MS_logo_KMICROSOFT SDL - DEVELOPER STARTER KIT:

THREAT MODELING PRINCIPLES (LEVEL 100)

Guide

Version 1.0

The following documentation provides presenter’s notes for the Microsoft Security Development Lifecycle (SDL) Threat Modeling 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 3](#_Toc233633891)

[1.1 Introduction 3](#_Toc233633892)

[1.2 System Requirements 3](#_Toc233633893)

[1.3 Presentation Themes 3](#_Toc233633894)

[2.0 Microsoft SDL Threat Modeling 3](#_Toc233633895)

[Overview 3](#_Toc233633896)

[Presentation Transcript 3](#_Toc233633897)

[Presentation Voiceover 4](#_Toc233633898)

[Slide 2 – Title Slide 4](#_Toc233633899)

[Slide 3 – Agenda 4](#_Toc233633900)

[Slide 4 – Microsoft Security Development Lifecycle (SDL) 4](#_Toc233633901)

[Slide 5 – Microsoft SDL Threat Modeling Overview 5](#_Toc233633902)

[Slide 6 – Microsoft SDL Threat Modeling Illustrated 6](#_Toc233633903)

[Slide 7 – When to Threat Model 6](#_Toc233633904)

[Slide 8 – Who Performs/Drives Threat Modeling? 6](#_Toc233633905)

[Slide 9 – What to Threat Model 7](#_Toc233633906)

[Slide 10 – Top Microsoft SDL Threat Modeling Advantages and Disadvantages 7](#_Toc233633907)

[Slide 11 – The Microsoft SDL Threat Modeling Process 8](#_Toc233633908)

[Slide 12 – Step 1: Diagramming 9](#_Toc233633909)

[Slide 13 – Data Flow Diagrams (DFDs) Elements 9](#_Toc233633910)

[Slide 14 – Additional Element: Trust Boundaries 9](#_Toc233633911)

[Slide 15 – Step 2: Threat Enumeration 9](#_Toc233633912)

[Slide 16 – STRIDE Threat Types 10](#_Toc233633913)

[Slide 17 – Identifying STRIDE Threats by Data Flow Diagram Element Type 10](#_Toc233633914)

[Slide 18 – Step 3: Mitigation 11](#_Toc233633915)

[Slide 19 – Examples of Standard Mitigations 11](#_Toc233633916)

[Slide 20 – Step 4: Validation 12](#_Toc233633917)

[Slide 21 – The Microsoft SDL Threat Modeling Tool 12](#_Toc233633918)

[Slide 22 – Microsoft SDL Threat Modeling Requirements 12](#_Toc233633919)

[Slide 23 – Conclusion 13](#_Toc233633920)

[Slide 24 - Appendix 14](#_Toc233633921)

[Slide 25 – Microsoft Security Development Lifecycle (SDL) 14](#_Toc233633922)

[Slide 26 – Microsoft Writing Secure Code Book Series 14](#_Toc233633923)

[Slide 27 – Microsoft Developer Network (MSDN) Security Developer Center 14](#_Toc233633924)

[Slide 28 – Secure Development Blogs 14](#_Toc233633925)

[Slide 29 – Hunting Security Bugs 14](#_Toc233633926)

[Slide 30 – Additional SDL Training 15](#_Toc233633927)

[3.0 The Microsoft Security Development Lifecycle (SDL): Training and Resources 15](#_Toc233633928)

[Microsoft SDL Process Guidance 15](#_Toc233633929)

[Microsoft SDL 15](#_Toc233633930)

[Threat Modeling 16](#_Toc233633931)

[Secure Application Development 16](#_Toc233633932)

[SDL for Online Services and Agile 16](#_Toc233633933)

[Microsoft Privacy Guidelines 16](#_Toc233633934)

[4.0 References 16](#_Toc233633935)

# 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:

* Microsoft SDL Threat Modeling Principles (Level 100) presentation.

## 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 Microsoft SDL Threat Modeling

## Overview

Since its inception in 2004, the Microsoft SDL has enabled Microsoft to develop and deliver safer and more trusted applications to customers. One of the core pieces of the Microsoft SDL is the Microsoft SDL Threat Modeling process.

The Microsoft SDL Threat Modeling process is a process that enables application development teams to understand security threats to a system, determine risks from those threats, and establish appropriate mitigations. When performed correctly, this process enables application development teams to deliver safer and more trustworthy applications to customers with higher efficiency and confidence that known classes of security vulnerabilities are understood and effectively addressed.

This is a level 100 presentation meant to familiarize you with Microsoft SDL Threat Modeling fundamentals and principles.

The insights gleaned by Microsoft, which are incorporated in its SDL, and more specifically, in this presentation focusing on Microsoft SDL Threat Modeling 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 provides a transcript for each slide contained in the Microsoft SDL Threat Modeling Principles (Level 100) presentation. The precise transcript text provided herein is also incorporated into the notes section of each slide in the Microsoft SDL Threat Modeling Principles (Level 100) presentation for ease of reference.

