Microsoft SDL: Agile Development

Nick Coblentz, CISSP
Senior Security Consultant
AT&T Consulting
Nick.Coblentz@gmail.com
http://nickcoblentz.blogspot.com
http://www.twitter.com/sekhmetn

June 24, 2010

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

The OWASP Foundation
http://www.owasp.org
Bio

■ AT&T Consulting:
  ▸ Application Security
    ▪ Penetration testing
    ▪ Code review
    ▪ Architecture and design reviews
    ▪ Application security program development
    ▪ Secure development methodology improvement

■ Research
  ▸ ISSA Journal: Web Application Security Portfolios
  ▸ SAMM Interview Template
  ▸ Reducing Info Disclosure in ASP.NET Web Services and WCF Data Services
  ▸ Turn Application Assessment Reports into Training Classes
  ▸ Observed Secure Software Development Stages
  ▸ Vulnerability Tracking, Workflow, and Metrics with Redmine
  ▸ Using Microsoft's AntiXSS Library 3.1
Fire Starter: Agile Development and Security

I am a big fan of the Agile project development methodology, especially Agile with Scrum. I love the granularity and focus the approach requires. I love that at any given point in time you are working on the most important feature or function. I love the derivative value of communication and subtle form of peer pressure that Scrum

“...Agile hurts secure code development.”

But it comes with one huge caveat: Agile hurts secure code development. There, I said it. Someone had to. The Agile process, and even the scrum leadership model, hamstrings development in the area of building secure products. Security is not a freakin’ task card. Logic flaws are not well documented, discreet tasks to be assigned. Project managers (and unfortunately most ScrumMasters) learned security by skimming a 'For Dummies' book at Barnes & Noble while waiting for their lattes, but these are the folks making the choices as to what security should make it into the iterations. Just like general IT security, we end up wrapping the Agile process in a security blanket or bolting on security after the code is complete, because the process as we know it is not well suited to secure development.

Adrian Lane:
http://securosis.com/blog/agile-development-and-security/
Microsoft SDL For Agile Released

Announcing SDL for Agile Development Methodologies

Hi everyone, Bryan here. There is a common misconception that because the SDL was originally created for Microsoft's big showcase box products like Windows and SQL Server, that it only works for those kinds of products. This is of course patently false: virtually every Microsoft product and online service, large or small, follows the SDL. Many other organizations outside of Microsoft are also successfully implementing the SDL. However, while the content of the SDL – its requirements and recommendations – may be universal, the structure of the SDL as originally designed is more suited to long-running waterfall- or spiral-style development methodologies. Consider the classic "chevron" SDL graphic:

Microsoft SDL

- **Training**
  - Core training

- **Requirements**
  - Define quality gates/bug bar
  - Analyze security and privacy risk

- **Design**
  - Attack surface analysis
  - Threat modeling

- **Implementation**
  - Specify tools
  - Enforce banned functions
  - Static analysis

- **Verification**
  - Dynamic/Fuzz testing
  - Verify threat models/attack surface

- **Release**
  - Response plan
  - Final security review
  - Release archive

- **Response**
  - Response execution
Microsoft Security Development Lifecycle (SDL)

Components:
- Best Practices
- Processes
- Standards
- Security Activities
- Tools

Goal:
“minimize security-related vulnerabilities in the design, code, and documentation and to detect and eliminate vulnerabilities as early as possible in the development life cycle.”
Which Software?

SDL applies to software that:

- Is used in Business environments
- Stores or transmits PII
- Communicates over the Internet or other networks

Source: Microsoft’s Product Website
SDL Principles and Process

**SD3+C**
- Secure by Design
- Secure by Default
- Secure in Deployment
- Communications

**PD3+C**
- Privacy by Design
- Privacy by Default
- Privacy in Deployment
- Communications

---

**Training**
- Core training

**Requirements**
- Define quality gates/bug bar
- Analyze security and privacy risk

**Design**
- Attack surface analysis
- Threat modeling

**Implementation**
- Specify tools
- Enforce banned functions
- Static analysis

**Verification**
- Dynamic/Fuzz testing
- Verify threat models/attack surface

**Release**
- Response plan
- Final security review
- Release archive

**Response**
- Response execution
What is Agile Development?

Product Backlog → Sprint Backlog → 24 hours → 2-4 weeks → Potentially Shippable Product Increment

