</td>
</tr>
<tr>
<td>A. Conduct technical security awareness training</td>
<td>A. Conduct role-specific application security training</td>
</tr>
<tr>
<td>B. Build and maintain technical guidelines</td>
<td>B. Utilize security coaches to enhance project teams</td>
</tr>
<tr>
<td></td>
<td>A. Create formal application security support portal</td>
</tr>
<tr>
<td></td>
<td>B. Establish role-based examination/certification</td>
</tr>
</tbody>
</table>

...more on page 42
OpenSAMM & SDLC

For Each Level

- Objective
- Activities
- Results
- Success Metrics
- Costs
- Personnel
- Related Levels

Education & Guidance

Offer development staff access to resources around the topics of secure programming and deployment

ACTIVITIES

A. Conduct technical security awareness training

Either internally or externally sourced, conduct security training for technical staff that covers the basic tenets of application security. Generally, this can be accomplished via instructor-led training in 1-2 days or via computer-based training with modules taking about the same amount of time per developer. Course content should cover both conceptual and technical information. Appropriate topics include high-level best practices surrounding input validation, output encoding, error handling, logging, authentication, authorization. Additional coverage of commonplace software vulnerabilities is also desirable such as a Top 10 list appropriate to the software being developed (web applications, embedded devices, client-server applications, back-end transaction systems, etc.). Wherever possible, use code samples and lab exercises in the specific programming language(s) that applies.

To rollout such training, it is recommended to mandate annual security training and then hold courses (either instructor-led or computer-based) as often as required based on development head-count.

B. Build and maintain technical guidelines

For development staff, assemble a list of approved documents, web pages, and technical notes that provide technology-specific security advice. These references can be assembled from many publicly available resources on the Internet. In cases where very specialized or proprietary technologies permeate the development environment, utilize senior, security-savvy staff to build security notes over time to create such a knowledge base in an ad hoc fashion. Ensure management is aware of the resources and briefs oncoming staff about their expected usage. Try to keep the guidelines lightweight and up-to-date to avoid clutter and irrelevance. Once a comfort-level has been established, they can be used as a qualitative checklist to ensure that the guidelines have been read, understood, and followed in the development process.

RESULTS

- Increased developer awareness on the most common problems at the code level
- Maintain software with rudimentary security best-practices in place
- Set baseline for security knowledge among technical staff
- Enable qualitative security checks for baseline security knowledge

SUCCESS METRICS

- >50% development staff briefed on security issues within past 1 year
- >75% senior development/architect staff briefed on security issues within past 1 year
- Launch technical guidance within 3 months of first training

COSTS

- Training course buildout or license
- Ongoing maintenance of technical guidance

PERSONNEL

- Developers (1-2 days/yr)
- Architects (1-2 days/yr)

RELATED LEVELS

- Policy & Compliance - 2
- Security Requirements - 1
- Secure Architecture - 1
Approach to iterative improvement

• Since the twelve Practices are each a maturity area, the successive Objectives represent the “building blocks” for any assurance program

• Simply put, improve an assurance program in phases by:
  1. Select security Practices to improve in next phase of assurance program
  2. Achieve the next Objective in each Practice by performing the corresponding Activities at the specified Success Metrics
OpenSAMM & SDLC

Conducting assessments

- SAMM includes assessment worksheets for each Security Practice

<table>
<thead>
<tr>
<th>Education &amp; Guidance</th>
<th>YES/NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>✦ Have most developers been given high-level security awareness training?</td>
<td></td>
</tr>
<tr>
<td>✦ Does each project team have access to secure development best practices and guidance?</td>
<td></td>
</tr>
<tr>
<td>✦ Are most roles in the development process given role-specific training and guidance?</td>
<td></td>
</tr>
<tr>
<td>✦ Are most stakeholders able to pull in security coaches for use on projects?</td>
<td></td>
</tr>
<tr>
<td>✦ Is security-related guidance centrally controlled and consistently distributed throughout the organization?</td>
<td></td>
</tr>
<tr>
<td>✦ Are most people tested to ensure a baseline skill-set for secure development practices?</td>
<td></td>
</tr>
</tbody>
</table>

EG 1

EG 2

EG 3
Creating Scorecards

- Gap analysis
  - Capturing scores from detailed assessments versus expected performance levels
- Demonstrating improvement
  - Capturing scores from before and after an iteration of assurance program build-out
- Ongoing measurement
  - Capturing scores over consistent time frames for an assurance program that is already in place
A phased approach

Phase 1
- Raise awareness about application security

Phase 2
- Improve the security of web applications

Phase 3
- Implement metrics for web application security
A phased approach – Phase 1

Raising awareness about web application security

- Hands-on training about Introduction to Web Application Security.
- Development of an Application Security Group wiki page.
A phased approach – Phase 2

Improving the security of web applications

- Hands-on Training for Developers about OWASP ESAPI & Swingset
- Perform cross security testing of web apps across teams.
- Develop Application Development Standards (ASVS)
Implementing metrics for web application security

- Implement a Security Maturity Model (OpenSAMM)
- Perform regular audits of critical Web Apps.

A phased approach – Phase 3

- Security Architecture Documentation
- Authentication
- Session Management
- Access Control
- Input/Output validation
- Cryptography
- Error Handling & Logging
- Data Protection
- HTTP Security
- Security Configuration
### Example - Small Project Costs to Handle XSS

<table>
<thead>
<tr>
<th>Cost Area</th>
<th>Typical</th>
<th>With Standard XSS Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSS Training</td>
<td>1 days</td>
<td>2 hours</td>
</tr>
<tr>
<td>XSS Requirements</td>
<td>2 days</td>
<td>1 hour</td>
</tr>
<tr>
<td>XSS Design (Threat Model, Arch Review)</td>
<td>2.5 days</td>
<td>1 hour</td>
</tr>
<tr>
<td>XSS Implementation (Build and Use Controls)</td>
<td>7 days</td>
<td>16 hours</td>
</tr>
<tr>
<td>XSS Verification (Scan, Code Review, Pen Test)</td>
<td>3 days</td>
<td>12 hours</td>
</tr>
<tr>
<td>XSS Remediation</td>
<td>3 days</td>
<td>4.5 hours</td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td><strong>18.5 days</strong></td>
<td><strong>4.5 days</strong></td>
</tr>
</tbody>
</table>

Source: Aspect Security Consulting
Cost of doing nothing?

Average annualized cyber crime cost weighted by attack frequency

The most expensive cyber crimes are web-based, malicious code and malicious insider attacks, activities which account for 90 percent of all cyber crime costs per organization on an annual basis.

“First Annual Cost of Cyber Crime Study” by Ponemon Institute
The elapsed time to resolve a cyber attack is very likely to be associated, or correlated, with total costs.
“First Annual Cost of Cyber Crime Study” by Ponemon Institute
Recommendations

1. Define Scope (start small)
2. Get Buy-in from Management
3. Approach Specific Development Team
4. Train them about Application Security
5. Perform Application Security Review
6. Measure and Document Results
7. Start again (extend scope)
Conclusion

Why an Application Security Program?

- Reduces Cost of Development, Recovery and Fixes.
- Reduces Cost of Outsourced Security Testing.
- OWASP provides free tools & resources.
- Implementation costs are minimal.
- Phased approach is proving effective.
Q&A

Want to stay in touch or provide feedback?

FCERULLO@CYCUBIX.COM

Thank you!