Il processo SDL in Microsoft: problematiche, vantaggi e risultati

Feliciano Intini
Chief Security Advisor
Microsoft Italy

http://blogs.technet.com/b/feliciano_intini
http://twitter.com/felicianointini
Software Vulnerability Disclosures

Operating system, Browser and Application Disclosures

Microsoft Security Intelligence Report v9 (1H10): www.microsoft.com/sir
How We Got Here

• Through 1980s, security was about insiders
  – Studies and experiments demonstrated potential for attacks on software
  – No real examples
  – “Nobody would ever...”

• Computer security treated as a theoretical problem
  – Prove it’s secure and you’re done forever
  – Market proved unsympathetic (or absent) – projects canceled, no real products
How We Got Here

• PC and Internet changed the rules
  – Viruses, information sharing, “outside” and “inside” indistinguishable
  – Vulnerability research for reputation

• Vulnerability research led to security response process
  – Fix the problems when they’re found

• “Secure Windows Initiative” to make software secure
  – Assigned three program managers to review Windows
  – Evolved to training and “bug bashes”
As I've talked with customers over the last year - from individual consumers to big enterprise customers - it's clear that everyone recognizes that computers play an increasingly important and useful role in our lives. At the same time, many of the people I talk to are concerned about the security of the technologies they depend on...
How We Got Here: The Security Push Era

• Security push
  – Team-wide stand-downs and training
  – Threat model, review code, run tools, conduct tests, modify defaults
  – (Relatively) quick way to significant improvement
  – Immature and ad hoc processes

• “Security science”
  – Identify and remove new classes of vulnerabilities

• Security “audit”
  – Independent review – what did the push miss?
Selling the Process

• Security pushes were an “obviously” necessary response...

• Security pushes achieved rapid improvements (some dramatic) but...

• Leverage comes from early (design time) focus on security

• Ongoing attacks demonstrated continued need

• Executive buy-in surprisingly easy in retrospect
  – Everyone understood what bad things could happen
  – Security pushes had accomplished enough to allow us to claim we could do this
The Microsoft SDL

**Goals**

- Protect Microsoft customers by reducing the number of vulnerabilities
- Reducing the severity of vulnerabilities

**Key Principles**

- Prescriptive yet practical approach
- Proactive - not just “looking for bugs”
- Eliminate security problems early
- Secure by design
The Classic SDL at Microsoft

Ongoing Process Improvements

- Education
  - Core training
- Technology and Process
  - Requirements: Analyze security and privacy risk, Define quality gates
  - Design: Threat modeling, Attack surface analysis
  - 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
- Accountability
  - Response

Ongoing Process Improvements
SDL for Agile at Microsoft

- Requirements defined by frequency, not phase
  - Every-Sprint (most critical)
  - One-Time (non-repeating)
  - Bucket (all others)

- Great for projects without end dates, like cloud services
Managing Change

• The first (2004) iteration of the SDL was pretty rough
  – Developed rapidly based on security push lessons

• Initial updates at 6-month intervals
  – Responses to new threats
  – New application classes (privacy, online services)
  – New requirements and techniques (e.g. banned APIs, new fuzzers)

• Since SDL v4 (October 2007), annual updates
  – More time for tool development
  – More time for beta and feedback
  – More time for usability

• Every update receives both broad and senior review
Process Improvement Timeline at Microsoft...

2002-2003
- Bill Gates writes “Trustworthy Computing” memo early 2002
- “Windows security push” for Windows Server 2003
- Security push and FSR extended to other products

2004
- Microsoft Senior Leadership Team agrees to require SDL for all products that:
  - Are exposed to meaningful risk and/or
  - Are Process sensitive data

2005-2007
- SDL is enhanced
  - “Fuzz” testing
  - Code analysis
  - Crypto design requirements
  - Privacy
  - Banned APIs
  - and more...
  - Windows Vista is the first OS to go through full SDL cycle

Now
- Optimize the process through feedback, analysis and automation
- Evangelize the SDL to the software development community:
  - SDL Process Guidance
  - SDL for Agile
  - SDL Optimization Model
  - SDL Pro Network
  - SDL Threat Modeling Tool
  - SDL Process Templates
Automation and Tools

• At Microsoft today, the SDL requires three classes of tools
  – Automated tools to help find (and remove or mitigate) security problems
  – Automated tools to help product teams record and track their compliance with the SDL
  – Automated tools to help the MSEC PM (security advisor) help the product teams

• We started with only the first (problem finders)

• All three are critical to our implementation of the SDL – and we’ve changed our release cadence largely in recognition of this fact
Things we have learned

• “There is nothing special about security”
  – It’s simply part of getting the job done.

