The Security Development Lifecycle

OWASP

24 June 2010

Steve Lipner
Senior Director of Security Engineering Strategy
Trustworthy Computing
Microsoft Corporation
SLipner@microsoft.com
+1 425 705-5082

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

Overview

- How we got here
- Selling the process
- The SDL at Microsoft
- Managing change
- Automation and tools
- The Simplified SDL: Adapting the SDL to new organizations
- Objections
- Resources
- Questions?

How We Got Here

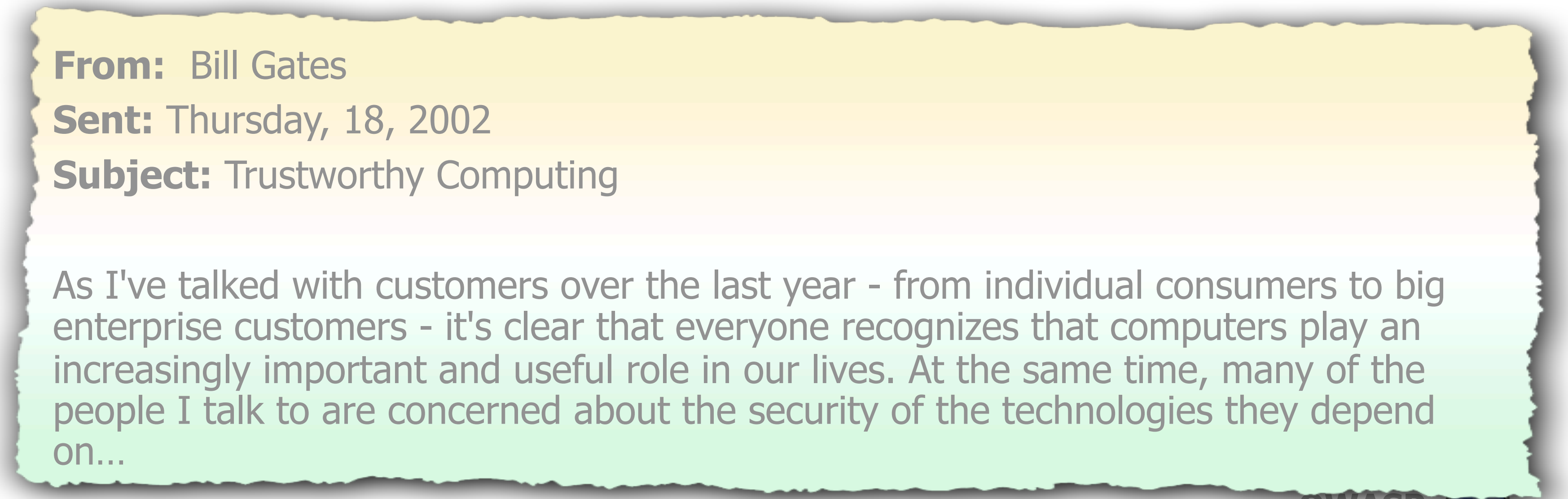
- Through 1980s, security was about insiders
  - Studies and experiments demonstrated potential for attacks on software
  - No real examples
  - “Nobody would ever…”
- Computer security treated as a theoretical problem
  - Prove it’s secure and you’re done forever
  - Market proved unsympathetic (or 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”

