MS_logo_KMICROSOFT SDL - DEVELOPER STARTER KIT:

SQL INJECTION (LEVEL 200)

Version 1.0

The following questions accompany the materials for the Microsoft SDL - Developer Starter Kit SQL Injection (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, 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](#_Toc233632015)

[1.1 Introduction 2](#_Toc233632016)

[2.0 SQL Injection (Level 200) Questions 2](#_Toc233632017)

# 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: SQL Injection (Level 200) 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 SQL Injection (Level 200) 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 SQL Injection (Level 200) Questions

**Question #1:** The following method accepts a string parameter named EmployeeNumber and dynamically creates and executes a SQL query created using the potentially un-trusted data. Does this method contain a SQL injection vulnerability?

public void InsertAccountNumber(String EmployeeNumber)

{

// Create a dynamic SQL statement

String sqlQuery = “SELECT \* FROM HrTable WHERE Num = @pENum”;

SqlParameter pENum = new SqlParamater(“@pENum”,SqlDbType.NVarChar,150);

pENum.Value = EmployeeNumber;

SqlComand sqlCmd = new SqlCommand(sqlQuery,

new SqlConnection(connectionString));

sqlCmd.Parameters.Add(pENum);

// Execute the sql command

…

}

1. The code above does not contain a SQL injection vulnerability.
2. The code above contains a SQL injection vulnerability because length of the un-trusted data may exceed 150 characters.
3. The code above contains a SQL injection vulnerability because the EmployeeNumber parameter is inserted directly into the SQL query.
4. An application dependency assignment.

**Answer:** The correct answer is “**A**”. The code above does not contain a SQL injection vulnerability. The dynamic SQL query is constructed using the placeholder @pENum for the un-trusted data EmployeeNumber. The un-trusted data is parameterized using a SqlParameter object and the type and length of the un-trusted data is explicitly defined. Since the un-trusted data is parameterized prior to execution, this will indicate to the SQL database engine not to execute any executable SQL statements that may be embedded in EmployeeNumber.

**Question #2:** Which of the following platforms are susceptible to SQL injection attacks?

1. Oracle.
2. Microsoft SQL Server.
3. MySQL.
4. All of the above.

**Answer:** The correct answer is “**D**”. SQL injection attacks target application logic based on the standardized SQL language. Therefore, any database product or platform that understands SQL is potentially susceptible to this type of attack.

**Question #3:** Which of the following mitigations can be used to effectively reduce the risk from SQL injection attacks?

1. IPsec.
2. SSL.
3. Parameterized queries and input validation.
4. All of the above.

**Answer:** The correct answer is “**C**”. The risk from SQL injection attacks can be effectively reduced using parameterized queries coupled with input validation. Parameterized queries mark incoming parameters as data-only, which instructs the underlying database engine not to execute any SQL statements that may be embedded in those parameters. Input validation helps to ensure that any incoming data conforms to expected formats, types, ranges and lengths. Since SQL injection attacks occur at the application-level, answers b and c are incorrect because both of these technologies are implemented before the application-level.

**Question #4:** SQL injection attacks can originate from which of the following sources?

1. Authenticated users.
2. Non-authenticated users.
3. All of the above.

**Answer:** The correct answer is “**C**”. SQL injection attacks are possible whenever an application reads input, constructs a dynamic query based on that input and then executes the query accordingly. Since input can originate from both authenticated and non-authenticated users, both of these types of users are potentially sources of SQL injection attacks.

**Question #5:** Which of the following code scanning tools can help you identify potential SQL injection vulnerabilities in your application implementation?

1. Microsoft FxCop.
2. Microsoft Source Code Analyzer for SQL injection for ASP.
3. All of the above.

**Answer:** The correct answer is “**C**”. Each of the tools from Microsoft listed above can help developers and testers identify SQL injection vulnerabilities for various programming languages. Microsoft FxCop will be effective for applications developed through using the .NET Framework. Finally, the referenced Microsoft Source Code Analyzer tool is effective for applications developed with classic ASP.

**Question #6:** A developer is creating an application with a database backend and has asked you to review his implementation. The code shown below contains two methods. The first method, SaveAccountNumbers, reads an array of account numbers and then writes those account numbers into a local file. The second method, ProcessAccountNumbers, picks up any saved account numbers and inserts them into a database. The application developer suspects that his code may be vulnerable to a SQL injection attack, but is unsure. Which of the statements below is true about the code shown below?

public void SaveAccountNumbers(String[] AccountNumbers)

{

// Save the account number

using (StreamWriter sw = new StreamWriter(“accounts.txt”))

{

foreach (String AccountNum in AccountNumbers)

{

sw.WriteLine(AccountNum);

}

}

// Process the account numbers now!

ProcessAccountNumbers();

}

public void ProcessAccountNumbers()

{

// Read account numbers  
 using (StreamReader sr = new StreamReader(“accounts.txt”))

{

String AccountNumber;

// Read and process account numbers

while ((AccountNumber = sr.ReadLine()) != null)

{

String SQLQuery = String.Format(

“INSERT INTO AccountsTable VALUES (‘{0}’)”, AccountNumber);

SqlCommand command = new SqlCommand(SQLQuery,);

// Execute SQL command object …

}

}

}

1. The above code does not contain a SQL injection vulnerability because the dynamic query executed is not created with any un-trusted data.
2. The above code does not contain a SQL injection vulnerability because only SELECT statements are affected by SQL injection attacks.
3. The above code contains a SQL injection vulnerability because the query was dynamically generated then executed using un-trusted data read from the accounts.txt file.
4. The above code contains a SQL injection vulnerability because only INSERT statements are affected by SQL injection attacks.

**Answer:** The correct answer is “**C**”. The data that was written into the accounts.txt file was read from un-trusted sources and therefore any dynamic SQL queries that are built using data from accounts.txt may be susceptible to SQL injection attacks. Answer “A” is incorrect because a dynamic SQL query is indeed built and executed using un-trusted data. Answers “B” and “D”, which suggest that only certain SQL statements are susceptible to SQL injection attacks, are also incorrect. SQL injection attacks are not necessarily bound to certain SQL statements, such as SELECT and INSERT, and are a potential threat whenever a SQL query is dynamically developed and executed using un-trusted data.