MS_logo_KMICROSOFT SDL - DEVELOPER STARTER KIT:

COMPILER DEFENSES (LEVEL 300)

Version 1.0

The following questions accompany the materials for the Microsoft SDL - Developer Starter Kit Compiler Defenses (Level 300) 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](#_Toc233598831)

[1.1 Introduction 2](#_Toc233598832)

[2.0 Compiler Defenses (Level 300) Questions 2](#_Toc233598833)

# 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: Compiler Defenses (Level 300) 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 Compiler Defenses (Level 300) 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 Compiler Defenses (Level 300) Questions

**Question #1:** Image randomization works by:

1. Randomizing the offsets of libraries and applications so as to reduce known address predictability that may aid malicious users in low-level execution flow control attacks.
2. Placing a random memory value positioned between local variables and return addresses in stack frames.
3. Modifying the endianness of an application to at load time to randomly use big-endian, little-endian and medium-endian ordering.
4. Randomizes the value of the return address in each stack frame.

**Answer:** The correct answer is “**A**”. Image randomization works by randomizing the image base of libraries and applications at load time. This helps to prevent the successful execution of malicious user attacks that depend on known memory offsets. Answer b describes the strategy employed by the Visual C++ compiler /GS buffer overflow protection feature, and answers c and d are fictitious application protection strategies.

**Question #2:** The goal of the Visual C++ compiler /GS is to protect which of the following memory regions?

1. Heap forward pointers.
2. Heap backwards pointers.
3. Stack frame return address pointers.
4. Locally declared variables.

**Answer:** The correct answer is “**C**”. The Visual C++ compiler /GS flag is designed to protect against direct write attempts to the return address within stack frames. Security cookies are not used to protect heap-memory or locally declared variables, which means answers ”A”, “B” and “D” are incorrect.

**Question #3:** The /SAFESEH linker flag can be used to help protect applications from attacks that:

1. Attempt to execute code in data pages.
2. Attempt to corrupt exception handlers that are stored on the stack.
3. Attempt to corrupt exception handlers that are stored on the heap.

**Answer:** The correct answer is “**B**”. The /SAFESEH linker flag causes the address of exception handlers to be written in the portable executable (PE) header. When an exception is raised, the address for the corresponding exception handler is retrieved from the stack and checked against the valid addresses previously stored. If the retrieved exception handler address from the stack does not match any known valid exception handler addresses, the process is halted by the operating system. Answer “A” is incorrect because it corresponds to the protection strategy employed by the /NXCOMPAT flag. Answer “C” is incorrect because exception handlers are stored on the stack and not the heap.

**Question #4:** Buffer overflow attacks often attempt to jump into code that is stored within allocated application buffers, such as those stored within application stacks and heaps. Knowing this, an engineer comes up with a strategy whereby all non-code memory regions will be marked as non-executable. Then, whenever any attempt to execute code from these non-code regions is made, the operating system will halt the process, thus also halting the current attack. This strategy is similar to the strategy employed by which of the following compiler-enabled defenses?

1. Visual C++ /GS buffer overflow protection.
2. Data Execution Prevention (DEP).
3. Image randomization.
4. Safe Exception Handling.

**Answer:** The correct answer is “**B**”. Data Execution Prevention (DEP) works in this fashion and monitors and prevents any attempts to execute code from non-code memory regions.

**Question #5:** A malicious user attempts to exploit a buffer overflow vulnerability by jumping into a known address of a function within ntdll.dll and observes that his attack is consistently failing. Upon closer inspection with a debugger, he notices that the address of the function that he is attempting to jump to changes after each application load, similar to what is shown below. Which compiler-enabled defense is preventing the malicious user from successfully exploiting the buffer overflow vulnerability?

* function!ntdll.dll (0x12341234)
* function!ntdll.dll (0xabcd1234)
* function!ntdll.dll (0x1234abcd)
* function!ntdll.dll (0xababcdcd)

1. Visual C++ /GS buffer overflow protection.
2. Data Execution Prevention (DEP).
3. Image randomization.
4. Safe Exception Handling.

**Answer:** The correct answer is “**C**”. Whenever an application is compiled with the /DYNAMICBASE linker flag, that application will be randomly rebased at load time, thus making execution flow control attacks that rely on known addresses much more difficult to successfully employ.