MS_logo_KMICROSOFT SDL - DEVELOPER STARTER KIT:

BANNED APIS (LEVEL 200)

Presenter's Guide

Version 1.0

The following documentation provides presenter’s notes for the Microsoft Security Development Lifecycle (SDL) Banned APIs (Level 200) 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, Visual C#, Visual C++, Visual Studio, Windows, Windows Server, SQL 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](#_Toc233529387)

[1.1 INTRODUCTION 2](#_Toc233529388)

[1.2 System Requirements 2](#_Toc233529389)

[1.3 Presentation Themes 2](#_Toc233529390)

[2.0 Microsoft SDL Code Analysis 2](#_Toc233529391)

[Overview 2](#_Toc233529392)

[Presentation Transcript 3](#_Toc233529393)

[Presentation Voiceover 3](#_Toc233529394)

[Slide 2 – Title Slide 3](#_Toc233529395)

[Slide 3 – Agenda 3](#_Toc233529396)

[Slide 4 – Microsoft Security Development Lifecycle (SDL) 3](#_Toc233529397)

[Slide 5 – Microsoft Security Development Lifecycle (SDL) Banned APIs Overview 4](#_Toc233529398)

[Slide 6 – Examples of Microsoft SDL Banned APIs 5](#_Toc233529399)

[Slide 7 – Examples of Real-World Risks from Banned APIs 5](#_Toc233529400)

[Slide 8 – The “n” Functions Are Also Banned 5](#_Toc233529401)

[Slide 9 – Unsafe “n” Function Example 5](#_Toc233529402)

[Slide 10 – Detecting Instances of Banned APIs 6](#_Toc233529403)

[Slide 11 – Banned API Alternatives 6](#_Toc233529404)

[Slide 12 – Safer Alternative #1: StrSafe 7](#_Toc233529405)

[Slide 13 – StrSafe Example 7](#_Toc233529406)

[Slide 14 – Safer Alternative #2: Safe CRT 7](#_Toc233529407)

[Slide 15 – Safe CRT Example 8](#_Toc233529408)

[Slide 16 – Choosing StrSafe Versus Safe CRT 8](#_Toc233529409)

[Slide 17 – Conclusion 8](#_Toc233529410)

[Slide 18 - Appendix 9](#_Toc233529411)

[Slide 19 – Microsoft Security Development Lifecycle (SDL) 9](#_Toc233529412)

[Slide 20 – Microsoft Writing Secure Code Book Series 9](#_Toc233529413)

[Slide 21 – Microsoft Developer Network (MSDN) Security Developer Center 9](#_Toc233529414)

[Slide 22 – Secure Development Blogs 9](#_Toc233529415)

# 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.

* Banned APIs (Level 200) 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 Code Analysis

## Overview

When developing applications, it has been Microsoft’s experience that certain functions within the C runtime library, when incorrectly used, can lead to serious vulnerabilities in code. Several *Microsoft Security Bulletins* and accompanying patches have been required due to the incorrect use of such functions. As a result, Microsoft has had to commit significant engineering and testing efforts to produce each of these security bulletins and patches. A list of these problematic C runtime functions has been documented and the C runtime functions have been banned for use in new applications through the Microsoft SDL Banned APIs list. Any application that is to be developed in alignment with the Microsoft SDL must adhere to this list. The Microsoft SDL Banned APIs list also documents the appropriate alternative functions that have been approved for use in lieu of banned functions.

The insights gleaned by Microsoft, which are incorporated in its SDL, and more specifically, in this presentation focusing on Banned APIs, 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 Banned APIs (Level 200) presentation. The precise transcript text provided herein is also incorporated into the notes section of each slide in the Microsoft PowerPoint Banned APIs (Level 200) presentation itself for ease of reference.

## Presentation Voiceover

A voiceover of the Banned APIs (Level 200) presentation transcript below, approximately 25 minutes in length is also available to assist the presenter in becoming sufficiently acclimated with the subject matter addressed in the Banned APIs (Level 200) presentation, as well as to better understand the author’s perspective behind each slide in the presentation.

