MS_logo_KMICROSOFT SDL - DEVELOPER STARTER KIT:

SECURE DESIGN PRINCIPLES (LEVEL 100)

Guide

Version 1.0

The following documentation provides presenter’s notes for the Microsoft Security Development Lifecycle (SDL) Secure Design Principles (Level 100) presentation.

For the latest information, please see [http://www.microsoft.com/sdl](http://go.microsoft.com/?linkid=9672761).

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in, or introduced into a retrieval system, or transmitted in any form, by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

© 2009 Microsoft Corporation. All rights reserved.

Microsoft, SQL Server, Visio, Visual Basic, Visual C#, Visual C++, Visual Studio, Windows, Windows Server, and Windows Vista are trademarks of the Microsoft group of companies.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Contents

[1.0 Security Development Lifecycle Content 2](#_Toc233612831)

[1.1 Introduction 2](#_Toc233612832)

[1.2 System Requirements 2](#_Toc233612833)

[1.3 Presentation Themes 2](#_Toc233612834)

[2.0 SDL Secure Design Principles 2](#_Toc233612835)

[Overview 2](#_Toc233612836)

[Presentation Transcript 2](#_Toc233612837)

[Presentation Voiceover 2](#_Toc233612838)

[Slide 2 – Title Slide 3](#_Toc233612839)

[Slide 3 – Agenda 3](#_Toc233612840)

[Slide 4 – Microsoft Security Development Lifecycle (SDL) 3](#_Toc233612841)

[Slide 5 – SDL Secure Design Principles 4](#_Toc233612842)

[Slide 6 – SDL Core Principle: Attack Surface Reduction 4](#_Toc233612843)

[Slide 7 – Attack Surface Example 5](#_Toc233612844)

[Slide 8 – Attack Surface Analysis 5](#_Toc233612845)

[Slide 9 – Attack Surface Analysis Tips 5](#_Toc233612846)

[Slide 10 – It’s Not Just About Turning Stuff Off 6](#_Toc233612847)

[Slide 11 – Attack Surface Reduction Examples 6](#_Toc233612848)

[Slide 12 – SDL Core Principle: Basic Privacy 6](#_Toc233612849)

[Slide 13 – Important Note: Security Does Not Always Guarantee Privacy 6](#_Toc233612850)

[Slide 14 – Primary objectives When Developing Privacy-aware applications 7](#_Toc233612851)

[Slide 15 – Understanding Application Behaviors and Concerns 7](#_Toc233612852)

[Slide 16 – Microsoft Privacy Guidelines for Developing Products and Services 8](#_Toc233612853)

[Slide 17 – SDL Core Principle: Threat Modeling 8](#_Toc233612854)

[Slide 18 – Threat modeling In a Nutshell 9](#_Toc233612855)

[Slide 19 – Microsoft SDL Threat Modeling Tool 9](#_Toc233612856)

[Slide 20 – SDL Core Principle: Defense In Depth 9](#_Toc233612857)

[Slide 21 – Defense In Depth Example 10](#_Toc233612858)

[Slide 22 – SDL Core Principle: Least Privilege 10](#_Toc233612859)

[Slide 23 – Least Privilege Example 10](#_Toc233612860)

[Slide 24 – Least Privilege Tips 11](#_Toc233612861)

[Slide 25 – SDL Core Principle: Secure Defaults 11](#_Toc233612862)

[Slide 26 – Secure Defaults Example 11](#_Toc233612863)

[Slide 27 – Conclusion 12](#_Toc233612864)

[SLIDE 28 - Appendix 12](#_Toc233612865)

[Slide 29 – Microsoft Security Development Lifecycle (SDL) 12](#_Toc233612866)

[Slide 30 – Threat Modeling Resources 12](#_Toc233612867)

[Slide 31 – Microsoft Developer Network (MSDN) Security Developer Center 13](#_Toc233612868)

[Slide 32 – Secure Development Blogs 13](#_Toc233612869)

# 1.0 Security Development Lifecycle Content

## 1.1 Introduction

“The Microsoft Security Development Lifecycle (SDL) is an industry-leading software security assurance process. A Microsoft-wide initiative and a mandatory policy since 2004, the SDL has played a critical role in embedding security and privacy in Microsoft software and culture. Combining a holistic and practical approach, the SDL introduces security and privacy early and throughout all phases of the development process. It has led Microsoft to measurable and widely-recognized security improvements in flagship products such as Windows Vista, Windows Server (2003 and 2008) and SQL Server. Microsoft is publishing the detailed SDL process guidance as part of its commitment to enable a more secure and trustworthy computing ecosystem.” -- [The Microsoft SDL 3.2 Whitepaper](http://go.microsoft.com/?linkid=9672762)

