Building More Secure Commercial Software: The Trustworthy Computing Security Development Lifecycle

Steven B. Lipner
Microsoft Corporation

With the growth of the Internet as a vehicle for commercial, governmental, and personal communications and information sharing, the importance of providing trustworthy computing facilities that will resist hostile attack has grown dramatically. In response to this growing need, Microsoft has developed the Trustworthy Computing Security Development Lifecycle (SDL), an integrated process for improving the security of commercial software as it is being developed. This paper describes the phases of the SDL from initial requirements definition through the Final Security Review before software release, and summarizes some of the improvements in security demonstrated by software that has completed the SDL.

Introduction

It is imperative that all software vendors address security threats. Security is a core requirement for software vendors, driven by market forces, the need to protect critical infrastructures, and the need to build and preserve widespread trust in computing. A major challenge for all software vendors is to create more secure software that requires less updating through patches and less burdensome security management.

For the software industry, the key to meeting today's demand for improved security is to implement repeatable processes that reliably deliver measurably improved security. Therefore, software vendors must transition to a more stringent software development process that focuses, to a greater extent, on security. Such a process is intended to minimize the number of security vulnerabilities extant in the design, coding, and documentation and to detect and remove those vulnerabilities as early in the development lifecycle as possible. The need for such a process is greatest for enterprise and consumer software that is likely to be used to process inputs received from the Internet, to control critical systems likely to be attacked, or to process personally identifiable information.

There are three facets to building more secure software: repeatable process, engineer education, and metrics and accountability. This document focuses on the repeatable process aspect of the SDL, although it does discuss engineer education and provide some overall metrics that show the impact to date of application of a subset of the SDL.

If Microsoft's experience is a guide, adoption of the SDL by other organizations should not add unreasonable costs to software development. In Microsoft's experience, the benefits of providing more secure software (e.g., fewer patches, more satisfied customers) outweigh the costs.

The SDL involves modifying a software development organization's processes by integrating measures that lead to improved software security. This document summarizes those measures and describes the way that they are integrated into a typical software development lifecycle. The intention of these modifications is not to totally overhaul the process, but rather to add well-defined security checkpoints and security deliverables.

This document outlines, at a very high level, the concepts of the SDL, shown in Figure 1.

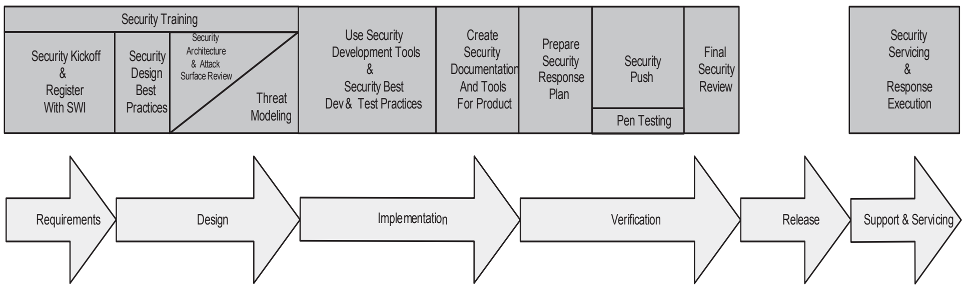
Security Training
Security Kickoff & Register With SWI
Security Design Best Practices
Security Architecture & Attack Surface Review
Threat Modeling
Use Security Development Tools & Security Best Dev & Test Practices
Create Security Documentation And Tools For Product
Prepare Security Response Plan
Security Push
Pen Testing
Final Security Review
Security Servicing & Response Execution
Requirements
Design
Implementation
Verification
Release
Support & Servicing

Figure 1: SDL Improvements to the Microsoft development process.

The Security Development Lifecycle Process

An organization that seeks to develop secure software must take responsibility for ensuring that its engineering population is appropriately educated. At Microsoft, all personnel involved in developing software must go through yearly "security refresher" training.

Requirements Phase

The need to consider security "from the ground up" is a fundamental tenet of secure system development. During the requirements phase, the product team makes contact with the central security team to request the assignment of a security advisor who serves as point of contact, resource, and guide as planning proceeds. The security advisor assists the product team by reviewing plans, making recommendations, and ensuring that the security team plans appropriate resources to support the product team's schedule. The security advisor remains the product team's point of contact with the security team from project inception through completion of the Final Security Review and software release.