## Presentation Voiceover

A voiceover of the Microsoft SDL Threat Modeling Principles (Level 100) presentation transcript below, approximately 40 minutes in length, is also available to assist the presenter in becoming sufficiently acclimated with the subject matter addressed in the Microsoft SDL Threat Modeling 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 Microsoft SDL Threat Modeling Principles (Level 100) presentation introduces the role that the Microsoft Security Development Lifecycle (SDL) fulfills in trusted application development. It also provides an overview of the Microsoft SDL Threat Modeling process which is a core requirement for all applications developed with the Microsoft SDL.

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 Microsoft SDL Threat Modeling fundamentals and principles. These fundamentals and principles will be built upon in subsequent SDL presentations.

### Slide 3 – Agenda

In this presentation, we will complete an overview of the Microsoft SDL, as well as an overview of the Microsoft SDL Threat Modeling process. The steps required to threat model an application using this process will be covered, including the freely available threat modeling tool and the specific Microsoft SDL threat modeling requirements.

### 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 vulnerabilities with certain severity are addressed and resolve before the software is officially released.

**Design phase:** Eradicating coding issues 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 Microsoft SDL process.

As was previously indicated, this presentation focuses on the Microsoft SDL Threat Modeling process and how it can be used to uncover application threats early in the software development lifecycle (SDLC). With respect to specific phases of the Microsoft SDL, this presentation focuses on the Design, Implementation and Verification phases.

### Slide 5 – Microsoft SDL Threat Modeling Overview

The Microsoft SDL Threat Modeling process is a process to identify security threats to a system and to establish appropriate mitigations. When performed correctly, this process enables application development teams to deliver safer and more trustworthy applications to customers with higher efficiency and confidence that known classes of security vulnerabilities are understood and effectively addressed.

Any application developed using the Microsoft SDL must have threat models completed for all at-risk features and functionality. More about the specific Microsoft SDL threat modeling requirements will be discussed later in this presentation, but for now it is important to note that threat modeling is a key requirement of the Microsoft SDL.

The Microsoft SDL Threat Modeling process is broken into four major steps: Diagramming, Threat Enumeration, Mitigation and Validation. In the first step, Diagramming, the application being threat modeled is expressed as a data flow diagram (DFD) to drive the overall risk analysis process. Then during the Threat Enumeration step, threats to the modeled system are identified. After threats have been identified, mitigations to those threats are selected during the Mitigation step. Finally, during the Validation step, threat models are validated for completeness and accuracy.

One of the key benefits of the Microsoft SDL Threat Modeling process over other threat modeling processes is that it can be performed by both security and non-security experts. This property of the Microsoft SDL Threat Modeling process is especially beneficial in scenarios where in-house security expertise may not be available or when hiring outside expertise is not feasible. More aspects regarding this particular benefit and others will be discussed later in the presentation.

As we go through the Microsoft SDL Threat Modeling process, you are encouraged to think about your own applications that you are developing, various components of that application and how those components can be threat modeled using this process.

It is important to note that several other threat modeling processes exist. The process discussed in this presentation is the one used internally at Microsoft. No claims are being made as to which process is the best; however, the process presented in this presentation has been very useful in helping Microsoft deliver safer and most trustworthy applications to its customers.

Lastly, the insights gleaned by Microsoft, which are incorporated in its SDL, and more specifically, in this presentation focusing on Microsoft SDL Threat Modeling 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 – Microsoft SDL Threat Modeling Illustrated

What does threat modeling an application using the Microsoft SDL Threat Modeling process look like?

(Mouse click)

First, an application begins with a design that addresses a problem or fulfills some business scenario. Users may enter data into the application, and that data may be processed by the application in some way.

(Mouse click)

With the Microsoft SDL Threat Modeling process, application designs are expressed or modeled as a set of data flow diagrams (DFDs). Data flow diagrams provide a standard graphical representation of the flow of data within a system.

(Mouse click)

The next step in the Microsoft SDL Threat Modeling process is to identify threats to the application design, as shown with the red boxes. Can a malicious user spoof a legitimate user? Can a malicious user elevate their privileges across any elements? The Microsoft SDL Threat Modeling process uses what is known as the STRIDE approach to answer these and other questions, and will be discussed in more depth later.

After threats are identified, mitigations are selected for each of the threats, as shown by the green checkmark.

### Slide 7 – When to Threat Model

Threat modeling yields the greatest value to application development teams when it is performed during the design phase of the software development lifecycle (SDLC). At the design phase, application designers have the greatest flexibility to make changes to an application to address threats. In Microsoft’s experience, it is much easier to modify an application design than it is to work backwards and modify an application that has been already implemented in code to address discovered threats.

In addition to being easier to make application changes during the design phase, it has also been Microsoft’s experience that it is less costly. Depending on the significance of the change, application changes done during the design phase may require some resources from the application design team. However, no significant developer, tester or security tester resources are required, because at this stage no application code has been implemented and therefore no application code needs to be modified.

Contrast the above to the cost of re-engineering an application later in the software development lifecycle after an application has been implemented in code. Re-engineering implemented applications requires designer resources to approve application changes, developer resources to complete the code changes and tester resources to ensure that functionality of the application has not been compromised after those changes. Security testing resources may also be required to ensure that new vulnerabilities have not been introduced due to the code changes.