How We Got Here

- Thought we'd done "better" with XP, and then…
  - Code Red
  - Nimda
  - UPNP



From: Bill Gates

Sent: Thursday, 18, 2002

Subject: Trustworthy Computing

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…

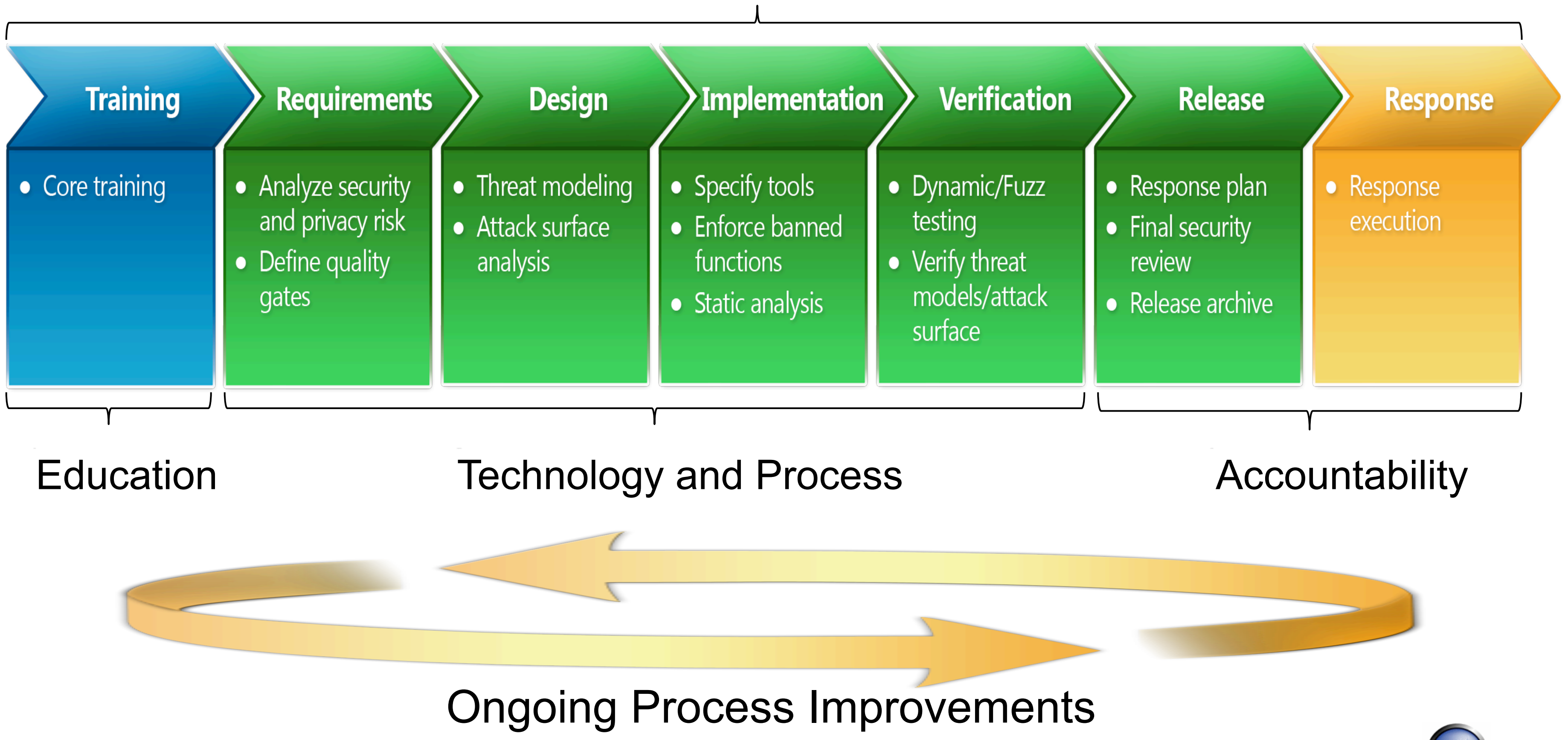
How We Got Here: The Security Push Era

- 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

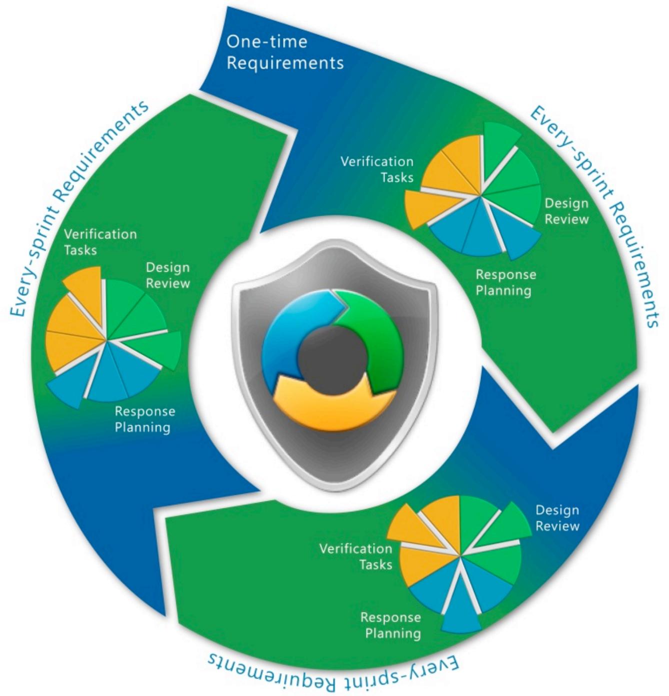
The Classic SDL at Microsoft



OWASP

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



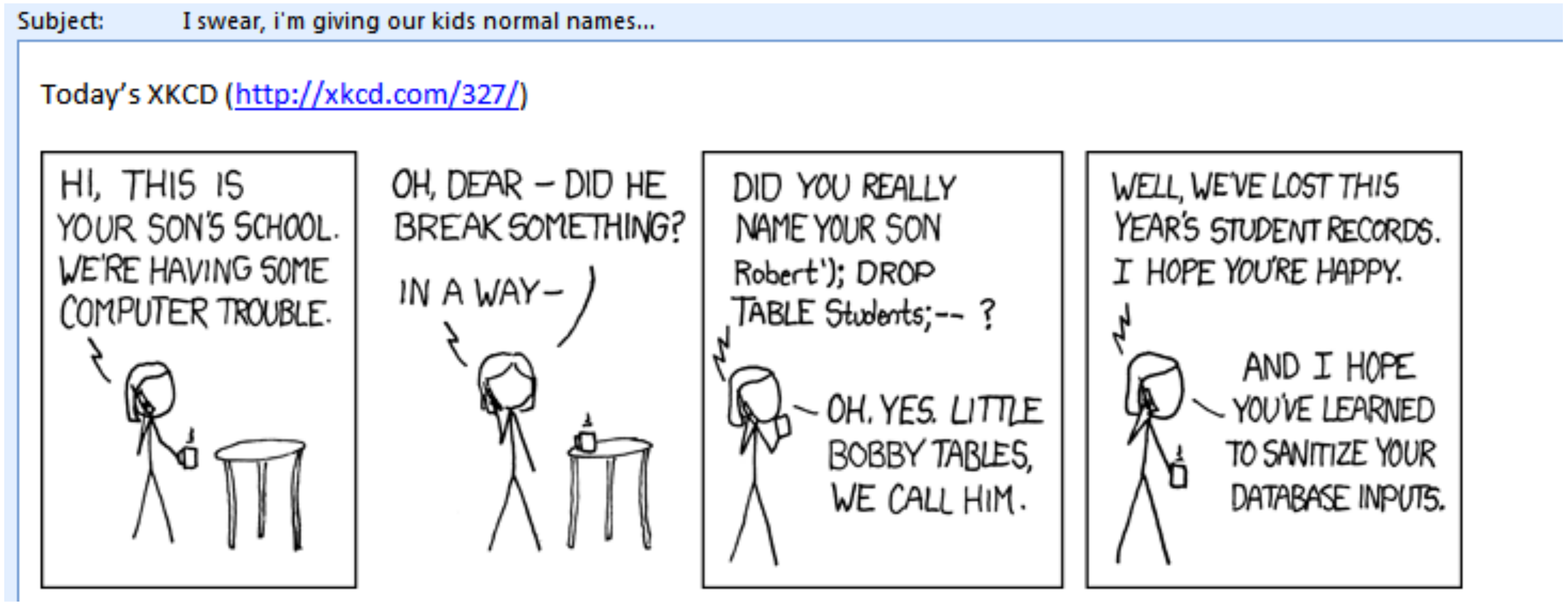
Managing Change

- 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

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

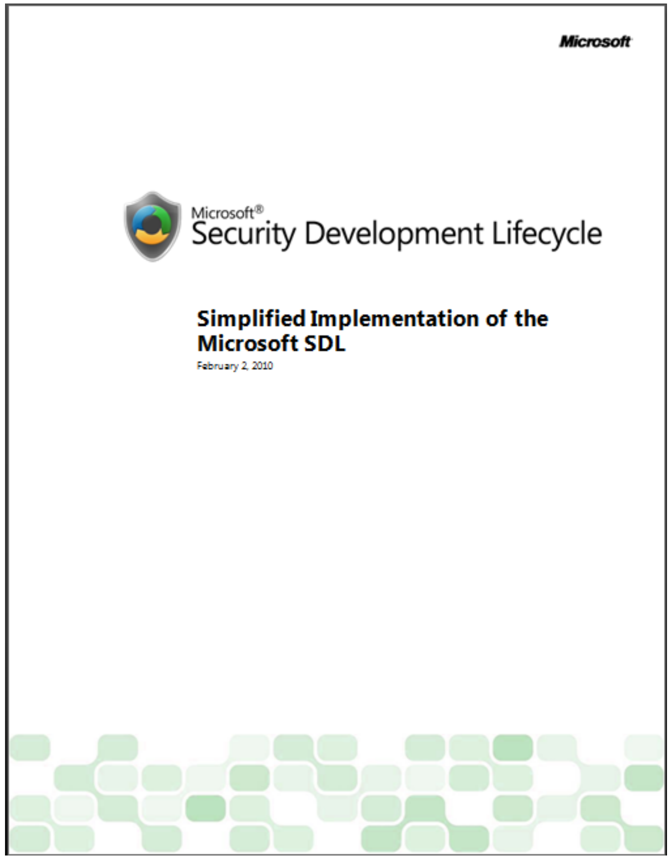
Who Needs the SDL?