To help promote the adoption and awareness of the Microsoft SDL, Microsoft is developing content and demonstrations specifically for external developer audiences. The remainder of this document provides individuals who will present this content internally within their respective organizations with a transcript for the Microsoft SDL Training:

* SDL Core Training – Secure Design Principles (Level 100)

## 1.2 System Requirements

In order to use this content, a system that is capable of running [Microsoft PowerPoint 2003](http://www.microsoft.com/powerpoint) or later is required.

## 1.3 Presentation Themes

The Microsoft PowerPoint deck that accompanies this Presenter’s Guide has been intentionally provided with very limited graphics and formatting. The Microsoft PowerPoint presentation materials have been designed in this fashion to enable individuals who will present this content internally within their respective organizations to incorporate the content into custom PowerPoint themes, styles, and templates with minimal required effort.

# 2.0 SDL Secure Design Principles

## Overview

Building trusted software requires that security and privacy be considered at every stage of an application’s software development lifecycle (SDLC). One of the most effective stages within which security and privacy best practices can have the greatest impact is the design stage. Decisions regarding the design and architecture of an application greatly influence the final state of the application, and as such, effectively incorporating security and privacy considerations during the design stage will favorably impact the resulting security and privacy posture of the application.

The insights gleaned by Microsoft, which are incorporated in its SDL, and more specifically, in this presentation focusing on Secure Design Principles, are provided as a way for external developer communities to enhance its application development practices and the security of its applications.

## Presentation Transcript

This Presentation Transcript section of this document provides a transcript for each slide in the Secure Design Principles (Level 100) presentation. The precise transcript text provided herein is also incorporated into the notes section of each slide in the Microsoft PowerPoint Secure Design Principles (Level 100) presentation itself for ease of reference.

## Presentation Voiceover

A voiceover of the Secure Design Principles (Level 100) presentation transcript below, approximately 34 minutes in length is also available to assist the presenter in becoming sufficiently acclimated with the subject matter addressed in the Secure Design Principles (Level 100) presentation, as well as to better understand the author’s perspective behind each slide in the presentation.

### Slide 2 – Title Slide

The Secure Design Principles (Level 100) presentation introduces the role that the Microsoft Security Development Lifecycle (SDL) fulfills in trusted application design and provides an overview of secure design principles employed within the SDL, including: a) attack surface reduction; b) basic privacy; c) threat modeling; d) defense in depth; e) least privilege; and f) secure defaults.

Addressing this subject matter will enable our organization to enhance our application development practices and the security of our applications

*Note:* This is a level 100 presentation meant to familiarize you with security design fundamentals and principles. These fundamentals and principles will be built upon in later SDL presentations.

### Slide 3 – Agenda

In this presentation we will complete a high-level overview of the SDL and the important role it fulfills in the design stage of an application’s software development lifecycle. We will also review the secure design principles employed within the SDL that has helped Microsoft better deliver safer and more trusted applications to its customers since the inception of the SDL in 2004.

### Slide 4 – Microsoft Security Development Lifecycle (SDL)

The Microsoft SDL is a holistic and comprehensive approach that leverages education, process, technology and executive commitment to consistently create more secure software internally within and external of Microsoft. Since 2004, all internal Microsoft developers have been required to adhere to the SDL, and Microsoft has updated the SDL every six (6) months to address any emerging threats since its inception.

True to its name, the SDL was created to complement (rather than disrupt) the software development life cycle. The core phases and principles of the SDL include:

**Training phase:** Every Microsoft developer must complete mandatory security training focusing on secure application development practices. Training session topics include topics such as threat modeling, secure development and testing practices, and security for application development managers.

**Requirements phase:** Requirements for security and privacy must accompany functional requirements of the software that is being created. Such requirements may include the use of encryption, authentication, and other security measures based on the business requirements, exposure and sensitive data. To that end, a security and privacy risk analysis is performed at this stage. In addition, the threshold for security and privacy (or “bug-bar”) is defined during this phase to ensure that bugs with certain severity are addressed and resolve before the software is officially released.