### Slide 8 – Who Performs/Drives Threat Modeling?

As discussed in the previous slide, threat modeling is best performed in the design phase of an application’s software development lifecycle (SDLC). At this phase, the greatest flexibility to make application changes is available and at the lowest cost. The next question you may have is who in the application development team is responsible for performing threat modeling? Should security experts be responsible for performing threat modeling, or can non-security experts still participate in this process?

Security experts will be able to threat model an application faster and more effectively than non-security experts, however, it may not be realistic to assume that security expertise will always be available to application development teams. In response to this, the Microsoft SDL Threat Modeling process was designed so that it can be performed by both security experts and non-security experts. So the answer to the original question is, “Anyone within an application team that is familiar with the design of an application could use the Microsoft SDL Threat Model process in order to conduct necessary threat modeling activities.” Furthermore, non-security experts can use this process and still arrive at the proper results. More about how this is achieved is discussed later in this presentation.

Threat models should be updated and reviewed whenever a change to an application’s functionality and feature set is made. Application designers (or sometimes referred to as program managers) are uniquely positioned to have visibility to these types of changes. It is for this reason that it is advisable for application designers to be ultimately responsible for the threat modeling process. With that said, developers and testers should still be part of the threat modeling process. Developers and testers are most familiar with the implementation of an application; they are able to provide highly contextual input on mitigations, and they can help ensure that mitigations identified through the threat modeling process are properly reflected in the application code implementation.

### Slide 9 – What to Threat Model

Thus far, we have established when to perform threat models and who should perform threat modeling. The final question that needs answering is, “What should be threat modeled?”

The application as a whole should be threat modeled since the potential for attack is present throughout the entire application. Individual threat models can be created for certain components of an application; however, the overall threat modeling process should account for the entire application.

Security and privacy features in particular should be threat modeled. These are the controls present in an application that help reduce the risk from attack. Therefore, security and privacy features should be threat modeled to ensure that they are properly designed to be resilient to current attacks. Examples might include, but are not limited to, authentication and authorization subsystems.

Any application feature whose failure has security or privacy implications should be threat modeled. Examples of such a feature may be code that is responsible for loading the appropriate user profile after a user has been authenticated. If code such as this fails to load the correct user profile under certain conditions, this could facilitate what is known as an information disclosure attack. If personal information is exposed in this attack, then the attack could also have privacy implications.

Finally, application features that cross trust boundaries should be threat modeled. Trust boundaries will be discussed in more detail shortly, but briefly trust boundaries are points within an application where data flows from one privilege level to another.

### Slide 10 – Top Microsoft SDL Threat Modeling Advantages and Disadvantages

Let’s now take some time to look at some of the top advantages and disadvantages of the Microsoft SDL Threat Modeling process. After this slide, we will dive into the actual process and see how it can be used to model an application, identify threats and select corresponding mitigations.

(Mouse click)

The first advantage is that this process can be used to find threats to an application design early in the software development lifecycle (SDLC). As discussed earlier, identifying threats and addressing them with mitigations early in the software development lifecycle is much easier and less costly than if those threats are addressed later.

(Mouse click)

Another advantage of the Microsoft SDL Threat Modeling process is that it can be used by both security and non-security experts. Application development teams may not have in-house security expertise or it may not be feasible to hire outside security expertise. In these scenarios, the ability for non-security experts to use the Microsoft SDL Threat Modeling process and still arrive at a baseline set of results is highly beneficial.

(Mouse click)

Microsoft SDL threat models provide insight into the anticipated threats to an application design for application development teams. Designers can take this understanding and design applications that address those threats. Developers can then take those more attack-resilient designs and develop safer and more trustworthy applications. Security testers can also benefit from threat models. With the insight provided by threat models, security testers can focus their efforts on areas of the application that have high potential for attack rather than on those with lower potential for attack. The types of anticipated attacks are also revealed during the threat modeling process and are useful in guiding other security assessment efforts.

(Mouse click)

The key disadvantage to any threat modeling process is the upfront costs required to successful employ the threat modeling process. In order to successfully employ and sustain a threat modeling process within an application development team, resources to train personnel, deploy software and setup the correct internal processes are required. Contrast this to other security assessment techniques and approaches, such as the use of code analysis tools, which, in most cases, can be immediately deployed against an existing code base to identify code vulnerabilities.

### Slide 11 – The Microsoft SDL Threat Modeling Process

Let’s now focus the discussion on the individual steps of the Microsoft SDL Threat Modeling process.

(Mouse click)

Every application begins with a vision. A vision captures the set of business objectives that an application is trying to address. Scenarios where the application will be used and use cases that describe how the application will be used are documented. Security features may be added to those scenarios and use cases, and corresponding security assurances and guarantees may be communicated to customers.

(Mouse click)

When application designs become available, the Microsoft SDL Threat Modeling process begins. In the first step of the Microsoft SDL Threat Modeling process, application designs are modeled or expressed as data flow diagrams (DFDs). Data flow diagrams provide a standard graphical representation of the flow of data within a system. In the next step, threats are enumerated by analyzing those data flow diagrams. Mitigations for identified threats are then selected. Finally, the threat models are validated for completeness and correctness.