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.

Adapting the SDL to Organizations Beyond Microsoft



Microsoft

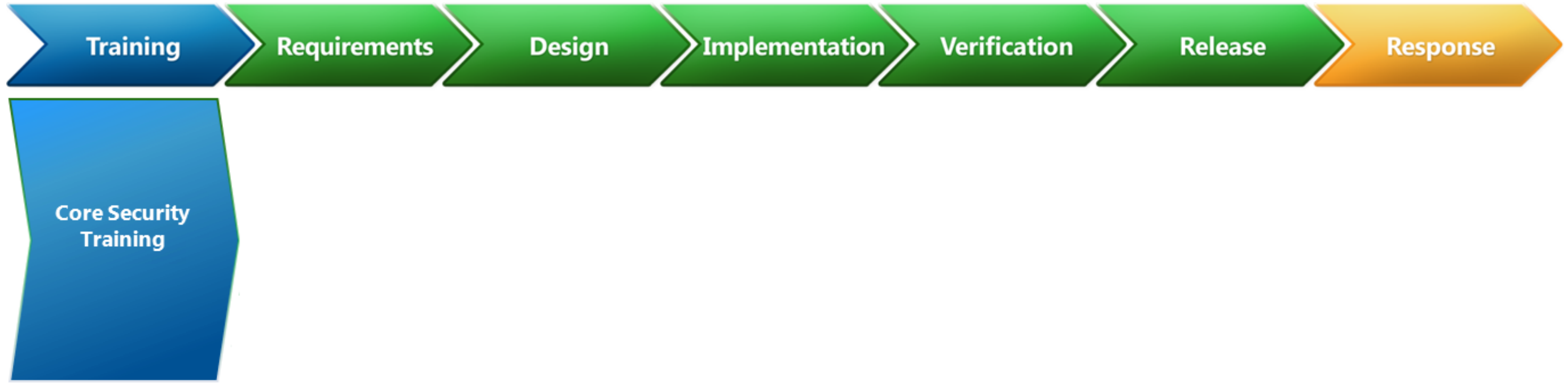
Microsoft® Security Development Lifecycle

Simplified Implementation of the Microsoft SDL

February 2, 2010

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

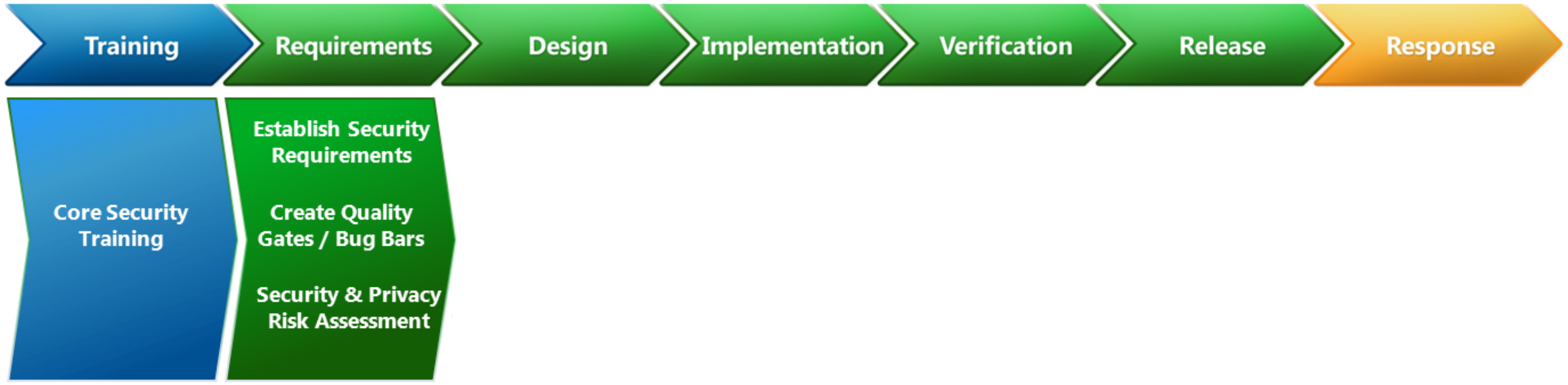
Pre-SDL Requirement: Security Training



Assess organizational knowledge – establish training program as necessary

- Establish training criteria
  - Content covering secure design, development, test and privacy
- Establish minimum training frequency
  - Employees must attend *n* classes per year
- Establish minimum acceptable group training thresholds
  - Organizational training targets (e.g. 80% of all technical personnel trained prior to product RTM)

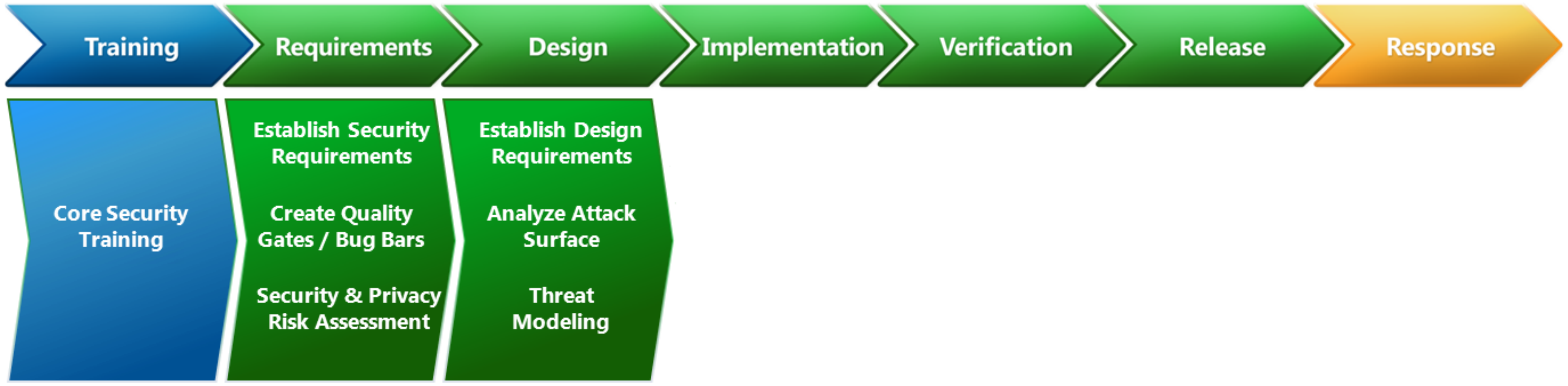
Phase One: Requirements


Training
Requirements
Design
Implementation
Verification
Release
Response
Core Security Training
Establish Security Requirements
Create Quality Gates / Bug Bars
Security & Privacy Risk Assessment