**Design phase:** Eradicating coding bugs with security implications is not sufficient. Design vulnerabilities can have a substantial detrimental impact on security and are much more difficult to address during the verification phase. To that end, threat modeling is a critical SDL requirement and a Microsoft security innovation that is recognized by analysts as the next evolution in creating more secure software. Through threat modeling, architects and developers at Microsoft are able to approach security in a structured and methodical way from an attacker’s perspective. This allows Microsoft to identify and reduce the attack surface and mitigate the risk of potential security design issues.

**Implementation phase:** This is the application code development phase where code is written by developers using industry best practices and analyzed with both internal and externals tools (such as static code analyzers and special security debuggers) to help ensure that those best practices are being followed. Requirements are also specified by the SDL in this phase to ensure that applications are built using the latest compilers versions and built-in compiler protection features.

**Verification phase:** This is the quality assurance phase within which rigorous security testing is conducted in addition to typical functional testing procedures.

**Release phase:** The final security review is the major milestone that a Microsoft product team must pass in order to release a product under the SDL. During this meeting, security experts and the development team review all of the activities, mitigations and security artifacts that are relevant to the project in order to ensure that the security quality requirements are satisfied. During this phase, the product team defines a response plan describing procedures, accountabilities and contact information in case security vulnerabilities are discovered after the product is operational and used by customers.

**Response phase:** After an application is released, the Microsoft Security Response Center (MSRC) handles any security issues that are uncovered “in the wild” and mobilize product teams within Microsoft to provide timely fixes for security issues.

In summary, secure software development requires executive commitment, ongoing process improvement, education and training (from VPs to product managers to developers to testers), tools to aid in detecting security vulnerabilities, and incentives and consequences to ensure everyone adheres to the SDL process.

As was previously indicated, this presentation focuses on the secure design principles of the SDL.

### Slide 5 – SDL Secure Design Principles

After the requirements for a solution has been identified in the early stages of an application’s software development lifecycle, the next step is to design and architect a solution that satisfies those identified requirements. Developing trusted applications requires that sound security and privacy decisions be made early in the design phase because decisions made at this stage will highly influence subsequent efforts in the latter stages of the software development lifecycle and the final state of the application. Microsoft has found that by adopting this approach, application development costs (such as those required to address and resolve security and privacy issues) are significantly reduced compared to if security and privacy were considered later in the SDLC or not at all. This is because applications developed against more secure and privacy aware designs tend to be exposed to fewer threats and contain less vulnerabilities. Microsoft helps ensure that security and privacy considerations are incorporated into its application design efforts through the SDL by applying the following secure design principles.

In the remainder of this presentation, we will briefly review each of these principles and mention how they can be applied to better ensure that application designs consist of sufficient and effective security and privacy best practices.

Lastly, the insights gleaned by Microsoft, which are incorporated in its SDL, and more specifically, in this presentation focusing on Secure Design Principles, are being shared with each of you as a way for our organization to enhance our application development practices and the security of our applications.

### Slide 6 – SDL Core Principle: Attack Surface Reduction

The attack surface of an application is the portion of the program (code and functionality) that is exposed to a particular person or another program. For example, an open network port and a user-interface are examples of an application’s attack surface.

One of the most effective secure design principles that can be used to protect an application from malicious acts is attack surface reduction (ASR). The principle of ASR is to minimize the attack surface while still satisfying the functional requirements of the application. Secure coding will reduce, but not eliminate all vulnerabilities in your application; however, by reducing the attack surface, you minimize the number of vulnerabilities that the attacker can discover and attempt to exploit.

### Slide 7 – Attack Surface Example

Here’s an example of attack surface. Let’s pretend we are a home security company and we want to protect this home from a burglar breaking in the home and stealing the valuables inside. What is our attack surface?

(Mouse click)

At the front of the house we have several windows and doors that a burglar could use or exploit.

(Mouse click)

At the side of the house we also have a couple windows a burglar could use or exploit.

(Mouse click)

And then finally, do not forget the chimney!

Like this house, our applications have various points that are exposed to people or other programs and computers. Each one of these can be exploited by a malicious user, and with attack surface reduction our goal is to minimize the number of potential vulnerabilities a malicious user could exploit.