### Slide 2 – Title Slide

The Banned APIs (Level 200) 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 Banned APIs, which is a list of C programming language functions which, when used incorrectly, can easily lead to severe security vulnerabilities. In addition to discussing banned functions, this presentation also discusses how to detect instances of banned functions and provides safer alternatives to those functions, as prescribed by the Microsoft SDL.

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

### Slide 3 – Agenda

In this presentation, we will complete an overview of the Microsoft SDL and the Microsoft SDL Banned APIs. We will also discuss techniques that may be used to detect instances of banned functions in code. The presentation will conclude by providing an overview of safer alternatives that may be employed in lieu of using banned functions, as prescribed by the Microsoft SDL. The specific alternatives to banned APIs that will be discussed in this presentation are StrSafe and Safe CRT.

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

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

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

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

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

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

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

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

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

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

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

As was previously indicated, this presentation focuses on the Microsoft SDL Banned APIs. With respect to specific phases of the Microsoft SDL, the Microsoft SDL Banned APIs list is applied during the Implementation and Verification phases.

### Slide 5 – Microsoft Security Development Lifecycle (SDL) Banned APIs Overview

When developing applications, it has been Microsoft’s experience that certain functions within the C runtime library, when incorrectly used, can lead to serious vulnerabilities in code. For example, C runtime functions, such as strcpy and strcat, when called incorrectly, can lead to a serious vulnerability known as a buffer overrun. The Microsoft SDL Banned APIs is a documented list of such functions that are prohibited for use in new applications developed by Microsoft. Banned functions must be removed from legacy applications over time. Any application that is to be developed in conformance with the prescribed practices of the Microsoft SDL process must adhere to this list.

Removing the use of banned functions from applications is an effective way of reducing the risk of application attacks by malicious users with very little engineering effort. Preventing the use of banned functions early in the software development lifecycle, rather than later in the software development lifecycle, can save organizations significant engineering efforts. As you will see later in this presentation, several past Microsoft Security Bulletins have been attributed to the incorrect use of functions like these.

You will notice that the Microsoft SDL Banned APIs list focuses almost exclusively on preventing the use of functions that can lead to buffer overflows. The Microsoft SDL team will be updating this list as new threats that can be effectively mitigated by using safer functions and libraries emerge. It is important to mention that the Microsoft SDL Banned APIs list is development platform independent. Development teams engineering applications based on Microsoft platforms and non-Microsoft platforms, such as Linux, can greatly benefit from this list.

It is important to mention that the Microsoft SDL Banned APIs list is most useful to organizations in the scenarios where application source code is available. In situations where a library is used without the availability of source code, additional care, such as only using that library in a least privilege scenario, should be taken.

Lastly, the insights gleaned by Microsoft, which are incorporated in its SDL, and more specifically, in this presentation focusing on Banned APIs, 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 – Examples of Microsoft SDL Banned APIs

Provided in this slide is a sample of the current functions listed in the Microsoft SDL Banned APIs list. As you may notice, several commonly used functions, such as strcpy and strcat, have been banned by the Microsoft SDL for use in all new applications. That is, any application that is to be developed in conformance with the prescribed practices of the Microsoft SDL process must not use any of these banned functions. In addition to specifying banned functions, the Microsoft SDL Banned APIs list also specifies the appropriate functions that should be used in lieu of the banned functions. These safer alternative libraries will be discussed in more detail later in this presentation.

More than 100 different functions are listed as banned functions in the Microsoft SDL Banned APIs list. The full list can be found in Chapter 19 of the Microsoft SDL book (see [http://www.microsoft.com/mspress/books/8753.aspx](http://go.microsoft.com/?linkid=9672766)) or online on MSDN (see [http://msdn.microsoft.com/en-us/library/bb288454.aspx](http://go.microsoft.com/?linkid=9672767)).