Opportunity to consider security at the outset of a project

- Establish Security Requirements
  - Project wide requirements – security leads identified, security bug tracking process mandated, architectural requirements set given the planned operational environment
- Create Quality Gates / Bug Bars
  - Minimum performance and quality criteria for each stage and for the project as a whole,
- Security and Privacy Risk Assessment
  - Risk assessment performed to determine critical components for the purposes of deep security and privacy review

Phase Two: Design

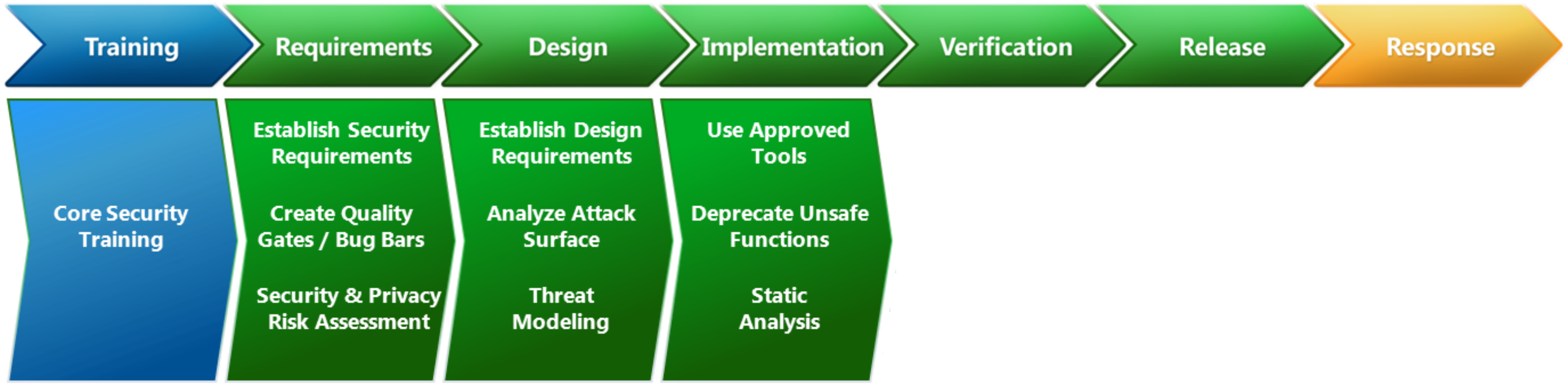


Define and document security architecture, identify security critical components

- Establish Design Requirements
  - Required activities which include creation of design specifications, analysis of proposed security technologies (e.g. crypto requirements) and reconciliation of plans against functional specs.
- Analyze Attack Surface
  - Defense in depth strategies employed – use of layered defenses used to mitigate severity.
- Threat Modeling
  - Structured, component-level analysis of the security implications of a proposed design.

OWASP

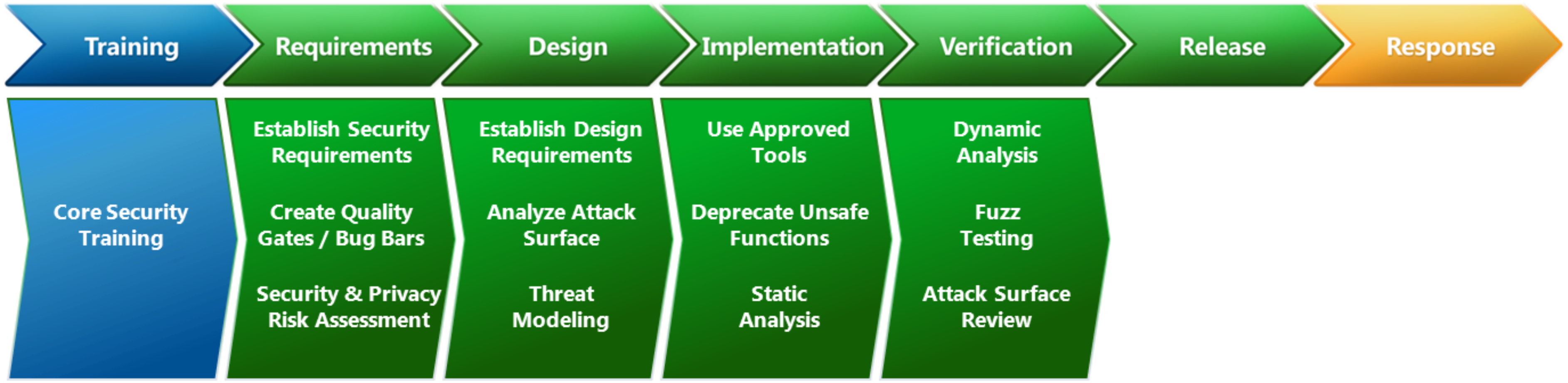
Phase Three: Implementation



Determine processes, documentation and tools necessary to ensure secure development

- Use approved tools
  - Approved list for compilers, security test tools, switches and flags; enforced project wide.
- Deprecate Unsafe Functions
  - Ban unsafe functions, APIs, when using native (C/C++) code.
- Static Code Analysis
  - Scalable in-depth code review, augmentation by other methods as necessary to address weaknesses in static analysis tools.