The requirements phase is the opportunity for the product team to consider how security will be integrated into the development process, identify key security objectives, and otherwise maximize software security while minimizing disruption to plans and schedules. As part of this process, the team needs to consider how the security features and assurance measures of its software will integrate with other software likely to be used together with its software.

Design Phase

The design phase identifies the overall requirements and structure for the software. From a security perspective, the key elements of the design phase are:

- Define security architecture and design guidelines: Define the overall structure of the software from a security perspective, and identify those components whose correct functioning is essential to system security.

- Document the elements of the software attack surface. Given that software will not achieve perfect security, it is important that only features that will be used by the vast majority of users be exposed to all users by default, and that those features be installed with the minimum feasible level of privilege.

- Conduct threat modeling. Using a structured methodology, the threat modeling process identifies threats that can do harm to each asset and the likelihood of harm being done (an estimate of risk), and helps identify countermeasures that mitigate the risk.

Implementation Phase

During the implementation phase, the product team codes, tests, and integrates the software. Steps taken to remove security flaws or prevent their initial insertion significantly reduces the likelihood that security vulnerabilities will make their way into the final version of the software.

The elements of the SDL that apply are:

- From the threat model task, understand which are the high-risk components.

- Apply coding and testing standards.

- Apply security testing tools including fuzz testing tools.

- Apply static-analysis code scanning tools.

- Conduct security code reviews.

Verification Phase

The verification phase is the point at which the software is functionally complete and enters beta testing. Microsoft introduced the security push during the verification phase of Windows Server 2003 and several other software versions in early 2002. The main purpose of the security push is to review both code that was developed or updated during the implementation phase and "legacy code" that was not modified.

Release Phase

During the release phase, the software should be subject to a Final Security Review ("FSR").

The FSR is an independent review of the software conducted by the central security team for the organization. Tasks include reviewing bugs that were initially identified as security bugs, but on further analysis were determined not to have impact on security, to ensure that the analysis was done correctly. An FSR also includes a review of the software's ability to withstand newly reported vulnerabilities affecting similar software. An FSR for a major software version will require penetration testing and, potentially, the use of outside security review contractors to supplement the security team.

Support and Servicing Phase

Despite the application of the SDL during development, state of the art development practices do not yet support shipping software that is completely free from vulnerabilities – and there are good reasons to believe that they will never do so. Product teams must prepare to respond to newly-discovered vulnerabilities in shipping software.

Part of the response process involves preparing to evaluate reports of vulnerabilities and release security advisories and updates when appropriate. The other component of the response process is conducting a post-mortem of each reported vulnerability and taking action as necessary. Depending on the nature and severity of reported vulnerabilities, this action may include repeating earlier phases of the process to address new threats.

Results of Implementing the Security Development Lifecycle at Microsoft

It is premature for Microsoft to make conclusive claims that the SDL improves the security of Microsoft software, but the results to date are encouraging.

Windows Server 2003 was the first operating system release at Microsoft that implemented large portions of the SDL. Figure 2 shows the number of security bulletins and the severity of each bulletin issued within the year after release for the two most recent Microsoft server operating systems: Windows 2000 and Windows Server 2003.