Whenever an application’s design or code implementation is changed, the Microsoft SDL Threat Modeling process begins again to reflect the new state of the application and to identify any new threats that may have emerged from those changes. As you can see, the Microsoft SDL Threat Modeling process is an iterative process that is performed in conjunction with the standard software development lifecycle (SDLC).

Let’s now take a closer look at each of the 4 steps of the Microsoft SDL Threat Modeling process.

### Slide 12 – Step 1: Diagramming

The first step of the Microsoft SDL Threat Modeling process is the Diagramming step. The objective of this step is to take an application design and model it as a data flow diagram (DFD).

There are several reasons why the Microsoft SDL Threat Modeling process uses data flow diagrams. The first reason is that data flow diagrams are a standard way to graphically represent data as it flows throughout an application. It is widely used and easily understood by most application development teams. This enables the Microsoft SDL Threat Modeling process to be used by a broader set of application development teams, and not just those using Microsoft technologies. The other reason for using data flow diagrams is that most application attacks are based on data flowing throughout a system. Data flow diagrams provide an excellent and natural way to model this broad characteristic of attacks.

In addition to the standard data flow diagram element set, the Microsoft SDL Threat Modeling process introduces the notion of trust boundaries, which are used to represent data as it flows from one privilege level to another.

### Slide 13 – Data Flow Diagrams (DFDs) Elements

The Microsoft SDL Threat Modeling process uses a standard set of data flow diagram elements.

The first element used in the Microsoft SDL Threat Modeling process is the *external entity* element. The external entity element is drawn as a rectangular box and represents aspects not within the control of an application. For example, users, Web sites used by the application and other systems are examples of external entities.

The next element is the *process* element. This element represents an application or code within an application, such as native code executables and .NET assemblies.

The third data flow diagram element used in the Microsoft SDL Threat Modeling process is the *data store* element. This element is used to represent data at rest, such as data stored in registry keys and databases.

The forth data flow diagram element used in the Microsoft SDL Threat Modeling process is the *data flow* element. This element is represented by a directed arrow and is used to describe how data flows from element to element.

### Slide 14 – Additional Element: Trust Boundaries

The Microsoft SDL Threat Modeling process uses *trust boundaries*, which are represented by a dotted line, but are not part of standard DFDs.

Trust boundaries are points within an application where data flows from one privilege level to another, such as network sockets, external entities and processes with different trust levels. It is critical to document trust boundaries in threat models because they indicate areas where threats can manifest, and often indicate areas within an application that must be analyzed further.

### Slide 15 – Step 2: Threat Enumeration

The next step in the Microsoft SDL Threat Modeling process is to identify threats for each data flow diagram element in the threat model.

Security experts can complete this step through performing brainstorming sessions and employing other informal methods. These informal methods may not be desirable since they lack objectivity and repeatability. A particular security expert, for example, may be more proficient at identifying certain types of threats versus others. Or, fatigue from long brainstorming sessions may skew identification efforts.

The alternative to brainstorming and other informal methods is to use the STRIDE approach. The STRIDE approach can be used by both security experts and non-security experts. This approach first documents the desirable security properties that an application must have and then documents the threats that could compromise those desired properties. The acronym STRIDE is used to represent the threats of spoofing, tampering, repudiation, information disclosure, denial of service and elevation of privilege. The original STRIDE threat set was derived from analyzing the issues encountered by the Microsoft Security Response Center (MSRC) and the Common Vulnerability and Exposures (CVE) list maintained at <http://cve.mitre.org>.

### Slide 16 – STRIDE Threat Types

As discussed in the previous slides, the STRIDE threat types document the desired application properties, as shown in the first column in the table shown here, such as authentication and availability. Then the threats that could compromise those desired properties are listed, as shown in the second column.

**Spoofing:** The first threat in the STRIDE acronym is spoofing threats. These threats allow a malicious user to pose as something or someone else, such as a legitimate user of an application or an external service. The desired property that is negatively affected by a spoofing threat is the authentication property, which enables an application to validate the identity of a principal.

**Tampering:** The letter “T” in the STRIDE acronym represents tampering threats. Tampering threats allow a malicious user to make unauthorized modifications to data or code. Both data that is at rest or in transit, such as data sent across the Internet, can potentially be manipulated. In the case of tampering, the desired application property of integrity is affected.

**Repudiation:** Whenever a malicious user is able to perform a malicious action against an application and that action cannot be traced or associated back to that malicious user, repudiation threats emerge. An online ecommerce application that cannot provide evidence that a customer has received a particular shipment even if that shipment was indeed received by the customer is exposed to repudiation threats. The opposite and desired property to repudiation is non-repudiation.

**Information Disclosure:** The exposure of information to users who are not authorized or intended to have access to that information constitutes an information disclosure threat. A malicious user who is able to read another user’s profile without granted authorization is an example of an information disclosure threat. The desired application property that is compromised in the case of information disclosure threats is the confidentiality property.