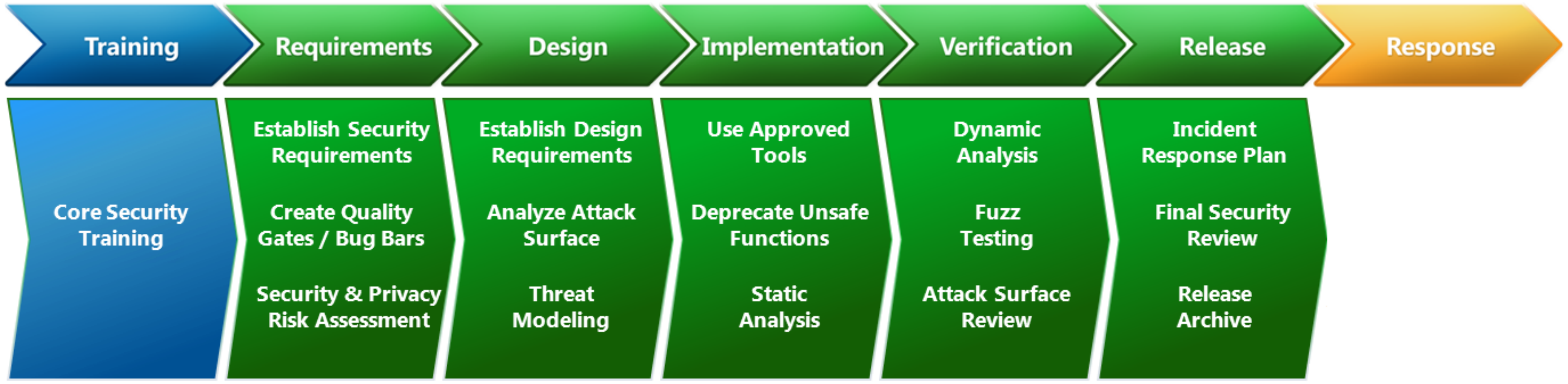
Phase Four: Verification



Verification of SDL security and privacy activities performed earlier in the process

- Dynamic Analysis
  - Runtime verification and analysis of programs to identify critical security problems
- Fuzz Testing
  - Specialized dynamic analysis technique used to deliberately cause program failure by injection of random, deliberately malformed inputs.
- Attack Surface / TM review
  - Re-review of attack surface and threat models when the program is "code complete" to ensure security assumptions and mitigations specified at design time are still relevant.

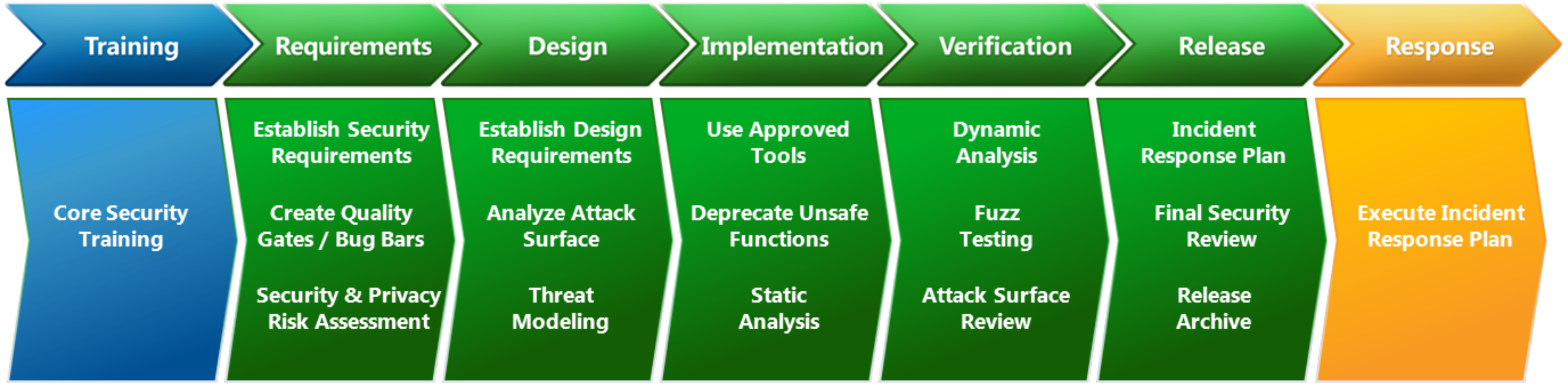
Phase Five: Release



Satisfaction of clearly defined release criteria – consistent with organizational policy

- Incident Response Plan
  - Creation of a plan that outlines engineering, management and "on-call" contacts, security servicing plans for all code, including 3rd party artifacts.
- Final Security Review
  - Deliberate examination of all security and privacy activities conducted during development
- Release Archive
  - SDL compliance certification and archival of all information and data necessary for post-release servicing of the software.

Post-SDL Requirement: Response



"Plan the work, work the plan…"

- Execute Incident Response Plan
  - Performance of activities outlined in response plan created during Release phase
- Other non-development, post-release process requirements
  - Root cause analysis of found vulnerabilities; failure of human, process, or automation. Addressed immediately and tagged for inclusion in next revision of SDL

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*

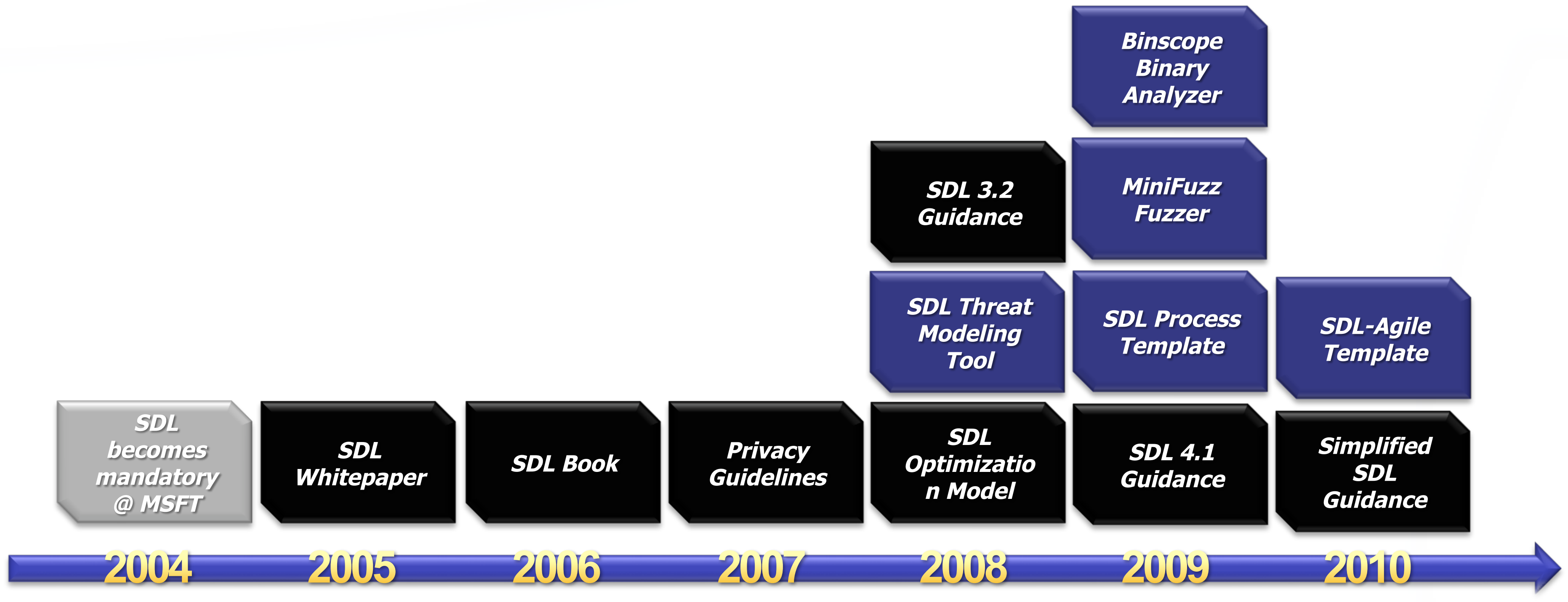
"…requires Microsoft-level resources to implement"