Source: http://www.scrumalliance.org/pages/what_is_scrum
## SDLC (Waterfall Methodology)

<table>
<thead>
<tr>
<th>January</th>
<th>February</th>
<th>March</th>
<th>April</th>
<th>May</th>
<th>June</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>Requirements</td>
<td>Design</td>
<td>Implementation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>July</th>
<th>August</th>
<th>September</th>
<th>October</th>
<th>November</th>
<th>December</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation (cont.)</td>
<td>Verification</td>
<td>Release</td>
<td>Response</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This Is NOT Agile Development

January
Agile Development

January

Sprint 1
- User Story 1
- User Story 2
- User Story 3
- User Story 4
- User Story 5
- User Story 5 (Cont.)

Sprint 2
- User Story 6
- User Story 7
- User Story 5 (Cont.)
- User Story 1
- User Story 2
- User Story 3
- User Story 4
- User Story 5
- User Story 5 (Cont.)
- User Story 6
- User Story 7
- User Story 8
Agile Development

- Cross-functional, self-organizing teams
- Short, time-boxed development iterations
- Delivery of small functional stories
- No *extensive* up front design or documentation

Source: http://www.scrumalliance.org/pages/what_is_scrum
Planning and Design

http://www.flickr.com/photos/acarlos1000
Planning and Design (cont.)
User Stories and Documentation

http://www.flickr.com/photos/fmcamargo
SDL SECURITY ACTIVITIES

Source: Simplified Implementation of the Microsoft SDL OWASP
SDL Security Activities

■ Training

■ Requirements
  ▸ Security Requirements
  ▸ Quality Gates/Bug Bars
  ▸ Security and Privacy Risk Assessment

■ Design
  ▸ Design Requirements
  ▸ Attack Surface Reduction
  ▸ Threat Modeling

■ Implementation
  ▸ Use Approved Tools
  ▸ Deprecate Unsafe Functions
  ▸ Static Analysis

■ Verification
  ▸ Dynamic Program Analysis
  ▸ Fuzz Testing
  ▸ Threat Model and Attack Surface Review

■ Release
  ▸ Incident Response Plan
  ▸ Final Security Review
  ▸ Release/Archive

■ Optional Activities
  ▸ Manual Code Review
  ▸ Penetration Testing
  ▸ Vulnerability Analysis of Similar Applications
Traditional SDL Pain Points for Agile

- Can’t complete all SDL activities in each sprint
- Requirements, architecture, and design evolves over time
- Threat model/documentation becomes dated quickly
- Data sensitivity, protection, and connections to third parties may not be immediately known
- Teams don’t include application security specialists
Microsoft SDL For Agile Development

SDL Requirement Categories:
- Every-Sprint
- Bucket
  - Verification Tasks
  - Design Review Tasks
  - Response Planning Tasks
- One-Time

Source: Microsoft SDL v4.1a
Every-Sprint SDL Requirements

“...so essential to security that no software should ever be released without these requirements being met.”

Examples:

- Update the threat model
- Communicate privacy-impacting design changes to the team’s privacy advisor
- Fix all issues identified by code analysis tools for unmanaged code
- Follow input validation and output encoding guidelines to defend against cross-site scripting attacks
Bucket SDL Requirements

- Teams prioritize the pool of tasks over many sprints
- Each sprint, one task from each bucket completed
- Each task must be completed at least every 6 months

Examples:

- Security Verification Tasks
  - Run fuzzing tools
  - Manual and automated code review

- Design Review Tasks
  - Conduct privacy review
  - In-depth threat model

- Response Planning Tasks
  - Define security/privacy bug bar
  - Create support documents
One-Time Requirements

Why?
- Repetition not necessary
- Must occur at the beginning of the project
- Not possible at the beginning of the project

Examples:
- Configure bug tracking system (3 months)
- Identify security/privacy experts (1 month)
- Baseline threat model (3 months)
- Establish a security response plan (6 months)
# SDL-Agile Appendix