### Slide 8 – Attack Surface Analysis

In order to reduce the attack surface of an application, application designers need to first know how to measure the attack surface.

The attack surface is defined by the set of interfaces, or entry points, to the program. Attack surface analysis (ASA) is the process of identifying and understanding all of the entry points that comprise the attack surface, and is successfully performed by enumerating all of the interfaces, protocols, and code execution paths. Another important element of ASA is understanding the trust levels required to access each entry point.

For each entry point, you must consider the importance of the feature that it enables. For features that are not important to a vast majority of the users, turn the feature off, disable it by default, or do not even install it by default; force the users that really want or need the feature to take explicit action to obtain that feature. This way, any vulnerability related to that specific feature will affect a very small percentage of the product’s user base.

Next consider which specific classes of users require that feature, and then restrict its use to those classes. For example, do not default to making the feature remotely accessible, do not default to allowing anonymous access, do not default to running with more privilege than is needed, etc.

A significant aspect of ASR is restricting who has access to a particular product feature, and how such users may obtain and use that access.

### Slide 9 – Attack Surface Analysis Tips

ASA is an iterative process: for each feature that you analyze, you must also analyze all of its sub-features. And again, you want to restrict access to features as much as possible.

For example, if your application in general processes files, configure it to read only the most common file types that it accepts; force the user to explicitly configure it to process the less commonly used file types. Also ensure that when the program writes files that it sets the proper ownership and rights on the file; do not create executable files unless those files must be executable.

Disable older, faulty, and less used protocols, such as SSL V2 and PCT. Force users to use more robust alternatives, such as SSL V3 and TLS, or force them to explicitly configure applications to accept those older protocols.

If your application provides a service or implements a protocol, restrict the commands that it accepts by default to those that are most commonly needed and used. Force the administrator to explicitly configure other commands if they want to accept them.

### Slide 10 – It’s Not Just About Turning Stuff Off

While ASR focuses on restricting access, it is not strictly about disabling or not installing features.

For instance, instead of using UDP as a network protocol, use TCP which can be more easily secured. Or instead of making a network service Internet accessible, make it local network accessible only until unless needed.

You can also use ASR to enforce the principle of least privilege, which will be discussed later, by designing the program to run with the lowest set of privileges required to perform its function.

### Slide 11 – Attack Surface Reduction Examples

Here are some examples of how ASR has been applied in the latest versions of Microsoft products that have previously encountered security complications:

* Authentication before interaction achieves ASR by disallowing anonymous access by default;
* Firewall on by default closes all but specifically required ports;
* Many services are now off by default, and when turned on are running as low privileged network services;
* When services are necessary out-of-the-box, they are restricted to localhost access; and
* Functions or features that have been proven to introduce unnecessary and undesirable risk in the past are also now turned off by default.

Note: “Network service” refers to the lesser privileged account running the service. Therefore an attacker that defeats the security of IIS 6 obtains relatively few and weak privileges, whereas that attacker defeating the security of IIS 4 or 5 was rewarded with Admin privileges.

### Slide 12 – SDL Core Principle: Basic Privacy

In addition to the SDL design principle of attack surface reduction, another core principle that needs to be considered when developing trusted software is that of privacy. Privacy, like security, is another key factor when developing trusted applications; however, they are not the same.

Privacy focuses on the control and choices users have regarding the use, collection and distribution of their personal information. Security, on the other hand, is applied to protect assets, including personal information, from threats.

Again, when designing trusted applications, both privacy and security together need to be evaluated. The SDL helps application designers create more privacy-aware applications by establishing during the design phase privacy best practices, standards, and guidelines.

### Slide 13 – Important Note: Security Does Not Always Guarantee Privacy

A common myth regarding the relationship between security and privacy is that if a system is sufficiently secure then privacy is also preserved. However, this may not always be the case. A security breach can certainly result in a loss of privacy (for instance, credit card information may be accessed by unauthorized users), but it is also possible for a secure system to cause a loss of privacy without a breach.

Consider this secure, but privacy-violating scenario: Securely storing personal information and then sending that information using a securely encrypted communication channel to third parties without properly notifying and receiving consent from the user may be securely implemented but obviously does not take into consideration the rights of the user- some rights may have legal implications! In this scenario, the user’s privacy is compromised due to the inappropriate act of a user / application vs. due to a security breach.