**Denial of Service (DoS):** Applications need to be available to legitimate users, especially in the case of ecommerce applications. The desired property is therefore availability. The threat that could negatively affect an application’s availability to legitimate users is the denial of service threat. Denial of service threats enable malicious users to deny or degrade a service to legitimate users.

**Elevation of Privilege (EoP):** The last threat in the STRIDE acronym is elevation of privilege threats. These threats are created whenever a malicious user is able to transition from one privilege level to another without proper authorization. For example, a malicious user from the Internet (i.e., an anonymous user) who is able to elevate their privilege level by compromising an application running as SYSTEM is an example of an elevation of privilege threat. The desired application property that is affected by elevation of privilege threats is the authorization property.

### Slide 17 – Identifying STRIDE Threats by Data Flow Diagram Element Type

As mentioned earlier, the Microsoft SDL Threat Modeling process can be used by non-security experts to enumerate potential threats to an application. The chart shown here shows the common STRIDE threat types by data flow diagram element types and can be used as a baseline for enumerating threats, especially for non-security experts.

To use this chart, each data flow diagram (DFD) element in the application models is analyzed for appropriate threats.

(Mouse click)

For instance, if the data flow diagram of the modeled application contains an external entity element, then the threats of spoofing and repudiation must be considered.

(Mouse click)

Alternatively, if the data flow diagram of the modeled application contains a process element, then all STRIDE threats must be considered.

You may notice that the check mark in the data store/repudiation store is red. This color coating is used to indicate that data stores are sometimes affected by repudiation threats. Data stores are affected by repudiation threats whenever the data stores themselves are a log.

### Slide 18 – Step 3: Mitigation

After threats have been enumerated, the next step in the Microsoft SDL Threat Modeling process is to select mitigations to address those threats. In order of preference, the four ways to address threats in the Microsoft SDL Threat Modeling process are outlined below:

(Mouse click)

The first approach is to redesign the application to eliminate identified threats. This is typically the preferred mitigation path; however, there may be situations where an application cannot be further redesigned without removing vital and valid business functionality. In this situation other mitigation options may be more suitable.

(Mouse click)

The next mitigation approach is to use standard mitigations. These types of mitigations are well understood and are a preferred mitigation path whenever redesign is not possible. Examples of standard mitigations are using access control lists to protect registry keys, using SSL to protect network connections and using the principle of least privilege to isolate elevation of privilege attacks. Examples of standard mitigations will be shown in the next slide.

(Mouse click)

If standard mitigations cannot be applied, then the use of a unique or custom mitigation may be more appropriate. A unique mitigation is often difficult, time consuming and risky to implement. Implementing a unique mitigation properly requires deep expertise in the particular technology the mitigation is being developed for, such as databases and network protocols. A unique mitigation may appear to provide sufficient controls when in fact it is not; therefore, it is recommended that you work with security experts when implementing these types of mitigations.

(Mouse click)

Finally, the last approach to mitigating threats is to accept the risk in accordance with policies. In some situations, redesigning an application, using standard mitigations or using a unique mitigation may not be practical. For instance, it may be less expensive to accept an application risk than it is to redesign the application or implement a standard mitigation. Or there may be certain assumptions or external dependencies that an application makes that mitigate certain threats that may need to be communicated to users.

At Microsoft, it is company policy not to ship any applications with moderate, important or critical vulnerabilities unless formally documented senior executive approval is received.

### Slide 19 – Examples of Standard Mitigations

Here are examples of standard mitigations by STRIDE threat types that are provided to help you consider how to mitigate threats. For example, a standard mitigation to address spoofing threats may be to use IPsec, digital signatures or message authentication codes.

Chapter 9 of the Microsoft SDL book provides a more detailed chart of recommended standard mitigations, as prescribed by the Microsoft SDL. Refer to [http://www.microsoft.com/mspress/books/8753.aspx](http://go.microsoft.com/?linkid=9672766) for more information.

### Slide 20 – Step 4: Validation

The last step of the Microsoft SDL Threat Modeling process is the Validation step. The objective of this step is to help ensure that threat models accurately reflect application design and potential threats.

Models should be validated to ensure that they accurately and sufficiently reflect the actual application design; otherwise, inaccuracies or insufficiencies could result in some threats being overlooked. Some model validation guidelines to follow include:

1. Ensuring that your data flow diagram contains at least one trust boundary;
2. Ensuring that all data flow paths are properly represented; and
3. Ensuring that all data sources are represented in the model.

Enumerated threats should also be validated for completeness. Each data flow diagram element, for instance, should have the appropriate set of threats enumerated. The chart that maps STRIDE threat types against data flow diagram elements is a good baseline to use to ensure that threats have been sufficiently enumerated.

Mitigations should be labeled clearly if they are a standard mitigation or a unique mitigation. Mitigations should also have a tracking item open in tracking databases so that the status of that mitigation can be tracked.

Finally, any assumptions or dependencies noted during the threat modeling process should have at least one corresponding test case to validate that assumption. If an application depends on an external system, then the owners of that system should be conferred with to validate those dependencies and any related assumptions.

### Slide 21 – The Microsoft SDL Threat Modeling Tool

Microsoft has published a tool that is freely available for download called the Microsoft SDL Threat Modeling Tool. This tool helps application designers model their applications for threats, as well as identify and manage corresponding mitigations.