### Slide 7 – Examples of Real-World Risks from Banned APIs

Here are examples of previous Microsoft Security Bulletins and patches that helped to establish the need for banning certain functions. For Microsoft, the incorrect use of banned APIs in applications have required the creation of numerous Microsoft Security Bulletins and accompanying patches. Each of these bulletins requires an intensive engineering process to fix the affected code and to produce a remediating patch. Additionally, several lengthy test passes are performed by verification teams to better ensure that the patch does not negatively impact other systems or applications. Deployment strategies for distributing the patch to customers must also be planned, executed and verified. Documentation to inform customers must be prepared and distributed through various channels. All of these steps in this extensive process result in significant effort and costs due to the incorrect use of only a single C runtime function.

Since prohibiting the use of banned functions in new applications through the Microsoft SDL, Microsoft has experienced significantly less reported vulnerabilities in its products. As a result, Microsoft has had to commit less engineering resources for producing security bulletins, which has resulted in significant development savings. For more information regarding all banned APIs, I encourage you to visit Microsoft’s Technet Web site at the URL provided in this slide.

### Slide 8 – The “n” Functions Are Also Banned

The C “n” functions are often used by developers due to their considering them as a safe alternative to banned functions. In Microsoft’s experience, the “n” functions themselves are difficult to use and are just as likely to be misused as are their less secure counterparts. As a result, Microsoft has also banned the use of the “n” functions in new applications.

In particular, the “n” functions do not null-terminate overflowed buffers. This can become problematic since other functions reading these un-terminated buffers have no means of knowing where the buffer logically ends. Not having this information could lead to buffer overflow conditions in code. Also, whenever a buffer overflow occurs, no indication by the “n” functions is ever given to the developer.

Let’s now see an example of why the “n” functions may be difficult for developers to use safely.

### Slide 9 – Unsafe “n” Function Example

Here is a classic example of why “n” functions can be problematic. Can you see the problem with this code?

(Mouse click)

The problem is that during the strncpy of pszSrc into the szDest buffer, if pszSrc is greater than or equal to 50 bytes, strncpy will not null-terminate szDest. This is actually expected behavior, as it is defined in the C99 or ISO/IEC 9899:1999 specification. Since the destination buffer is not null-terminated, it can cause unexpected application behavior and could also cause a buffer overflow condition later in code.

(Mouse click)

The “n” functions can be an effective alternative to banned functions if they are used correctly. If organizations choose to continue to use the “n” functions, then extra care is required by developers to ensure that the size of destination buffers has been correctly calculated.

Due to the extra care required to use the “n” functions and the ease which these functions can still lead to serious vulnerabilities, Microsoft has chosen to take the more defensive approach by banning the “n” functions for use in new applications per the Microsoft SDL. In lieu of the banned functions, including the “n” functions, Microsoft recommends StrSafe and Safe CRT as safer alternatives. Both of these alternatives will be discussed later in this presentation.

### Slide 10 – Detecting Instances of Banned APIs

Prohibiting the use of banned functions in applications can greatly reduce the overall exposure to attack. These banned functions can be detected in several ways.

(Mouse click)

For developers using Microsoft Visual Studio 2005, the compiler will automatically emit warning messages whenever it detects a banned function instance at compile-time.

(Mouse click)

All C4996 compiler warnings should be investigated to ensure that the function being flagged is not banned for use. In addition to C4996 warnings, developers should also be weary of pragma declarations that disable C4996 warnings.

(Mouse click)

Another method that can be useful to detect instances of banned functions is to use the banned.h header file that is included in the Microsoft SDL book companion CD. When this header file is added as the first header file in an application, it will detect and warn developers of all banned function instances. Since C and C++ are implemented on a variety of different platforms, this header can also be used for applications built on non-Microsoft platforms.