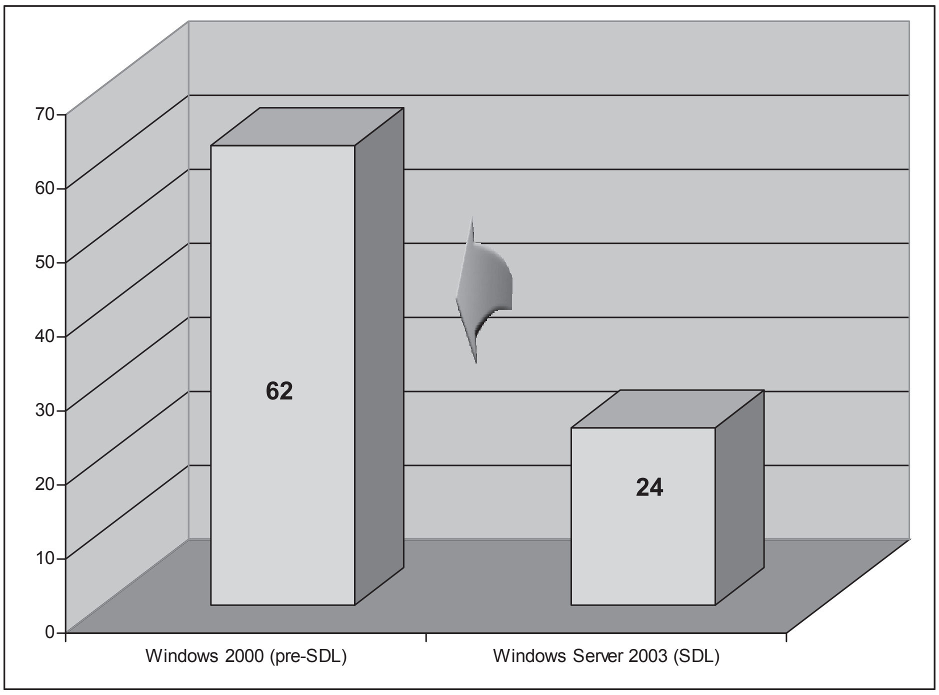

Figure 2: Windows pre- and post-SDL Critical and Important Security Bulletins

The SQL Server and Exchange Server product teams each conducted a security push (including threat modeling, code reviews, and security testing) before releasing a service pack. The results of the SQL Server security push were incorporated in SQL Server 2000 Service Pack 3, and the results of the Exchange Server security push were incorporated in Exchange 2000 Server Service Pack 3. Figures 3 and 4 shows the numbers of security bulletins released in equal periods before and after the release of the respective service.

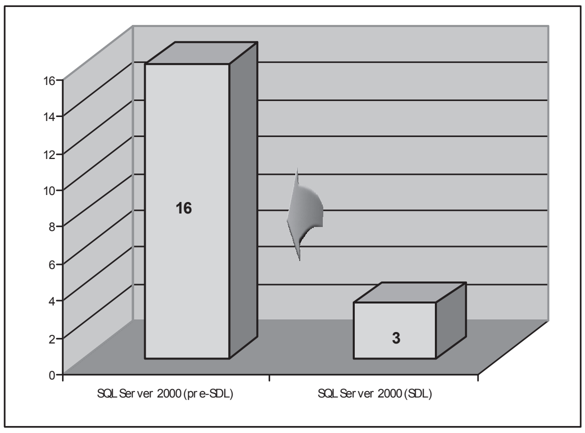

Figure 3: SQL Server 2000 pre- and post-SDL Security Bulletins

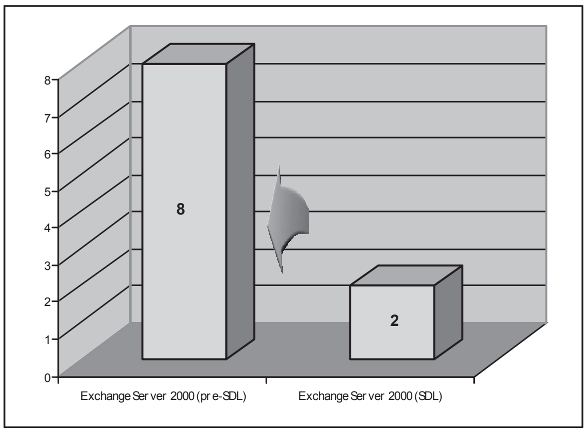

Figure 4: Exchange Server 2000 pre- and post-SDL Security Bulletins

Another encouraging example is the Internet Information Server component of Windows Server 2003 (IIS 6); in the two years since its release; Microsoft has issued one security bulletin affecting the Web server, and this was in a component that is not installed by default.

Conclusions

Microsoft's experience indicates that the SDL is effective at reducing the incidence of security vulnerabilities. Initial implementation of the SDL (in Windows Server 2003, SQL Server 2000 Service Pack 3, and Exchange 2000 Server Service Pack 3) resulted in significant improvements in software security, and subsequent software versions, reflecting enhancements to SDL, appear to be showing further improvements in software security.

Incremental implementation of the elements that comprise SDL has yielded incremental improvements, which we view as one sign of an effective process. The process is not perfect, and is still evolving – and is unlikely either to reach perfection or to cease evolving in the foreseeable future.

The development and implementation of the Security Development Lifecycle represent a major investment for Microsoft, and a major change in the way that software is designed, developed, and tested. The increasing importance of software to society emphasizes the need for Microsoft and the industry as whole to continue to improve software security; therefore, both this paper on the SDL and books on specific techniques have been published in an effort to share Microsoft's experience across the software industry.

References

Lipner, Steve and Michael Howard, "The Trustworthy Computing Security Development Lifecycle" http://msdn.microsoft.com/security./sdl

Notices