MS_logo_KMICROSOFT SDL - DEVELOPER STARTER KIT:

THREAT MODELING TOOL PRINCIPLES (LEVEL 100)

Guide

Version 1.0

The following documentation provides presenter’s notes for the Microsoft Security Development Lifecycle (SDL) Threat Modeling Tool 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](#_Toc233640406)

[1.1 Introduction 2](#_Toc233640407)

[1.2 System Requirements 2](#_Toc233640408)

[1.3 Presentation Themes 2](#_Toc233640409)

[2.0 Microsoft SDL Threat Modeling Tool 2](#_Toc233640410)

[Overview 2](#_Toc233640411)

[Presentation Transcript 2](#_Toc233640412)

[Presentation Voiceover 3](#_Toc233640413)

[Presentation Demonstrations 3](#_Toc233640414)

[Slide 2 – Title Slide 3](#_Toc233640415)

[Slide 3 – Agenda 3](#_Toc233640416)

[Slide 4 – Microsoft Security Development Lifecycle (SDL) 3](#_Toc233640417)

[Slide 5 – Microsoft SDL Threat Modeling Tool Overview 4](#_Toc233640418)

[Slide 6 – Microsoft SDL Threat Modeling Tool Benefits and Features 5](#_Toc233640419)

[Slide 7 – Using the Microsoft SDL Threat Modeling Tool 5](#_Toc233640420)

[Slide 8 – Step 1: Draw Diagrams 6](#_Toc233640421)

[Slide 9 – Step 2: Analyze Model 6](#_Toc233640422)

[Slide 10 – Step 3: Describe Environment 7](#_Toc233640423)

[Slide 11 – Step 4: Generate Report 7](#_Toc233640424)

[Slide 12 – Microsoft SDL Threat Modeling Tool Demonstration 7](#_Toc233640425)

[Slide 13 – Microsoft SDL Threat Modeling Tool Discussion Forum 7](#_Toc233640426)

[Slide 14 – Conclusion 7](#_Toc233640427)

[Slide 15 - Appendix 8](#_Toc233640428)

[Slide 16 – Microsoft Security Development Lifecycle (SDL) 8](#_Toc233640429)

[Slide 17 – Microsoft Writing Secure Code Book Series 8](#_Toc233640430)

[Slide 18 – Microsoft Developer Network (MSDN) Security Developer Center 8](#_Toc233640431)

[Slide 19 – Secure Development Blogs 8](#_Toc233640432)

[Slide 20 – Hunting Security Bugs 8](#_Toc233640433)

[Slide 21 – Additional SDL Training 9](#_Toc233640434)

[Slide 22 – STRIDE Threat Types 9](#_Toc233640435)

[3.0 References 10](#_Toc233640436)

# 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 *Threat Modeling Tool 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 Tool

## 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. To assist application development teams to use the Microsoft SDL Threat Modeling process, Microsoft has also published a threat modeling tool.

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

The insights gleaned by Microsoft, which are incorporated in its SDL, and more specifically, in this presentation focusing on the Microsoft SDL Threat Modeling Tool, are provided as a way for external developer communities to enhance their 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 Tool 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 Tool Principles (Level 100) presentation for ease of reference.

## 

## Presentation Voiceover

A voiceover of the Microsoft SDL Threat Modeling Tool Principles (Level 100) presentation transcript below, approximately 29 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 Tool Principles (Level 100) presentation, as well as to better understand the author’s perspective behind each slide in the presentation.

## 

## Presentation Demonstrations

A voiceover of the Threat Modeling Tool Principles (Level 100) presentation transcript below, approximately 29 minutes in length is also available to assist the presenter in becoming sufficiently acclimated with the subject matter addressed in the Threat Modeling Tool 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 Tool 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 Tool, which can be used by application development teams to more effectively and correctly use the Microsoft SDL Threat Modeling process.

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 Tool fundamentals and principles. These fundamentals and principles will be built upon in subsequent SDL presentations. Please also note that it is highly recommended that the Microsoft SDL Threat Modeling Principles (Level 100) presentation be presented prior to this presentation.

### 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 Tool. Benefits and features of the tool will be discussed along with the steps required to use the tool. A demonstration of the Microsoft SDL Threat Modeling Tool will be given to show how an application development team may create their own threat models using this tool. Finally, information regarding the discussion forum, which application development teams can use to discuss the tool and collaborate with other users, will be provided.

### 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 Tool 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 phase.

### Slide 5 – Microsoft SDL Threat Modeling Tool Overview

To help application development teams use the Microsoft SDL Threat Modeling process, which is a process to identify security threats to a system and establish appropriate mitigations, Microsoft has also published a threat modeling tool. The Microsoft SDL Threat Modeling Tool is a graphical user interface (GUI) tool that helps application designers create and analyze threat models. Application designers can also use this tool to document mitigation plans and create actionable reports that can be used by the remaining members of the application development team. Furthermore, with this tool, application teams can help ensure that they are using the Microsoft SDL Threat Modeling process more consistently and with greater efficiency as opposed to using the process in a highly manual fashion.