## Appendix Q: SDL-Agile Bucket Requirements

### Bucket A: Security Verification

<table>
<thead>
<tr>
<th>Title</th>
<th>Requirement/Recommendation</th>
<th>Applies to Online Services</th>
<th>Applies to Managed Code</th>
<th>Applies to Native Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debug the application with the Application Verifier enabled</td>
<td>Requirement</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Disable tracing and debugging in ASP.NET applications</td>
<td>Requirement</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Investigate and service any reported IIS crashes</td>
<td>Requirement</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Perform ActiveX control fuzzing</td>
<td>Requirement</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Perform attack surface analysis</td>
<td>Requirement</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Perform binary analysis (BinScope)</td>
<td>Requirement</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Perform COM object testing</td>
<td>Requirement</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Perform cross-domain scripting testing</td>
<td>Requirement</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Perform file fuzz testing</td>
<td>Requirement</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perform RPC fuzz testing</td>
<td>Requirement</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Conduct in-depth manual and automated code review for high-risk code</td>
<td>Recommendation</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Perform data flow testing</td>
<td>Recommendation</td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
## SDL-Agile Appendix: Deadlines

### Appendix R: SDL-Agile One-Time Requirements

<table>
<thead>
<tr>
<th>Title</th>
<th>Requirement/Recommendation</th>
<th>Completion Deadline (months)</th>
<th>Applies to Online Services</th>
<th>Applies to Managed Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avoid writable PE segments</td>
<td>Requirement</td>
<td>6</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Configure bug tracking to track the cause and effect of security vulnerabilities</td>
<td>Requirement</td>
<td>3</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Create a baseline threat model</td>
<td>Requirement</td>
<td>3</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Determine security response standards</td>
<td>Requirement</td>
<td>6</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Establish a security response plan</td>
<td>Requirement</td>
<td>6</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Identify primary</td>
<td>Requirement</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Final Security Review

- Occurs at the end of every sprint
- Checklist:
  - All every-sprint requirements have been completed
  - No one-time requirements have exceeded deadline
  - At least one requirement from each bucket category has been completed
  - No bucket requirements exceed the six month deadline
  - No security or privacy bugs are open that exceed the severity threshold
Making SDL-Agile Manageable

- Documented standards
- Security training
- Automation
  - Continuous Integration
    - Secure Configuration
    - Security Unit Tests
    - Automated Secure Code Analysis
    - Automated Deployment and Vulnerability Scanning
- Process
  - Continuous updates to the threat model
  - SDL Process Templates for VSTS
  - MSF-Agile + SDL Process Template
- Light on security artifacts/documentiation
Making SDL-Agile Manageable

- Tooling
  - Code Analysis/Scanning
    - CAT.NET
    - MiniFuzz
    - BinScope Binary Analyzer
    - Fiddler w/ Watcher
    - FxCop
  - MS Threat Modeling Tool
CAT.NET: Cross-site Scripting Vulnerability

Analysis Information
- Analysis Engine Version: 1.0.3455.36250
- Created by: [Redacted]
- Start time: Sunday, February 28, 2010 1:34:46 PM
- Stop time: Sunday, February 28, 2010 1:34:47 PM
- Elapsed time: 00:00:00.610000
- Data flow graph: 5 nodes, 5 edges
- Targets: C:\Users\[Redacted]\Desktop\UnsignedUnecryptedViewStateExploit\UnsignedUnecryptedViewStateExploit

Cross-Site Scripting (ACESEC05)
- 1 results

Result #1

Summary
A cross-site scripting vulnerability was found through a user controlled variable that enters the application at Default.aspx.cs:21 through the variable stack1 which eventually leads to a cross-site scripting issue at Default.aspx.cs:21.

Resolution
Use the Anti-XSS library to properly encode the data before rendering it

Entry Variable
stack1

Confidence
High

Source Context
<table>
<thead>
<tr>
<th>Line</th>
<th>Input Variable</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default.aspx.cs 21</td>
<td>lbIPayload.Text = txtBox1.Text;</td>
<td></td>
</tr>
<tr>
<td>Default.aspx.cs 21</td>
<td>Return from TextBox.get_Text</td>
<td>lbIPayload.Text = txtBox1.Text;</td>
</tr>
</tbody>
</table>
Making SDL-Agile Manageable

- Libraries
  - Web Protection Library (WPL)
    - Encoder/Anti-XSS Library
    - Security Runtime Engine (SRE)
    - Sanitizer.GetSafeHTML
# Web Protection Library - Encoder/AntiXSS

