MS_logo_KMICROSOFT SDL - DEVELOPER STARTER KIT:

SOURCE CODE ANNOTATION LANGUAGE (LEVEL 200)

Version 1.0

The following questions accompany the materials for the Microsoft SDL - Developer Starter Kit Source Code Annotation Language (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](#_Toc233629134)

[1.1 Introduction 2](#_Toc233629135)

[2.0 Source Code Annotation Language (Level 200) Questions 2](#_Toc233629136)

# 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: Source Code Annotation Language (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 Source Code Annotation Language (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 Source Code Annotation Language (Level 200) Questions

**Question #1:** What does the annotation shown in the code below indicate?

void OptionalSample

(

\_\_opt TCHAR \* buf

)

1. The argument buf can be NULL.
2. The argument buf cannot be NULL.
3. The argument buf is pre-validated for use.
4. The argument buf may or may not be supplied.

**Answer:** The correct answer is “**A**”. The \_opt annotation is a buffer annotation that is used to indicate that an argument can be NULL.

**Question #2:** If a developer wanted to indicate that an argument to a function is a buffer pointer and must not be NULL, which buffer annotation should they use?

1. \_deref\_opt.
2. \_deref.
3. \_opt.
4. \_in.

**Answer:** The correct answer is “**B**”. The \_deref annotation is used to indicate that an argument is a buffer pointer and that it must not be NULL. The \_deref\_opt parameter also indicates that the argument is a buffer parameter; however, the \_opt modifier indicates that the argument may be NULL. The \_opt argument alone indicates that an argument may be null, and finally the \_in annotation indicates that a function will only read from a given buffer.

**Question #3:** From which vulnerabilities does the source code annotation language protect applications?

1. Buffer overflow.
2. SQL injection.
3. Cross-site scripting.
4. LDAP injection.
5. None of the above.

**Answer:** The correct answer is “**E**”. The source code annotation language is used to help static analysis tools find more vulnerabilities with improved accuracy; it does not remediate vulnerabilities in code.

**Question #4:** If a developer wanted to annotate the following code to indicate to static analysis tools that the argument buf will be written to and that the element count is specified by the count variable, what combination of buffer annotations should the developer use?

int SampleFunction

(

TCHAR \* buf,

int count

)

1. \_out and \_bcount(count).
2. \_in and \_nz.
3. \_out and \_ecount(count).

**Answer:** The correct answer is “**C**”. The \_out annotation indicates that function SampleFunction will only write to the annotated buffer, and the \_ecount annotation is used to explicitly indicate the size of the buffer as an element count. Answer “A” is incorrect because \_bcount is used to indicate byte counts. Answer “B” is incorrect since \_in indicates that the buffer will only be read by the function, and \_nz indicates that the buffer may not be null-terminated and the presence of the null-terminating character does not indicate the end of the buffer.

**Question #5:** If a developer wanted to annotate the following code to indicate to analysis tools that the argument strbuf will be read and written to, and that it may be read or written up to the first null character, what combinations of buffer and advanced annotations should the developer use?

void ReadBuffer

(

LPTCH strbuf

)

1. \_in and \_format\_string.
2. \_inout and \_nullterminated.
3. \_in and \_nullterminated.
4. None of the above.

**Answer:** The correct answer is “**B**”. The buffer annotation \_inout indicates that the strbuf buffer will be read and/or written to, and the advanced annotation \_nullterminated indicates that strbuf is a buffer that may be read or written up to the first null character.