Refer to <http://msdn.microsoft.com/en-us/security/dd206731.aspx> for more information regarding this tool, such as links to download the current version of the tool, online videos and tutorials.

### Slide 22 – Microsoft SDL Threat Modeling Requirements

As we have seen in this presentation, threat modeling can help application development teams develop safer and more trustworthy applications. Applications that have been properly threat modeled are likely to be more resilient to malicious user attacks because security concerns have been considered and mitigated early in the design phase of the software development lifecycle (SDLC) rather than later after application code has been implemented. Since the inception of the Microsoft SDL in 2004, Microsoft has required that all product teams create threat models, and that all threat models must meet the following requirements.

(Mouse click)

The first requirement is that all functionality identified during the cost analysis phase (stage 2) of the Microsoft SDL must have corresponding threat models. Typically threat models must be considered for all at-risk code, all code written or licensed from a third party, all features and functionality in new products, and all features and functionality of updated versions of existing products.

(Mouse click)

The second requirement is that all threat models created must meet minimal quality requirements. All threat models must contain data flow diagrams (DFDs), assets represented by data flow diagram elements, enumerated threats and mitigations. More information regarding the minimal quality requirements for threat models can be found in chapter 9 of the Microsoft SDL book (see [http://www.microsoft.com/mspress/books/8753.aspx](http://go.microsoft.com/?linkid=9672766)).

(Mouse click)

The third requirement is that all threat models and reference mitigations must be reviewed and approved by at least one developer, one tester and one program manager. The review and approval should be performed by architects, developers, testers, program managers and others who are intimately familiar with the application being threat modeled. Not only does this help establish accountability for the threat models produced, but this also helps to further ensure that threat models are comprehensive and accurate.

(Mouse click)

The final requirement is that threat models and any associated documentation, such as functional and design specifications, must be stored using document control systems. Threat modeling captures a wealth of data regarding the threats and mitigations associated with an application. It is therefore important to document and preserve that data and any lessons learned for future iterations of the application.

### Slide 23 – Conclusion

This concludes the discussion on the Microsoft SDL Threat Modeling process. This process enables application development teams to identify security threats to a system, determine risks and establish appropriate mitigations.

The Microsoft SDL Threat Modeling process yields the greatest benefits to application development teams when it is performed early in the software development lifecycle (SDLC). Specifically, the design phase is the ideal point within the software development lifecycle to start threat modeling. At this stage of the software development lifecycle, any changes to an application can be done easily and with the lowest cost as compared to performing changes later in the software development lifecycle, such as during the implementation and verification stages. Additional benefits of the Microsoft SDL Threat Modeling process include the ability to use the results from threat models as a guide to focus security verification efforts. Security testers can quickly look at threat models and understand the components of an application that are most at risk based upon the types of threats present. Another advantage of the Microsoft SDL Threat Modeling process is that it can be used by non-security experts. Application development teams may not have in-house security expertise available, and Microsoft has designed its threat modeling process in such a way where the process can deliver proper results when employed by non-security experts.

The Microsoft SDL Threat Modeling process consists of four steps. The first is the Diagramming step where application designs are modeled as data flow diagrams (DFDs). The next step involves enumerating threats against the data flow diagrams. To do this, the Microsoft SDL Threat Modeling process leverages the STRIDE model and maps different STRIDE threat types back to specific data flow diagram elements. By doing this, the overall threat modeling process is more objective, repeatable, and can still be used by non-security experts. After threats are identified, responses to those threats are chosen. The four basic approaches to addressing threats using the Microsoft SDL Threat Modeling process in order of preference is to redesign to eliminate the threat, use standard mitigations, use custom mitigations and finally accept the threat according to company policies. The final step in the Microsoft SDL Threat Modeling process is to validate the threat models to better ensure that threat models are complete and comprehensive.

To aid application developer teams with threat modeling efforts, Microsoft has published a tool called the Microsoft Threat Modeling Tool. This tool is freely available for download and can help application designers model their applications for threats, as well as identify and manage corresponding mitigations.

Threat modeling is a required activity within the Microsoft SDL. Since the inception of the Microsoft SDL in 2004, Microsoft has required that all product teams create threat models in order to continue to deliver safer and more trustworthy applications to customers.

Lastly, the insights gleaned by Microsoft, which are incorporated in its SDL, and more specifically, in this presentation which focused on Microsoft SDL Threat Modeling 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 24 - Appendix

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

### Slide 25 – 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 26 – Microsoft Writing Secure Code Book Series

Microsoft has several publications on secure implementation including the industry leading Writing Secure Code series. Writing Secure Code is mandatory reading for software engineering teams at Microsoft and provides an in-depth discussion of common software weaknesses and effective remedies.

It also provides information with which testers can use to better ensure that the applications they are testing meet security quality assurance requirements.

### Slide 27 – 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 development teams (architects, developers and testers) can find a wealth of resources, including guidance and tools, to help them build safer applications using Microsoft technologies and platforms.