• Get to a knowledge baseline
  – You must raise the collective security IQ to a baseline level
  – Don’t try to make everyone a security expert

• You’re in or you’re out
  – Existing software development practices do not foster secure software, you must change your development process

• Executive Support is Key
  – If the execs don’t “get it” you’ll make marginal progress

• Deprecate old Functionality
  – Old functionality was developed in a different era, with different security landscape
  – Unfortunately, your users have become accustomed to the features!
Things we have learned

• Reduce Friction
  – Few software people are true security experts so we must make security as easy as possible for them
  – Automate, use static analysis tools, better libraries, updated C/C++ compilers

• You’ll never reach ‘perfection’
  – As long as attackers and researchers are drawing breath, new bugs will be found
  – The odds are against you

• Today’s DoS is tomorrow’s RCE
  – We have seen time and again what’s generally considered a DoS become a way to execute code: “Attacks only get better”

• You will never get the code right. Ever!
  – The software industry spends an incredible amount of time trying to get the code right
  – The SDL focuses a great deal on defenses, not just getting the code right
Why Defenses are so Important

Security Advisory 979352 – IE 0Day

<table>
<thead>
<tr>
<th>Internet Explorer 6</th>
<th>Windows 2000</th>
<th>Exploitable</th>
<th>Windows XP</th>
<th>Exploitable (current exploit effective for code execution)</th>
<th>Windows Vista</th>
<th>N/A (Vista ships with IE7)</th>
<th>Windows 7</th>
<th>N/A (Windows 7 ships with IE 8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Explorer 7</td>
<td>N/A</td>
<td>N/A (IE 7 will not install on Windows 2000)</td>
<td>Windows XP</td>
<td>Potentially exploitable (current exploit does not currently work due to memory layout differences in IE 7)</td>
<td>Windows Vista</td>
<td>IE Protected Mode prevents current exploit from working.</td>
<td>Windows 7</td>
<td>N/A (Windows 7 ships with IE 8)</td>
</tr>
<tr>
<td>Internet Explorer 8</td>
<td>N/A</td>
<td>N/A (IE 8 will not install on Windows 2000)</td>
<td>Windows XP</td>
<td>DEP enabled by default on XP SP3 prevents exploit from working.</td>
<td>Windows Vista</td>
<td>IE Protected Mode + DEP enabled by default prevent exploit from working.</td>
<td>Windows 7</td>
<td>IE Protected Mode + DEP enabled by default prevent exploit from working.</td>
</tr>
</tbody>
</table>

Malicious And Potentially Unwanted Software

Operating system trends

Number of computers cleaned for every 1,000 MSRT executions, by operating system, 2Q10
The SDL and the CWE/SANS Top 25

- The SDL addresses all CWE/SANS Top 2009 issues
- Through one or more of:
  - Education
  - Manual Process
  - Tools
  - Threat Model
Who Needs the SDL?

Subject: I swear, i'm giving our kids normal names...

Today's XKCD (http://xkcd.com/327/)

HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.

OH, DEAR - DID HE BREAK SOMETHING? IN A WAY -

DID YOU REALLY NAME YOUR SON Robert?); DROP TABLE Students;-- ?

OH, YES. LITTLE BOBBY TABLES, WE CALL HIM.

WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.

AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.
Objections to the SDL

“...only for Windows”
- Based on proven, generally accepted security practices
- Appropriate for non-Microsoft platforms

“...for shrink-wrapped products”
- Also covers Line of Business (LOB) and online services development

“...for waterfall or spiral development”
- Agile methods are also supported

“...requires Microsoft tools”
- Use the appropriate tools for the job

“...requires Microsoft-level resources to implement”
- SDL as its applied at Microsoft != SDL for other development organizations
- Some smaller organizations have adopted
Adapting the SDL to Organizations Beyond Microsoft

- Non-proprietary
- Scalable to organizations of any size
- Platform agnostic
- Based on the SDL process used at Microsoft
Who Uses the SDL?