- *SDL as its applied at Microsoft != SDL for other development organizations*
- *Some smaller organizations have adopted*

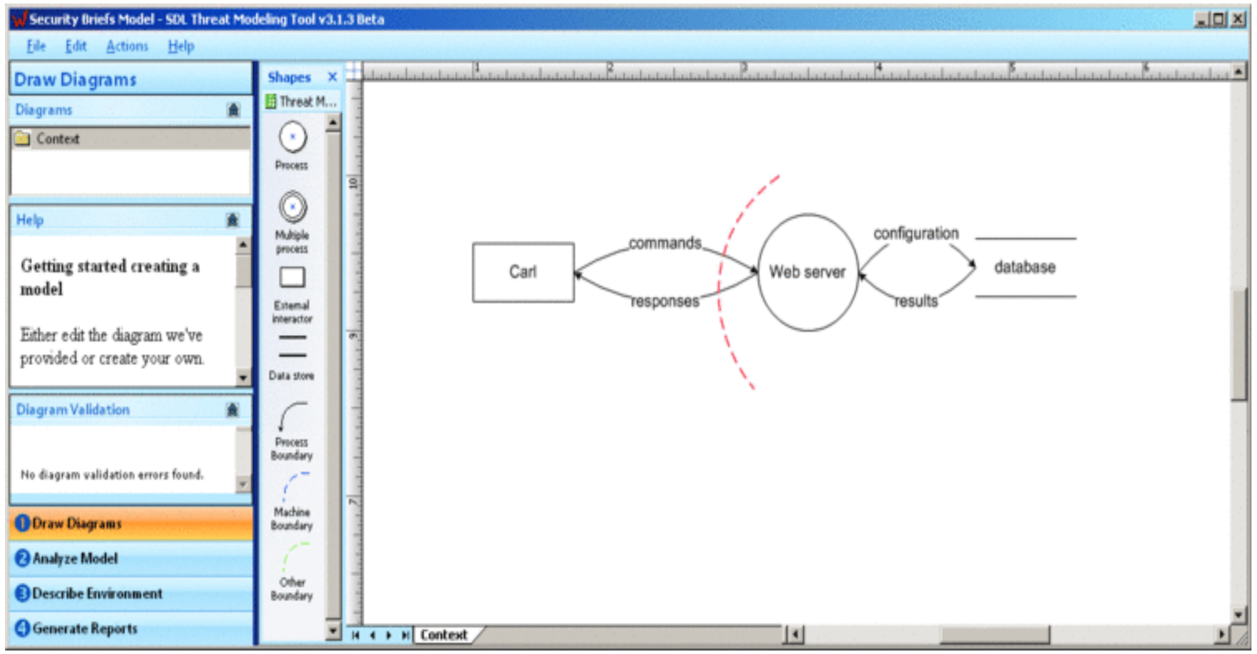
Who Uses the SDL?

- 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 Threat Modeling Tool



Getting started creating a model

Either edit the diagram we've provided or create your own.

No diagram validation errors found.

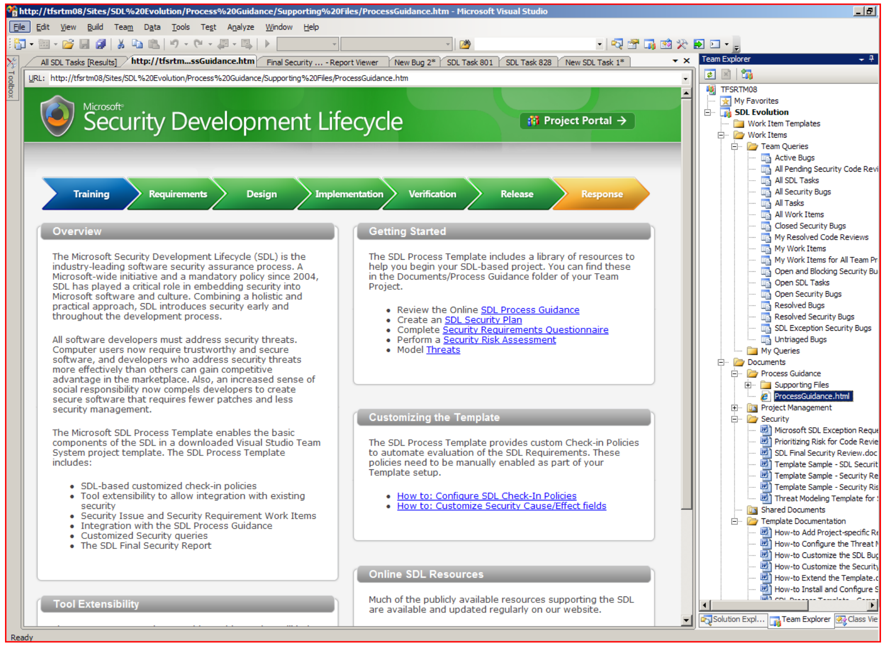
Transforms threat modeling from an expert-led process into a process that any software architect can perform effectively

Provides:

- Guidance in drawing threat diagrams
- Guided analysis of threats and mitigations
- Integration with bug tracking systems
- Robust reporting capabilities

OWASP

SDL Template for VSTS (Spiral)



Overview

The Microsoft Security Development Lifecycle (SDL) is the industry-leading software security assurance process. A Microsoft-wide initiative and a mandatory policy since 2004, SDL has played a critical role in embedding security into Microsoft software and culture. Combining a holistic and practical approach, SDL introduces security early and throughout the development process.