### Slide 14 – Primary objectives When Developing Privacy-aware applications

When developing privacy-aware applications, three primary objectives must be satisfied:

* Fulfilling legal obligations;
* Increasing customer trust; and
* Preventing blocked deployments

**Fulfilling Legal Obligations**

Depending on how an application behaves, certain controls and documentation must be established in order to fulfill legal obligations. For instance, if an application involves users that are children, then it must be compliant with the Children’s Online Privacy Protection Act (COPPA). Or if an application transfers any personally identifiable information (PII), regardless if sensitive or non-sensitive, then certain legal obligations arise. Several prevalent regulations involving PII include:

* COPPA - Children's Online Privacy Protection Act – Protects the privacy of children under 13 years of age, including PII and images with identifiable locations, etc.
* GLBA - Gramm-Leach-Bliley Act – Mandatory compliance for financial institutions to provide Privacy Notices to consumers regardless of whether the PII will be disclosed to external parties
* HIPPA - Health Insurance Portability and Accountability Act – Privacy Rule regulates use and disclosure of Protected Health Information that can be linked to an individual
* CFAA – Computer Fraud and Abuse Act – Outlines criminal offenses for accessing and modifying computer systems without authorization
* FTC – Federal Trade Commission – Bureau of Consumer Protection investigates and enforces laws with respect to fraud, privacy, and identity protection, among others.
* EU – European Union – Includes international law that supersedes member state law when privacy conflicts involving PII arise.

**Increasing Customer Trust**

Building great software is not enough; you need to earn and increase your customers’ trust in your software. By focusing on privacy considerations, you can earn required trust by designing more trusted applications that increase transparency in the user experience and will empower the user to control their personal data through guidance that is easy to understand and actionable. In a later slide, we will talk more about how we can achieve and can earn customer trust through adopting the SDL.

**Preventing Blocked Deployments**

The final privacy objective is to prevent blocked deployments. Blocked deployments are any instances where applications cannot be deployed into production environments due to some adverse behavior of the application or lack of documentation regarding an undesirable behavior. To prevent blocked deployments, designers and developers need to focus on features that have considerable privacy implications, such as continuous monitoring or discrete anonymous transfer, and ensure that appropriate and sufficient controls and disclosures have been applied accordingly.

For example, when Windows XP was rolled out, there were many new “phone home” scenarios that had not been extensively documented. In order to unblock deployments (to the tune of approximately $300 million US dollars; including customers, such as the US Air Force and the CIA), Microsoft needed to author a 180 page whitepaper that described all potential Internet Communication and the controls that administrators could use to redirect or disable such communication. Note that any ping to the Internet has privacy implications - even the transmission of IP addresses may need to be documented in the Privacy Notice.

### Slide 15 – Understanding Application Behaviors and Concerns

The primary privacy objectives associated with developing trusted applications were presented in the previous slide. As previously explained, certain behaviors of an application could create legal obligations that need to be met and how those behaviors could also block the deployment of an application. The table on this slide shows some of the common application behaviors and the legal obligations and blocked deployment scenarios that could arise because of those behaviors.

For example, if an application is designed for users under 13 years of age, then legal obligations, such as those described in the Children Online Privacy Protection Act (COPPA), must be met. As another example, if an application transfers personal information, then satisfying legal obligations from the Gramm-Leach-Bliley Act (GLBA) and the Health Insurance Portability and Accountability Act could be required.

Application designers need to understand the behavior of the applications and corresponding privacy concerns implied by those behaviors. Microsoft has developed the *Microsoft Privacy Guidelines for Developing Products and Services* to help application development teams better understand privacy implications associated with application behavior.

### Slide 16 – Microsoft Privacy Guidelines for Developing Products and Services

In order to help application development teams better develop privacy-compliant products and services, Microsoft has released the *Microsoft Privacy Guidelines for Developing Products and Services*. This document provides common definitions and rules for developing better privacy-compliant products and services.

The document is divided into two sections: The first section contains definitions and key concepts including data types, notice, consent, etc. The second section contains rules categorized by specific development scenarios, such as collection and transfer of personally identifiable information (PII), storage of data on the customer’s system, and onward transfer of PII to third parties. There are also specific scenarios for products or services that collect age or are attractive to children, and for products that are deployed in enterprises. Products deployed in enterprises are a special case, because the developers’ obligation transitions from the user to the enterprise administrator—you need to enable to enterprise administrator to fulfill their company’s privacy policies.