- Short answer: we don’t know
- You have to click through a EULA to download the tools, but you don’t have to register so...
- We have worked with some large organizations on adopting and adapting the SDL (mostly not public)
- We’ve seen the Errata survey, and had some users (large and small) tell us they’re using the SDL
- Finding the answer is one of our objectives for the next year
Resources at a glance...

- SDL Whitepaper
- SDL Book
- Privacy Guidelines
- SDL Optimization Model
- SDL Threat Modeling Tool
- SDL 3.2 Guidance
- SDL 4.1 Guidance
- SDL Process Template
- SDL-Agile Template
- Binscope Binary Analyzer
- MiniFuzz Fuzzer

SDL becomes mandatory @ MSFT
Transforms threat modeling from an expert-led process into a process that any software architect can perform effectively.

Provides:
- Guidance in drawing threat diagrams
- Guided analysis of threats and mitigations
- Integration with bug tracking systems
- Robust reporting capabilities
The SDL Process Template integrates SDL 4.1 directly into the VSTS software development environment.

- **Incorporates**
  - SDL requirements as work items
  - SDL-based check-in policies
  - Generates Final Security Review report
  - Third-party security tools
  - Security bugs and custom queries
  - A library of SDL how-to guidance

- **Integrates with previously released free SDL tools**
  - SDL Threat Modeling Tool
  - Binscope Binary Analyzer
  - Minifuzz File Fuzzer
MSF Agile + SDL Template for VSTS

- Incorporates SDL-Agile secure development practices directly into the Visual Studio IDE - now available as beta (planned release at the end of Q2CY10)

- Automatically creates new security workflow items for SDL requirements whenever users check in code or create new sprints

- Ensures important security processes are not accidentally skipped or forgotten

- Integrates with previously released free SDL tools
  - SDL Threat Modeling Tool
  - Binscope Binary Analyzer
  - Minifuzz File Fuzzer

- Will be updated for VS2010
Binscope Binary Analyzer

- Provides an extensive analysis of an application binary

- Checks done by Binscope
  - /GS - to prevent buffer overflows
  - /SafeSEH - to ensure safe exception handling
  - /NXCOMPAT - to prevent data execution
  - /DYNAMICBASE - to enable ASLR
  - Strong-Named Assemblies - to ensure unique key pairs and strong integrity checks
  - Known good ATL headers are being used

- Use either standalone or integrated with Visual Studio (VS) and Team Foundation Server (TFS)
MiniFuzz is a basic testing tool designed to help detect code flaws that may expose security vulnerabilities in file-handling code.

- Creates corrupted variations of valid input files
- Exercises the code in an attempt to expose unexpected application behaviors.
- Lightweight, for beginner or advanced security testing
- Use either standalone or integrated with Visual Studio (VS) and Team Foundation Server (TFS)
SDL and Creative Commons

• Up to this point, Microsoft has released SDL information using a license that did not allow for reproduction, inclusion or transfer of any part of our documentation or process without express written consent from Microsoft.

• From this point forward, Microsoft will be making our publicly available SDL documentation and other SDL process content available to the development community under a Creative Commons license.
  – Specifically, we will be using the license that specifies Attribution, Non-Commercial, Share Alike (cc by-nc-sa) terms. (http://creativecommons.org/licenses/by-nc-sa/3.0/)
  – This won’t apply for any of the SDL tools released by Microsoft – those will continue to use existing Microsoft licenses.

• First two papers republished under CC license:
  – “Simplified Implementation of the Microsoft SDL” whitepaper and the Microsoft Security Development Lifecycle (SDL) - Version 5.0
Summary

• You’re here, so you all understand the importance of building secure software

• Integrating security into a development process and organization requires commitment and time

• Our experience has shown that the SDL is an effective process – and that it can be applied beyond Microsoft

• We’ve made a lot of resources freely available to help other organizations apply the SDL
Online Resources

SDL Portal
http://www.microsoft.com/sdl

SDL Blog
http://blogs.msdn.com/sdl/

SDL Process on MSDN (Web)

Simplified Implementation of the Microsoft SDL
http://go.microsoft.com/?linkid=9708425