### Slide 28 – 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).

### Slide 29 – Hunting Security Bugs

Members of the Microsoft Office Security team have written a book that covers common application security issues and how to test for them. More information about this book can be found at [http://www.microsoft.com/mspress/books/8485.aspx](http://go.microsoft.com/?linkid=9672768).

### Slide 30 – Additional SDL Training

Additional SDL training content, such as the following is currently or will be available soon:

**Secure Design Principles:** This content provides application designers with the fundamentals and principles they require to design more secure applications. Other content related to secure design builds upon the knowledge established in this content.

**Secure Implementation Principles:** This content provides developers with the fundamentals and principles they require to develop more secure applications. Other content related to secure implementation builds upon the knowledge established in this content.

**Secure Verification Principles:** This content provides testers and quality assurance personnel with the fundamentals and principles they require to test secure applications. Other content related to secure testing builds upon the knowledge established in this content.

**SQL Injection Vulnerabilities:** SQL injection vulnerabilities are commonly encountered vulnerabilities in applications using a database. As more applications move towards the Web paradigm and are driven by databases, this vulnerability is expected to become even more prolific than is currently being realized. This content provides an overview of SQL injection vulnerabilities and how the SDL can be used to significantly reduce the risk of a SQL injection attack.

**Cross-Site Scripting Vulnerabilities:** Cross-site scripting vulnerabilities are the most commonly encountered Web-based vulnerabilities today. These types of vulnerabilities continue to plague the Web-application world and a user’s ability to trust the applications they are using. This content provides an overview of cross-site scripting vulnerabilities, and how the SDL can be applied to significantly reduce the risk of a cross-site scripting attack.

**Buffer Overflow Vulnerabilities:** Buffer overflows are considered the most dangerous application-level vulnerability. This content provides an overview of buffer overflows, and how the SDL can be used to significantly reduce the risk of a buffer overflow attack.

# 3.0 The Microsoft Security Development Lifecycle (SDL): Training and Resources

## Microsoft SDL Process Guidance

Detailed information on all stages and requirements of version 3.2 of the Microsoft SDL:

* Available on MSDN - [Microsoft SDL version 3.2 – Process Guidance](http://msdn.microsoft.com/en-us/security/cc420639.aspx)
* Available on the Download Center - [Microsoft SDL Version 3.2 Process Guidance](http://go.microsoft.com/?linkid=9672762) (.doc)

## Microsoft SDL

* Michael Howard's "Everything Developer Security" talk, TechEd Barcelona, Nov. 2007 – [Presentation](http://download.microsoft.com/download/5/1/2/5123a603-2077-49c6-8451-16f6979b7afb/Michael%20Howard%20Everything%20Developer%20Security.ppsx) and [Video](http://download.microsoft.com/download/5/1/2/5123a603-2077-49c6-8451-16f6979b7afb/Michael%20Howard%20Everything%20Developer%20Security.wmv)
* Michael Howard discusses threat modeling, TechEd Barcelona, Nov. 2007 – [Presentation](http://download.microsoft.com/download/b/f/9/bf92d114-76f1-4046-a54c-5b3887dd09f7/Michael%20Howard%20Threat%20Modeling.ppsx) and [Video](http://download.microsoft.com/download/b/f/9/bf92d114-76f1-4046-a54c-5b3887dd09f7/Michael%20Howard%20Threat%20Modeling.wmv)
* Security practitioners and experts discuss "A Proactive Approach to Building a Successful Security Development Lifecycle (SDL) Program", Nov. 2008 – [Presentation](http://download.microsoft.com/download/A/3/F/A3FDAE54-9911-4D13-BD22-76D4195A70AF/SDL%20Webcast_Nov08.pptx)
* Quiz: [Test Your Security IQ](http://msdn.microsoft.com/magazine/cc982154.aspx), Nov. 2008– Put your C/C++/C# security skills to the challenge by reviewing ten tricky code snippets devised by Michael Howard and Bryan Sullivan.
* SDL Series: a set of 8 articles investigating the Microsoft Security Development Lifecycle, Oct.2008 - Apr.2009
* Article #1: [Investigating the Microsoft Security Development Lifecycle](http://www.microsoft.com/windowsserver/compare/ReportsDetails.mspx?recid=75)
* Article #2: [Security Education at Microsoft](http://www.microsoft.com/windowsserver/compare/ReportsDetails.mspx?recid=77)
* Article #3: [The Microsoft Security Org Chart](http://www.microsoft.com/windowsserver/compare/ReportsDetails.mspx?recid=80)
* Article #4: [Threat Modeling at Microsoft](http://www.microsoft.com/windowsserver/compare/ReportsDetails.mspx?recid=81)
* Article #5: [Microsoft’s Security Toolbox](http://www.microsoft.com/windowsserver/compare/ReportsDetails.mspx?recid=82)
* Article #6: [Microsoft’s Security Response](http://www.microsoft.com/windowsserver/compare/ReportsDetails.mspx?recid=82)
* The [Security Development Lifecycle Blog](http://go.microsoft.com/?linkid=9672765) pulls together comments and insights from the Security Engineering team at Microsoft.
* Michael Howard and Steve Lipner, [The Security Development Lifecycle](http://go.microsoft.com/?linkid=9672766), Microsoft Press, Redmond, Washington, 2006
* Michael Howard and David LeBlanc, [Writing Secure Code](http://www.microsoft.com/MSPress/books/5957.aspx), Second Edition, Microsoft Press, Redmond, Washington, 2003

