MS_logo_KMICROSOFT SDL - DEVELOPER STARTER KIT:

SECURE IMPLEMENTATION PRINCIPLES (LEVEL 100)

Version 1.0

The following questions accompany the materials for the Microsoft SDL - Developer Starter Kit Secure Implementation 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 Microsoft SDL - Developer Starter Kit Content Comprehension Questions 2](#_Toc233622851)

[1.1 Introduction 2](#_Toc233622852)

[2.0 Secure Implementation Principles (Level 100) Questions 2](#_Toc233622853)

# 1.0 Microsoft SDL - Developer Starter Kit Content Comprehension Questions

## 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 has developed 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 questions that may be used to ascertain comprehension of the subject matter addressed within the Microsoft SDL Training Module: Secure Implementation Principles (Level 100) presentation. These questions have been designed to enable the presenter to ascertain the extent at which the participating personnel with application development responsibilities have comprehended the subject matter addressed in the Secure Implementation Principles (Level 100) training module, as well as enabling the presenter to assess participants’ ability to apply the subject matter addressed to practical secure and trustworthy application development scenarios.

# 2.0 Secure Implementation Principles (Level 100) Questions

**Question #1:** True or false: Data that crosses a trust boundary and is used within an application should be validated.

1. True.
2. False.

**Answer:** The correct answer is “**A**” (True). Applications that read data from users and other programs make certain assumptions about the data, such as the format of the data and the type of data. In certain cases when those expected formats are not adhered to and an application consumes the data, malicious users may be able to compromise the application. Therefore, any data from an un-trusted source should be validated.

**Question #2:** Input validation is most effective when which of the following is considered?

1. Data format.
2. Data length.
3. Data type.
4. Data range.
5. All of the above.

**Answer:** The correct answer is “**E**”. Input validation is most effective when data is evaluated for correctness regarding format, length type and range.

**Question #3:** True or false: Applications written in managed languages, such as C# and Java are immune from buffer overflow attacks.

1. True.
2. False.

**Answer:** The answer is “**B**” (False). Buffer overflows have traditionally been a problem for native programming languages, such as C and C++, and much less of a problem for managed languages, such as C# and Java. However, it cannot be assumed that managed languages are immune from these types of attacks since the runtime environment of these languages themselves is written in a native programming language. There are also constructs in managed programming languages that call directly to native code that when not used correctly could facilitate buffer overflow attacks.

**Question #4:** True or false: Developers should create their own custom cryptographic algorithms for production use.

1. True.
2. False.

**Answer:** The answer is “**B**” (False). Standard cryptographic algorithms undergo years of review and scrutiny by academic and industry experts prior to being considered for standard approval. Developers can certainly create custom cryptographic algorithms; however, those algorithms will not have undergone the same rigorous testing as have the standard algorithms. It will be difficult to provide the necessary assurance that attests to the robustness and safety of standard cryptographic algorithms. Therefore, developers should use standard cryptographic algorithms in lieu of custom developed algorithms whenever possible.

**Question #5:** A banking application reads an 8-digit account number from users and executes a series of tasks based on the account number. The application development team has indicated that they have implemented steps to validate all inputs and has asked for you to review the following code to ensure that sufficient validation is in place. How can the validation code shown below be improved?

public void Execute(String AccountNumber)

{

// Input validation

if ( (AccountNumber == null) || (AccountNumber.Length != 8) )

{

// Bad data, so return

return;

}

// Execute some critical task with CreditCardNumber

...

}

1. The above invalidation code is sufficient, no additional validation is required.
2. The above code should validate the length of the account number input.
3. The above code should validate the format and range of the account number input.

**Answer:** The correct answer is “**C**”. The validation code already tests for the length of the incoming data and to some degree the type of data. Therefore, ensuring that the incoming data meets the expected format (8 digits in a row) and range (every character is a digit) are the remaining criteria for which the validation code should check / verify.

**Question #6:** A developer has created some code to decrypt a stream (created outside the method shown below) and has asked you for your feedback. In particular, that developer wants to know if he has violated any general security implementation best practices. How does the following code violate general secure implementation best practices?

public void DecryptStoredInfo()

{

byte[] Key = new byte[] {0xAB, 0xCD, 0xEF, 0x12, 0x34};

byte[] IV = new byte[] {0x12, 0x34, 0x56, 0x78, 0x9A};

// Create a crypto stream with the stream (read elsewhere), Key and IV

CryptoStream cStream = new CryptoStream(SomeStream,

New TripDESCryptoServiceProvider().CreateDecryptor(Key,IV),

CryptoStreamMode.Read);

StreamReader sReader = new StreamReader(cStream);

// Execute some critical task with the decrypted data

...  
 }

1. The code above does not validate input.
2. The code above stores cryptographic secrets within source code.
3. The code above does not use standard cryptographic algorithms.
4. The code above does not violate any secure implementation guidelines.

**Answer:** The correct answer is “**B**”. Developers should avoid storing sensitive cryptographic information, such as keys, within source code. The code does not read any input data to validate so answer “A” is incorrect. Finally, Triple DES is a standard cryptographic algorithm and so answer “C” is also incorrect.