The authors of the Microsoft SDL book have also made this header file available as a download from the MSDN download center at <http://download.microsoft.com/download/2/e/b/2ebac853-63b7-49b4-b66f-9fd85f37c0f5/banned.h>.

(Mouse click)

The final method for detecting instances of banned functions is through conducting a code review process. In addition to looking for common coding patterns that can lead to vulnerabilities, verification teams should also search for instances of banned functions in code and for pragma declarations that disable banned function warnings.

### Slide 11 – Banned API Alternatives

In addition to documenting a detailed list of banned functions, the Microsoft SDL Banned APIs list also provides acceptable alternative functions developers can use. The Microsoft SDL recommends using the functions found in the StrSafe and Safe CRT libraries as safer alternatives. Though not required by the Microsoft SDL, if you are developing an application in C++, the std::string template class can also be used as a safer replacement. More information regarding this template class can be found at <http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38110>.

It is important to note that while replacing banned functions with safer alternatives can certainly improve the quality of code and reduce the exposure for attack, doing so does not guarantee or warrant applications to be free from vulnerabilities. While it may be more difficult to misuse these safer functions, misuse is still possible. Developers who use these safer functions should still take care to verify that destination buffer sizes are correct.

### Slide 12 – Safer Alternative #1: StrSafe

The first replacement library recommended by the Microsoft SDL is the StrSafe library. StrSafe is a library that provides additional processing to reduce exposure to buffer overflow attacks for C/C++ applications. The StrSafe library is distributed with the Windows Core SDK. This library has several noteworthy advantages. The first advantage is that functions in this library guarantee that all string operations are null terminated. If you recall, this was a key deficiency of the standard “n” functions, which did not always null terminate destination buffers. The next advantage is that all functions in this library return a status code that indicates the success or failure of a function call. Error codes are useful to developers to help them determine if an application error, such as a buffer overflow, has occurred. The final advantage is that each function in the StrSafe library requires a destination buffer size parameter, which helps ensure that the function does not write past the end of the buffer and cause a buffer overflow.

### Slide 13 – StrSafe Example

Let’s now see how StrSafe can be applied to existing code to make it more resilient to buffer overflow attacks.

(Mouse click)

Here is some sample code that reads two char \* parameters: s1 and s2. A buffer of 32 bytes is allocated on the stack and s1 is copied into the buffer. The parameter s2 is then concatenated into this buffer. Can anyone see what is wrong with this code?

There are two key security concerns regarding this code. The first is that s1 is copied into the temp buffer without checking the size of s1 to ensure that its contents can fit within a 32 byte buffer. If the length of s1 is greater than 32 bytes, this copy operation can lead to a buffer overflow condition. The second problem is that parameter s2 is then concatenated into the temp buffer with the assumption that there is sufficient space left for s2 after the strcpy of s1 is performed. If the length of s1 plus s2 exceeds the maximum size of the temp buffer, a buffer overflow will occur.

Now let’s see how StrSafe can be applied to reduce the exposure of buffer overflow attacks to this code.

(Mouse click)

To use StrSafe, developers simply need to include the strsafe.h header file and then replace each banned function with the appropriate StrSafe replacement. Notice that each function returns a status code that developers can use to assess the success or failure of a function call. Finally note that the StrSafe function takes a destination buffer size parameter to ensure that the function does not write past the end of the destination buffer and create a buffer overflow condition.

### Slide 14 – Safer Alternative #2: Safe CRT

The C runtime library included with Microsoft Visual Studio 2005 and higher has been enhanced to include more secure versions of common C runtime functions. This security enhanced C runtime library is known as the Safe CRT library. See <http://msdn.microsoft.com/en-us/library/8ef0s5kh(VS.80).aspx> for more information about the Safe CRT library.

The Safe CRT library provides a useful feature called Secure Template Overloads. Whenever \_CRT\_SECURE\_CPP\_OVERLOAD\_STANDARD\_NAMES is defined to 1, the compiler will automatically change banned function calls to use safer functions. This provides developers with an efficient way to automatically enhance older applications that may still be using banned functions to use more secure versions.