The official Microsoft SDL Threat Modeling Tool Web page, located at <http://msdn.microsoft.com/en-us/security/dd206731.aspx>, provides extensive information regarding the Microsoft SDL Threat Modeling Tool, such as links to download the current version of the tool, online videos and tutorials.

To use the Microsoft SDL Threat Modeling Tool, users must be running Microsoft Vista, Microsoft Server 2008 or higher. Also, users must have Microsoft Visio 2007 or higher installed. A 60-day trial version is available for download at <http://us20.trymicrosoftoffice.com/product.aspx?re_ms=oo&family=visioprofessional&culture=en-US>.

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

The Microsoft SDL Threat Modeling process is a process application designers can leverage to analyze their designs for potential threats and select mitigations. It is broken down into four major steps: Model, Enumerate Threats, Mitigate and Validate. The Microsoft SDL Threat Modeling Tool is designed to help streamline aspects of the overall threat modeling process and assist users in creating proper threat models.

(Mouse click)

The first benefit of the Microsoft SDL Threat Modeling Tool is process automation. The Microsoft SDL Threat Modeling Tool automates many aspects of the Microsoft SDL Threat Modeling process. Through automation, application teams can better ensure that they are following the process correctly and yielding the greatest results for their threat modeling efforts. Also with automation, larger and more complicated application designs can be threat modeled with greater ease and efficiency than would be possible while employing a manual approach.

(Mouse click)

Another advantage of using the Microsoft SDL Threat Modeling Tool is the guided analysis of potential threats and mitigations. The Microsoft SDL Threat Modeling Tool uses the STRIDE model (refer to slide 22 in the appendix for more information) to automatically generate a baseline of potential threats against data flow diagram (DFD) elements described in the threat model.

(Mouse click)

Application development teams that use Microsoft Visual Studio Team Foundation Server (refer to <http://msdn.microsoft.com/en-us/teamsystem/dd408382.aspx> for more information) will benefit from the Microsoft SDL Threat Modeling Tool’s reporting integration ability. Please refer to the help documentation that accompanies the Microsoft SDL Threat Modeling tool for further information on how the tool can be integrated with Microsoft Visual Studio Team Foundation Server.

(Mouse click)

Finally, the Microsoft SDL Threat Modeling Tool can create actionable reports, which can be used by the rest of the application development team to drive security and testing activities in the Implementation and Verification phase of the Microsoft SDL. For instance, the Microsoft SDL Threat Modeling Tool can analyze a threat model and provide recommendations as to which elements within the model should be fuzz tested.

### Slide 7 – Using the Microsoft SDL Threat Modeling Tool

There are four steps to using the Microsoft SDL Threat Modeling tool.

(Mouse click)

The first step is the *Draw Diagrams* step. In this step, application designers use visual controls to model their application designs as a data flow diagram (DFD). This data flow diagram will then be used later by the Microsoft SDL Threat Modeling Tool to drive the rest of the threat analysis.

(Mouse click)

The next step is the *Analyze Model* step. In this step, the Microsoft SDL Threat Modeling Tool will analyze your data flow diagram and produce a list of potential threats using the STRIDE model (refer to slide 22 in the appendix for more information). The STRIDE model describes common application security threats, such as spoofing, tampering and elevation of privilege, and is based on vulnerabilities observed by the Microsoft Security Response Center (MSRC) and the Common Vulnerabilities and Exposures (<http://cve.mitre.org>) listing. The Microsoft SDL Threat Modeling process uses a mapping between elements within data flow diagrams to threats within the STRIDE model. In this fashion, this tool can also be used by non-security experts whenever security expertise is not available and potential threats can be generated more consistently and objectively.

Another presentation called “Microsoft SDL Threat Modeling Principles (Level 100)” is available. This presentation addresses the Microsoft SDL Threat Modeling process in much more detail than is addressed in this Threat Modeling Tool Principles presentation.

(Mouse click)

The third step to use the Microsoft SDL Threat Modeling Tool is the *Describe Environment* step. This step captures information regarding how the application being threat modeled will be deployed. The information gathered in this step does not affect the threat model analysis results; information gathered captures additional information that may be useful in the overall threat analysis. One example of information that may be gathered could consist of security dependencies that your application may have and assumptions made by your application.

(Mouse click)

The last step is the *Generate Reports* step. In this final step, the Microsoft SDL Threat Modeling Tool will help generate different types of reports, such as analysis reports and recommended testing reports. Reports generated can then be saved as \*.mht (MIME HTML) standard format or may be printed for future use.

Let’s now take a look at the individual steps within the Microsoft SDL Threat Modeling Tool, and then later we will actually create a threat model using the tool.

### Slide 8 – Step 1: Draw Diagrams

The first step in using the Microsoft SDL Threat Modeling Tool, similar to the actual Microsoft SDL Threat Modeling process, is to model an application design as a data flow diagram (DFD).