All software developers must address security threats. Computer users now require trustworthy and secure software, and developers who address security threats more effectively than others can gain competitive advantage in the marketplace. Also, an increased sense of social responsibility now compels developers to create secure software that requires fewer patches and less security management.

The Microsoft SDL Process Template enables the basic components of the SDL in a downloaded Visual Studio Team System project template. The SDL Process Template includes:

- SDL-based customized check-in policies
- Tool extensibility to allow integration with existing security
- Security Issue and Security Requirement Work Items
- Integration with the SDL Process Guidance
- Customized Security queries
- The SDL Final Security Report

Tool Extensibility

Getting Started

The SDL Process Template includes a library of resources to help you begin your SDL-based project. You can find these in the Documents/Process Guidance folder of your Team Project.

- Review the Online SDL Process Guidance
- Create an SDL Security Plan
- Complete Security Requirements Questionnaire
- Perform a Security Risk Assessment
- Model Threats

Customizing the Template

The SDL Process Template provides custom Check-in Policies to automate evaluation of the SDL Requirements. These policies need to be manually enabled as part of your Template setup.

- How to: Configure SDL Check-In Policies
- How to: Customize Security Cause/Effect fields

Online SDL Resources

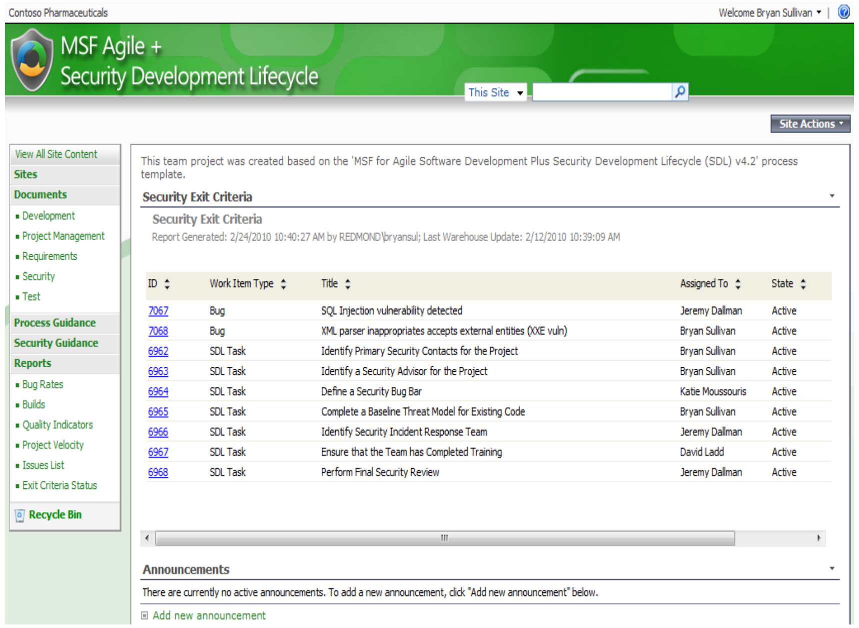
Much of the publicly available resources supporting the SDL are available and updated regularly on our website.

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

OWASP

MSF Agile + SDL Template for VSTS



This team project was created based on the 'MSF for Agile Software Development Plus Security Development Lifecycle (SDL) v4.2' process template.

Security Exit Criteria

Security Exit Criteria

Report Generated: 2/24/2010 10:40:27 AM by REDMOND\bryansul; Last Warehouse Update: 2/12/2010 10:39:09 AM

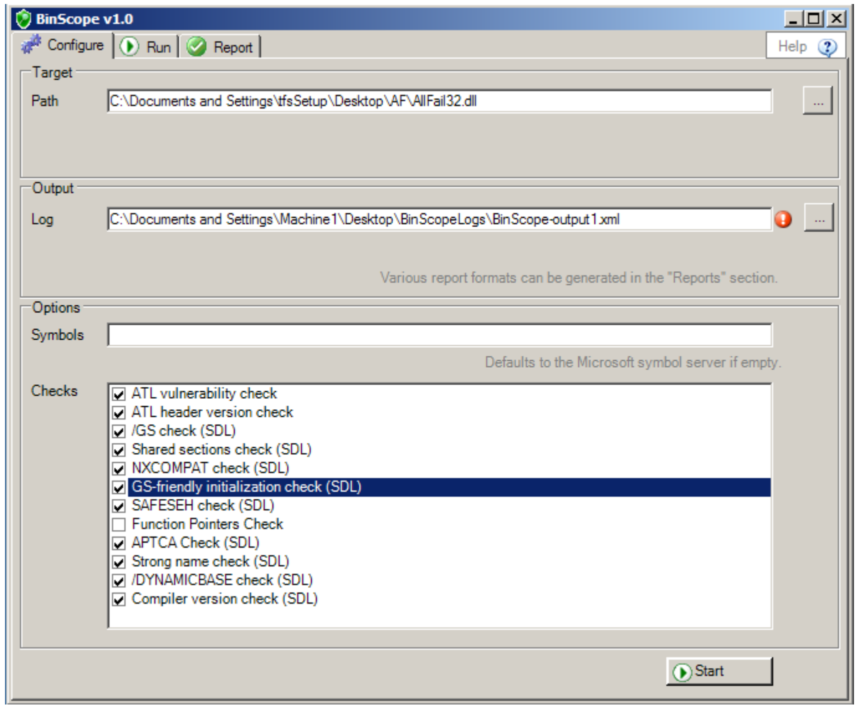
ID	Work Item Type	Title	Assigned To	State
7067	Bug	SQL Injection vulnerability detected	Jeremy Dallman	Active
7068	Bug	XML parser inappropriates accepts external entities (XXE vuln)	Bryan Sullivan	Active
6962	SDL Task	Identify Primary Security Contacts for the Project	Bryan Sullivan	Active
6963	SDL Task	Identify a Security Advisor for the Project	Bryan Sullivan	Active
6964	SDL Task	Define a Security Bug Bar	Katie Moussouris	Active
6965	SDL Task	Complete a Baseline Threat Model for Existing Code	Bryan Sullivan	Active
6966	SDL Task	Identify Security Incident Response Team	Jeremy Dallman	Active
6967	SDL Task	Ensure that the Team has Completed Training	David Ladd	Active
6968	SDL Task	Perform Final Security Review	Jeremy Dallman	Active

Announcements

There are currently no active announcements. To add a new announcement, click "Add new announcement" below.