At Microsoft, our goal is that our customers will be empowered to control the collection, use and distribution of their personal information through our products and services, and so any externally released application or Web site must comply with the clearly defined rules and guidelines set forth in the *Microsoft Privacy Guidelines for Developing Products and Services*.

Within the SDL, a privacy bug bar is defined and is used to measure the impact of non-compliance with the rules of the Microsoft Privacy Guidelines for Developing Products and Services.

**Downloading the Microsoft Privacy Guidelines**

The current version (version 2.1a released 04/26/2007) of the *Microsoft Privacy Guidelines for Developing Product and Services* can be downloaded from:

<http://www.microsoft.com/downloads/details.aspx?FamilyId=C48CF80F-6E87-48F5-83EC-A18D1AD2FC1F&displaylang=en>.

A link to the most current version of this document can be found under the privacy section of the SDL training and resources link, located at: <http://msdn.microsoft.com/en-us/security/cc448120.aspx>.

### Slide 17 – SDL Core Principle: Threat Modeling

In this section of the presentation, the process called threat modeling, which Microsoft uses through the SDL process to understand and address any and all threats to an application, will be briefly explained.

It is important not to confuse threats with vulnerabilities. A threat is simply what an adversary might *try* to do to compromise a protected resource in the system. A vulnerability is a specific way that a threat is exploitable based on an unmitigated attack path.

A person in an application design group with security expertise typically leads the threat modeling activities, which begin with identification of all potential threats to the system and the assets accessed by the system.

Threat models must be revisited periodically to account for new threats resulting from new and evolving attack techniques.

Please recall that this portion of the presentation is meant only to provide you with a brief introduction to the threat modeling process.

### Slide 18 – Threat modeling In a Nutshell

At a high level, threat modeling consists of a number of activities conducted during the design phase of the application development process. These activities begin by envisioning the application as it will be used by typical users in a typical environment, and continue by identifying all of the potential threats to the application and to assets accessed via the application. During this process all security-related assumptions and external dependencies are documented, as are the “external security notes” – notes to help users and administrators understand the security boundaries of the application currently being developed.

The threat modeling process continues by creating a number of data flow diagrams (DFDs), which model the trust boundaries of the application and its components and the flow of data between the application and its environment, as well as the flow of data between components within the application.

Now that you have a completed model, the next step in the threat modeling process is to determine the types of threats facing the application (from the malicious user’s perspective) and list all of the DFD elements. The DFD elements represent the application assets that need to be protected from attack.

Knowing what needs to be protected, and how they will be attacked, enables you to choose appropriate mitigations for each threat. Note that we are still in the design stage of application development! We are now designing security controls into the product based upon the most likely threats; the most cost-effective juncture to address such considerations.

At this point, we need to review the threat model, the DFD elements (a.k.a. ‘assets’) that need protection, and the mitigations or defenses/countermeasures, to ensure that the mitigations do indeed protect the assets from the threats. If anything is found to be amiss, this is the time to start from the beginning of the threat-modeling process once again.

### Slide 19 – Microsoft SDL Threat Modeling Tool

Microsoft has published the threat modeling tool it uses internally to help automate aspects of the threat modeling process. The tool is available for download at <http://msdn.microsoft.com/en-us/security/dd206731.aspx>.

### Slide 20 – SDL Core Principle: Defense In Depth

In addition to attack surface reduction, privacy and threat modeling, another key design principle that should be leveraged to create trusted applications is the principle of defense in depth.

A key perspective of defense in depth is beginning the application design process with the mindset that all applications and hardware will ultimately fail. If you go back to the burglar breaking into a house scenario, all the doors and windows to the house could be locked to keep the burglar out. Those might fail, so an alarm system could also be installed. The alarm system might fail, so the valuables inside the house could be placed inside a safe and so on. In the context of designing and developing trusted applications, this means that privacy (e.g., cryptographic algorithms used to protect sensitive or personal data) and security (e.g., firewalls) features or mechanisms defending our applications will inevitably fail. Unfortunately most applications today are designed and implemented in such a way that the application can be compromised when a single, and often only, layer of defense fails or is breached.

With the defense in depth principle, applications are built in such a way that if one defense layer fails, there are additional layers of defense that can provide protection to the application. Think about it this way, if a malicious user is going to compromise our application then we should make their job as difficult as possible by implementing multiple layers of defense that they need to breach vs. just one. That is defense in depth!