(Mouse click)

The Microsoft SDL Threat Modeling Tool provides users with a simple drag and drop Visio-based interface that allows application designers to visually model their designs within the tool.

### Slide 9 – Step 2: Analyze Model

After an application design has been modeled as a data flow diagram (DFD), the next step is the Analyze Model step. In this step potential threats against the data flow diagram are identified and mitigations for those threats are documented on a per-data-flow-diagram-element basis.

(Mouse click)

The Microsoft SDL Threat Modeling Tool generates a baseline set of potential threats for users using the STRIDE per element technique shown by the red boxes. Generating threats using STRIDE allows the Microsoft SDL Threat Modeling process and tool to be used by both security experts and non-security experts. Also, using STRIDE allows threats to be generated in a more repeatable and objective fashion as compared to enumerating potential threats using more informal methods. Mitigations, as shown by the green boxes, are also documented in this step.

### Slide 10 – Step 3: Describe Environment

The third step employed by the Microsoft SDL Threat Modeling Tool is the Describe Environment step. This step does not affect the set of baseline threats generated by the Microsoft SDL Threat Modeling Tool, but is useful to document any security dependencies or assumptions. For instance, does your application depend on any libraries that will be linked as part of the overall code generation process? These should be documented. Does your application assume that only an administrator will start the application process? This is another example of an assumption that should be documented. The Microsoft SDL Threat Modeling Tool provides an interface where users can easily enter these deployment environment notes.

### Slide 11 – Step 4: Generate Report

The final step is the Generate Report step. In this step, reports that summarize threats identified, mitigations selected and any other useful analysis details are created for users. Reports can be saved as \*.mht (MIME HTML) files or printed and shared among the application development team members.

### Slide 12 – Microsoft SDL Threat Modeling Tool Demonstration

Let’s now see a demonstration of the Microsoft SDL Threat Modeling Tool and see how easy it is to use the tool to model application designs, identify potential threats to those designs, document mitigations and generate reports.

(Mouse click)

In our demonstration we will threat model the actual use of the threat modeling tool itself.

(Start Microsoft SDL Threat Modeling Tool demonstration)

### Slide 13 – Microsoft SDL Threat Modeling Tool Discussion Forum

While the Microsoft SDL Threat Modeling tool is not a formally supported tool, there is a discussion forum available at the link shown here (<http://social.msdn.microsoft.com/Forums/en-US/sdlthreatmodeling/threads>) to discuss the tool and also report any issues with the tool.

### Slide 14 – Conclusion

This concludes the discussion of the Microsoft SDL Threat Modeling Tool. The Microsoft SDL Threat Modeling Tool is a graphical user interface (GUI) tool that application development teams can use to threat model their designs using the Microsoft SDL Threat Modeling process.

The Microsoft SDL Threat Modeling Tool provides several benefits and features to application development teams. The first is the high degree of automation of the Microsoft SDL Threat Modeling process. By using the Microsoft SDL Threat Modeling Tool, application development teams can better ensure that they are following the overall threat modeling process correctly and yielding the best results possible for their threat modeling efforts. The second is that a baseline of potential threats is created automatically using the STRIDE technique. This feature of the Microsoft SDL Threat Modeling Tool is especially beneficial in scenarios where security expertise may not exist among internal personnel or for whatever other reason may be unavailable. For application development teams using Microsoft Visual Studio Team Foundation Server, the third benefit is that Microsoft SDL Threat Modeling Tool can integrate with existing Team Foundation Server issue-tracking systems. Finally, the Microsoft SDL Threat Modeling Tool can create actionable reports, which can be used by the rest of the application development team to drive security and testing activities in the Implementation and Verification phase of the Microsoft SDL.

There are four basic steps to use the Microsoft SDL Threat Modeling Tool. The first is the Draw Diagram step where application designs are expressed as data flow diagrams (DFDs). The second step is the Analyze Model step where a baseline set of potential threats to the data flow diagram elements is generated by the tool and mitigations to those potential threats are documented. The third step is the Describe Environment step where details regarding the application, such as any dependencies or assumptions, are also documented. Finally, reports that summarize identified threats, mitigation paths and other useful information are generated in the last step called the Generate Reports step.

Currently the Microsoft SDL Threat Modeling tool is not a formally supported tool; however, Microsoft has provided an Internet discussion forum whereby users of the tool can discuss the tool, as well as collaborate with other users.

Lastly, the insights gleaned by Microsoft, which are incorporated in its SDL, and more specifically, in this presentation which focused on the Microsoft SDL Threat Modeling Tool, 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 15 - Appendix

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

### Slide 16 – 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 17 – 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 18 – 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 19 – 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 20 – 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 21 – 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.

### Slide 22 – STRIDE Threat Types

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.

# 3.0 References

Microsoft Corporation. “*SDL Threat Modeling Tool Manual*,” Microsoft SDL Threat Modeling Tool v3.1.3 Beta help documentation. February 2009.