- 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



Target

Path: C:\Documents and Settings\tfsSetup\Desktop\AF\AllFail32.dll

Output

Log: C:\Documents and Settings\Machine1\Desktop\BinScopeLogs\BinScope-output1.xml

Various report formats can be generated in the "Reports" section.

Options

Symbols:

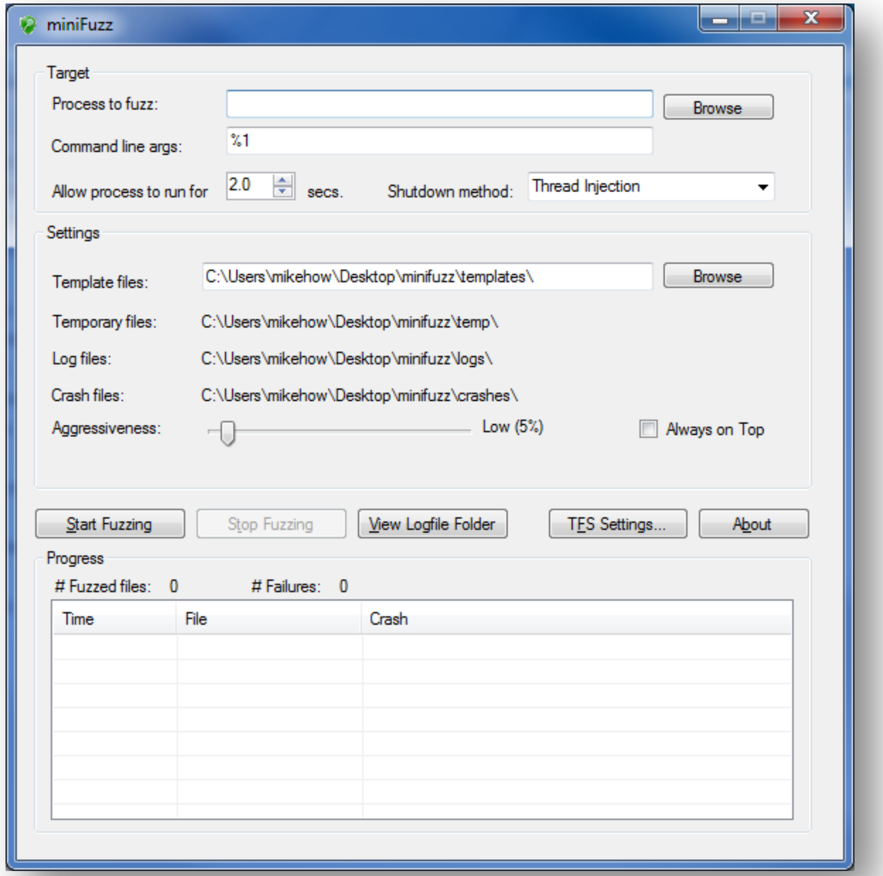
Defaults to the Microsoft symbol server if empty.

Checks:

- [x] ATL vulnerability check
- [x] ATL header version check
- [x] /GS check (SDL)
- [x] Shared sections check (SDL)
- [x] NXCOMPAT check (SDL)
- [x] GS-friendly initialization check (SDL)
- [x] SAFESEH check (SDL)
- [ ] Function Pointers Check
- [x] APTCA Check (SDL)
- [x] Strong name check (SDL)
- [x] /DYNAMICBASE check (SDL)
- [x] Compiler version check (SDL)

- 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 File Fuzzer



miniFuzz

Target

Process to fuzz:

Command line args: %1

Allow process to run for 2.0 secs. Shutdown method: Thread Injection

Settings

Template files: C:\Users\mikehow\Desktop\minifuzz\templates\

Temporary files: C:\Users\mikehow\Desktop\minifuzz\temp\

Log files: C:\Users\mikehow\Desktop\minifuzz\logs\

Crash files: C:\Users\mikehow\Desktop\minifuzz\crashes\

Aggressiveness: Low (5%) Always on Top

Progress

Fuzzed files: 0 # Failures: 0

Time	File	Crash

- 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)

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

Microsoft Security Development Lifecycle

United States Change | All Microsoft Sites

Security Development Lifecycle
Secure software made easier

Highlights

Play Elevation of Privilege threat modeling card game
Learn how to define and examine possible threats to software or computer systems.

Register for the March 11 TechNet webcast "Harmonizing Security Compliance and the SDL"
Find out how to harmonize a compliance-driven program with the secure engineering discipline.

MSF-Agile+SDL Process Template
Integrate the SDL into your Agile Development project.

Microsoft® Security Development Lifecycle

Recent Posts
Introducing the InfoSec Assessment & Protection Suite
Announcing SDL for Agile Development Methodologies

Introducing the InfoSec Assessment & Protection Suite

The Information Security Tools (IST) team has released the InfoSec Assessment & Protection (A&P) Suite. It's a suite made up of protection and assessment tools which include:

msdn
Windows Developer Center

Best Practices for the Security API
Certificate Enrollment API
Cryptography
Cryptography API: Next Generatio
Parental Controls
SDL - Process Guidance
SDL Process
Introduction
Pre-SDL Requirements: Se
Phase 1: Requirements
Phase 2: Design
Phase 3: Implementation
Phase 4: Verification
Phase 5: Release
Post-SDL Requirement: Re
SDL for Agile Development
SDL Process for Line-of-Busin
SDL Process – Appendix
Security Devices

MSDN › MSDN Library › Win32 and COM Development › Security

Microsoft Security Development Life

Introduction
Pre-SDL Requirements: Security Training
Phase 1: Requirements
Phase 2: Design
Phase 3: Implementation
Phase 4: Verification
Phase 5: Release
Post-SDL Requirement: Response

Content Disclaimer

The following documentation on the Microsoft Security Development
This documentation is not an exhaustive reference on the SDL proces

Microsoft

Microsoft® Security Development Lifecycle

Simplified Implementation of the Microsoft SDL
February 2, 2010

SDL Portal

http://www.microsoft.com/sdl

SDL Blog

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

SDL Process on MSDN (Web)

http://msdn.microsoft.com/en-us/library/cc307748.aspx

Simplified Implementation of the Microsoft SDL

http://go.microsoft.com/?linkid=9708425

OWASP

Questions?