### Slide 21 – Defense In Depth Example

The easiest way application designers can get started with this powerful principle is to evaluate their application designs and ask themselves if one layer of defense is breached, what other layers can provide additional protection to the application and the assets that it is protecting.

Here is an example of defense in depth and how this principle can be leveraged to make trusted applications more difficult to compromise. Consider a malicious user who wants to gain unauthorized access to sensitive data stored in a server, such as credit numbers or other personally identifiable information (PII).

(Mouse click)

Again most applications today are designed and developed with a single layer of defense in mind – typically a firewall.

(Mouse click)

If a malicious user is able to breach this single layer of defense, then that user is able to compromise the application.

(Mouse click)

On the other hand, an application designed with the defense in depth principle has multiple layers of defense protecting it. For instance, defense layers like input validation (application-level), smart card (host-level) access, and IP security (network-level) are examples of additional layers of defense that could protect an application.

(Mouse click)

So now, even if one layer of defense fails, there are additional layers of defense that can provide protection to the application and the attack is halted. In our example, while both the firewall and the second defense layer failed, the 3rd and 4th layers of defense were still able to halt the attack, thereby protecting the sensitive data.

### Slide 22 – SDL Core Principle: Least Privilege

The previous SDL secure design principle of defense in depth started with the notion that all software and hardware would fail at some point. With the least privilege principle, we assume that all applications will be compromised. However, thru employing the principle of least privilege, should a malicious user compromise an application the amount of damage that may be inflicted is limited.

### Slide 23 – Least Privilege Example

Pretend that we have a malicious user and an application running on a system that the user is hoping to compromise.

(Mouse click)

In this example, the application is running in an administrative or local system state. That is, the application has the same rights as any administrative user on the system.

(Mouse click)

When our malicious user here compromises the application, because that application is running in an administrative state, the malicious user can now use the application to perform malicious actions, such as changing system passwords, reading users’ files, and accessing any data on that system. In fact, because the malicious user is essentially an administrator on that system through the compromised application, the user can do whatever is desired.

(Mouse click)

Now see what happens when a malicious user compromises the same application, but this time the application is running using the least privilege principle. That is, it is running in a lower privileged state, such as a network service.

(Mouse click)

Now when the malicious user compromises the application, the malicious user cannot perform malicious actions, such as changing system passwords, reading user files, etc., because the application that the malicious user is using to perform those nefarious actions does not have the privileges (i.e., access) to do so. In applying the least privilege principle, we have greatly limited the potential damage a malicious user can apply to a compromised system. It is still not an ideal situation; we would rather the malicious user not be able to compromise the application at all, but if the malicious user does compromise the application, then at least we can limit the amount of damage that may be inflicted.

### Slide 24 – Least Privilege Tips

Here are a few tips when using least privilege to design applications that are to be more resilient to malicious attack.

Think minimally. Ask yourself, what is the minimum access your application needs to function correctly?

If your application requires higher privileges, elevate those privileges only when required and release those privileges immediately after the purposes of those privileges have been satisfied.

### Slide 25 – SDL Core Principle: Secure Defaults

In the previous section an overview of the principle of least privilege was provided. In this section, another important and last principle known as, “secure defaults” will be presented. Recall that with the attack surface reduction principle, any non-critical part of an application that was exposed to a human or system was removed or disabled by default to reduce the number of exposed vulnerabilities a malicious user could use to compromise an application. The secure defaults principle considers the situation where part of an application needs to be exposed to a human or system by default and how this may be conducted more safely and securely.

Microsoft, through the SDL process, has used this principle to better ensure that customers have safer experiences with our applications out-of-the-box, rather than after extensive and often manual configuration activities must be performed. With this principle, it is left to the user to reduce the security and privacy of an application, and not left to Microsoft / the manufacturer of the software.

Malicious users commonly scan networks for applications or devices that are known to be insecure by default, such as wireless routers and web servers. These applications are easy to compromise. With secure defaults, this ability is taken away from malicious users and helps keeps your customers safer.

### Slide 26 – Secure Defaults Example

Regarding the secure defaults principle, designers need to evaluate the various parts of their application from the perspective of what is the most secure or privacy-aware manner in which this part may be configured.