## Threat Modeling

The following are resources to help get started with threat modeling:

* [Uncover Security Design Flaws Using the STRIDE Approach](http://msdn.microsoft.com/en-us/magazine/cc163519.aspx), Nov. 2006 – Learn how the STRIDE (Spoofing, Tampering, Repudiation, Information disclosure, Denial of Service, Elevation of privileges) model helps uncover and mitigate security design vulnerabilities.
* Michael Howard discusses threat modeling, TechEd Barcelona, Nov. 2007 – [Presentation](http://download.microsoft.com/download/b/f/9/bf92d114-76f1-4046-a54c-5b3887dd09f7/Michael%20Howard%20Threat%20Modeling.ppsx) and [Video](http://download.microsoft.com/download/b/f/9/bf92d114-76f1-4046-a54c-5b3887dd09f7/Michael%20Howard%20Threat%20Modeling.wmv)
* Michael Howard and Adam Shostack walk you through the SDL Threat Modeling Tool, Nov. 2008 - [Video](http://mfile.akamai.com/14853/wmv/microsofttec.download.akamai.com/14853/TechEdOnline/Videos/452_low.asx)
* [Getting Started with the SDL Threat Modeling Tool](http://msdn.microsoft.com/en-us/magazine/2009.01.securitybriefs.aspx), Jan. 2009 – Follow Deb (a developer), Paul (a program manager), and Tim (a tester) through the process of developing their first threat model.
* [Reinvigorate Your Threat Modeling Process](http://msdn.microsoft.com/en-us/magazine/cc700352.aspx), Jul. 2008 – Learn from Microsoft’s lessons learned and apply them to your organization.
* [Threat Models Improve Your Security Process](http://msdn.microsoft.com/magazine/dd148644.aspx), Nov. 2008 – Learn how to think about secure design from a more holistic perspective by using threat models to drive your security engineering process, primarily helping you prioritize code review, fuzz testing, and attack surface analysis tasks.
* [The Trouble with Threat Modeling](http://blogs.msdn.com/sdl/attachment/7702305.ashx), a series from the SDL Blog
* [Experiences Threat Modeling at Microsoft](http://blogs.msdn.com/sdl/attachment/8991806.ashx) from the Security Modeling Workshop at MODELS 08 SDL
* [All threat modeling posts](http://blogs.msdn.com/sdl/archive/tags/threat+modeling/default.aspx) from the SDL Blog
* [Threat Modeling for Line Of Business (LOB) Applications](http://go.microsoft.com/fwlink?linkid=77002) – the Microsoft ACE approach

## Secure Application Development

* [Securing Applications with the .NET Framework](http://msdn.microsoft.com/library/fkytk30f(VS.71).aspx)  
  The common language runtime and the .NET Framework provide many useful classes and services that enable developers to easily write security code. These classes and services also enable system administrators to customize the access that code has to protected resources. In addition, the runtime and the .NET Framework provide useful classes and services that facilitate the use of cryptography and role-based security.
* [Patterns & Practices – Security Guidance for Application Development](http://msdn.microsoft.com/library/ms998408.aspx)

## 

## SDL for Online Services and Agile

* [SDL embraces Web](http://msdn.microsoft.com/magazine/cc794277.aspx), Sept. 2008 – Get detailed information on the new online service SDL requirements.
* Bryan Sullivan discusses "More Secure Online Services Powered by the Microsoft Security Development Lifecycle", Oct. 2008 – Presentation and [Video](http://www.microsoft.com/events/series/detail/webcastdetails.aspx?seriesid=15&webcastid=5103)
* [Agile SDL: Streamline Security Process for Agile Development](http://msdn.microsoft.com/magazine/dd153756.aspx), Nov. 2008 – Get to know the new SDL/Agile methodology

## 

## Microsoft Privacy Guidelines

* [Privacy guidelines](http://www.microsoft.com/downloads/details.aspx?FamilyID=c48cf80f-6e87-48f5-83ec-a18d1ad2fc1f&displaylang=en) for developing software products and services that are based on Microsoft internal guidelines and experience incorporating privacy into the software development process.

# 4.0 References

Hernan, Shawn et al. “Uncover Security Design Flaws Using the STRIDE Approach,” <http://msdn.microsoft.com/en-us/magazine/cc163519.aspx>. January 2009.

Howard, Michael et al. The Security Development Lifecycle. Redmond, WA: Microsoft Press, 2006.

Microsoft Corporation. “The Microsoft SDL Threat Modeling Tool 3.0,” <http://msdn.microsoft.com/en-us/security/dd206731.aspx>. January 2009.

Shostack, Adam. “Experiences Threat Modeling at Microsoft,” <http://blogs.msdn.com/sdl/attachment/8991806.ashx>. January 2009.

Shostack, Adam. “Reinvigorate your Threat Modeling Process,” <http://msdn.microsoft.com/en-us/magazine/cc700352.aspx>. January 2009.