## Encoder Methods

The **Encoder** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CssEncode</code></td>
<td>Encodes input strings used in Cascading Style Sheet (CSS) elements.</td>
</tr>
<tr>
<td><code>HtmlAttributeEncode</code></td>
<td>Encodes input strings for use in HTML attributes.</td>
</tr>
<tr>
<td><code>HtmlEncode</code></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><code>JavaScriptEncode</code></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><code>UrlEncode</code></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><code>XmlAttributeEncode</code></td>
<td>Encodes input strings for use in XML attributes.</td>
</tr>
<tr>
<td><code>XmlEncode</code></td>
<td>Encodes input strings for use in XML.</td>
</tr>
</tbody>
</table>
The Security Runtime Engine (SRE)

- “The Security Runtime Engine (SRE) is an HTTP module that acts like a gatekeeper to protect ASP.NET web applications from cross-site scripting (XSS) attacks.”
- “It works by inspecting each control that is being reflected by ASP.NET and then automatically encoding data of vulnerable controls in their appropriate context.”
- SRE Configuration Editor GUI Tool
The Security Runtime Engine (SRE)
The Security Runtime Engine (SRE)
Making SDL-Agile Manageable

- Deployment
  - Web Application Configuration Analyzer (WACA)
  - Microsoft Baseline Security Analyzer
  - Web.config Security Analyzer (WCSA)
Web Application Configuration Analyzer

Quick Actions
- Scan a machine
- View a report
- Set scan credentials
- Map Team Foundation Server fields

About
Web Application Configuration Analyzer (WACA) analyzes application configuration for security best practices related to General Application, IIS, ASP.NET Application and SQL Server settings. Machine can be scanned remotely to identify any misconfigurations. It provides detailed report on multiple instances of checks for further analysis. Violations in the report can be exported to Excel or Visual Studio Team Foundation Server.

Rules
The following tree lists the breakdown of rules that are currently checked by the tool.

- Rules (109)
  - General Application Rules (33)
  - IIS Application Rules (63)
  - SQL Application Rules (13)
Web Application Configuration Analyzer

Microsoft Web Application Configuration Analyzer v1.0 - [New Scan 3]

New Scan

Enter the fully qualified name of the machine to scan. Ex: server.orgunit.organization. Current user or the specified scan credentials should have administrator access to the machine. Click "Scan" to start scanning, progress of the scan will be displayed in the progress bar.

- Machine Name: [input field]
- SQL Instance Name: [input field]
- Rule Category:
  - General Application Rules
  - IIS Application Rules
  - SQL Application Rules

[Progress bar] [Scan] [Cancel]
Web Application Configuration Analyzer
1. Debugging Enabled
You have enabled debugging on your application. This could be used by attackers to extract useful information such as detailed information and stack trace, etc.

Secure Configuration
<configuration>
  <system.web>
    <compilation debug="false"/>
  </system.web>
</configuration>

References

2. Web cookies are not HttpOnly
Your configuration allows cookies to be be accessed from client-side scripting technologies. This can lead an attacker to perform Cross-Site Scripting and session hijacking.

Secure Configuration
<configuration>
  <system.web>
    <httpCookies httpOnlyCookies="true"/>
  </system.web>
</configuration>

References
- http://www.owasp.org/index.php/HTTPOnly

3. Web cookies doesn't require SSL
Your configuration allows web cookies to be transferred in clear text form. An attacker that can able to monitor your network, can easily steal your session.

Secure Configuration
Making SDL-Agile Manageable

- Education, secure coding standards, **automation** and **tools** play a significant role in making secure Agile development efficient and economical

- Don’t forget:
  - Periodic manual security activities are also a must
  - All of this must fit within a **repeatable, mature process**
Summary and Questions

More Information:  
http://www.microsoft.com/sdl

Nick Coblentz, CISSP  
Senior Consultant, AT&T Consulting  
Nick.Coblentz@gmail.com  
http://nickcoblentz.blogspot.com  
http://www.twitter.com/sekhmetn

- Microsoft releases SDL-Agile Guidance in Nov. 2009
- Treats SDL Activities like team-prioritized User Stories
  - 3 Categories: One-time, Every-time, and Bucket
- Increased success with the implementation of training, automation, tools, and standards