Here are some examples:

* **Firewall.** Microsoft Windows can be configured with the firewall on or off. By default, the latest and future versions of Windows come with the firewall turned on by default.
* **SSL Socket.** If your application can read data through an SSL socket, then by default it should be configured to use only the latest secure protocol versions, such as v3, TLS, etc., and avoid insecure versions, such as v2.
* **User Access Over Anonymous or Authenticated Channels.** If your application has the option of allowing users to access it over anonymous or authenticated channels, then by default it should use authenticated.
* **Password Complexity.**  If your application can require users to have complex passwords, then that feature should be enabled by default.
* **Storing User Passwords as Hashes.** If your application can store user passwords as hashes or clear text, then it should store passwords as hashes by default.

### Slide 27 – Conclusion

This concludes the discussion on the SDL Secure Design Principles. In this presentation we completed a high-level overview of the SDL and the important role it fulfills in the design stage of an application’s software development lifecycle. We noted that when security and privacy considerations are sufficiently and effectively incorporated early into an application’s software development lifecycle, such as in the design phase, the overall number of threats an application is exposed to and the number of vulnerabilities an application may contains will be substantially reduced. Additionally, the overall cost of maintaining trusted applications will be reduced due to the number of remediation efforts required to address post-deployment security and privacy issues will most likely be reduced.

Finally, we explored the core secure design principles leveraged by the SDL, which are:

* **Attack Surface Reduction.** This principle emphasizes the importance of reducing the overall number of possible points in an application that malicious users can use to attack that application.
* **Basic Privacy.** This principle concentrates on the importance of fulfilling certain legal obligations, increasing customer trust and unblocking deployments based on an application’s behavior.
* **Threat Modeling.** This technique gives application designers a structured and methodical way of understanding and analyzing threats to an application.
* **Defense in Depth.** This principle focuses on how the use of multiple layers of defense for an application greatly reduces the likelihood that a malicious user will be able to exploit it.
* **Least Privilege.** This principle emphasizes the importance of limiting the amount of damage a malicious user can inflict in the event an application is compromised.
* **Secure Defaults.** Finally, this principle concentrates on better ensuring customers’ safe experiences with an application out-of-the-box rather than being required to perform an extensive series of custom configurations.

Lastly, the insights gleaned by Microsoft, which are incorporated in its SDL, and more specifically, in this presentation which focused on Secure Design Principles, have been shared with each of you as a way for our organization to enhance our application development practices and the security of our applications.

### SLIDE 28 - Appendix

This section provides additional slides, materials and information to supplement the main contents of the presentation.

### Slide 29 – Microsoft Security Development Lifecycle (SDL)

This diagram compares the security engineering steps of the SDL to the software engineering steps of the classic SDLC (software development lifecycle). The blue outer ring represents traditional software development and the orange inner circle represents the SDL. Notice that the security engineering steps are incorporated into the existing software engineering steps and that any engineering task can be supplemented with a security engineering task.

Both of these development lifecycles, or collections of engineering steps, apply to the software development lifecycle regardless of the particular development model you use (for example waterfall, Agile, etc.) The small pewter colored circles represent the various milestones in your model and are an excellent time for ensuring that the steps in both the security and software development lifecycles have been adequately addressed.

The SDL process has been documented and published in *The Security Development Lifecycle* book (Microsoft Press 2006, ISBN: 9780735622142), and the official Web site can be accessed at [http://www.microsoft.com/sdl](http://go.microsoft.com/?linkid=9672761).

### Slide 30 – Threat Modeling Resources

This slide provides additional information and links if you would like to learn more about Microsoft’s threat modeling process.

### Slide 31 – Microsoft Developer Network (MSDN) Security Developer Center

Microsoft also has a security developer center located at [http://msdn.microsoft.com/security](http://go.microsoft.com/?linkid=9672763) where developers can find a wealth of resources, including guidance and tools, to help them build safer applications using Microsoft technologies and platforms.

### Slide 32 – Secure Development Blogs

Visit the [SDL Blog](http://go.microsoft.com/?linkid=9672765) to get the most current ideas and thoughts from Microsoft SDL team members.

Visit [Michael Howard’s Blog](http://go.microsoft.com/?linkid=9672764) to read all about how security can be effectively incorporated into the software development process from the author of the popular book, *Writing Secure Code* (Howard, Michael and David LeBlanc, Microsoft Press, Redmond, Washington, 2003).