### Slide 15 – Safe CRT Example

Let’s now see how Safe CRT can be used to improve the resiliency of C code to buffer overflow attack.

(Mouse click)

Here is some sample code that allocates a 32 byte buffer called temp onto the stack, and then copies the s1 argument into temp. Can anyone see what is wrong with this code?

The problem with this sample code is that if s1 is longer than 32 bytes, then the strcpy of s1 into temp will result in a buffer overflow condition. Now let’s see how Safe CRT can be used to reduce the exposure of attack for this code.

(Mouse click)

Like StrSafe, developers can manually fix existing code to use safer functions found in Safe CRT; however the Safe CRT library comes with a useful feature called Secure Template Overloads. When this feature is enabled, the compiler will automatically replace banned functions with safer functions. To use Secure Template Overloads, developers need only to define \_CRT\_SECURE\_CPP\_OVERLOAD\_STANDARD\_NAMES to 1. The compiler will then look for banned functions and then convert banned functions into safer versions, as shown below. Also, buffer sizes, if known at compile-time, will be automatically provided by the compiler.

### Slide 16 – Choosing StrSafe Versus Safe CRT

There may be some confusion as to specifically which of StrSafe and Safe CRT alternative libraries developers should select when developing applications in the absence of banned APIs. Either of these libraries will be appropriate depending on specific application engineering situations. For example, if developers are engineering kernel mode applications, StrSafe is the most appropriate library to use. However, if other vulnerabilities in addition to buffer overflows are a concern to developers, then it is most appropriate to use the Safe CRT library.

In some cases you may not have a choice. Certain banned functions may not have safer alternatives available in one library, but may have safer alternatives in another library. For example, a safer alternative function to the C runtime function itoa is available in Safe CRT, but not in StrSafe.

This table has been excerpted from Chapter 19 of the Microsoft SDL book and is useful in helping developers decide which library is most appropriate for corresponding application engineering situations.

### Slide 17 – Conclusion

This concludes the discussion on the Microsoft SDL Banned APIs. When developing applications, it has been Microsoft’s experience that certain functions within the C runtime library, when incorrectly used, can lead to serious vulnerabilities in code. Several Microsoft Security Bulletins and accompanying patches have been required due to some of these functions being incorrectly used by developers. By simply avoiding the use of these functions altogether, Microsoft could have prevented the need for several past security bulletins. Microsoft has therefore created the Microsoft SDL Banned APIs list. This list documents the set of known functions that can easily lead to vulnerabilities, and also provides safer function replacements. Any new application that is developed by Microsoft using the Microsoft SDL is prohibited from using these banned functions. Further, Microsoft has established a strategy to ensure that all banned functions will be removed from its legacy applications over time.

Traditionally, developers looked to the C “n” functions as a safe replacement for banned functions; however, due to the limitations of the “n” functions and the ease at which these functions can also be misused, Microsoft has also banned the use of “n” functions for safer alternatives.

Developers can easily detect instances of banned functions in their code through a variety of methods. One method is to use the Microsoft Visual Studio 2005 compiler, which automatically alerts developers to the presence of banned functions. Another method is to use the banned.h header file that is found on the Microsoft SDL book companion CD and online on MSDN. The final method is to include flag instances of banned functions during security code review efforts.

The functions in the StrSafe and Safe CRT libraries are the recommended alternatives for the functions list in the Microsoft Banned APIs list. While these alternatives make it more difficult for developers to accidentally introduce vulnerabilities into code, they can still be used incorrectly. Developers should ban the use of functions found in the Microsoft SDL Banned APIs list, as well as remain vigilant in following other security implementation best-practices.

Lastly, the insights gleaned by Microsoft, which are incorporated in its SDL, and more specifically, in this presentation which focused on Banned APIs, 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 18 - Appendix

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

### Slide 19 – 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 20 – 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 21 – 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 22 – 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).