WideCharToMultiByte

The destination string buffer must be long enough to hold the same number of characters, not bytes, as contained in the source string.

Sean Barnum, Cigital, Inc. [vita[1]]

Copyright © 2007 Cigital, Inc.

2007-04-23

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 7243 bytes

Attack Category	• Malicious Input
Vulnerability Category	• Multibyte Character • Buffer Overflow
Software Context	• String Conversion MACROS • National Language Support
Location	• winnls.h
Description	The WideCharToMultiByte function maps a wide-character string to a new character string. The new character string is not necessarily from a multibyte character set. The destination string buffer must be long enough to hold the same number of characters, not bytes, as contained in the source string. Using the WideCharToMultiByte function incorrectly can compromise the security of your application. Calling the WideCharToMultiByte function can easily cause a buffer overrun because the size of the In buffer equals the number of WCHARs in the string, while the size of the Out buffer equals the number of bytes. To avoid a buffer overrun, be sure to specify a buffer size appropriate for the data type the buffer receives. The MultiByteToWideChar function performs the inverse operation of the WideCharToMultiByte function but suffers the same vulnerability.
APIs	**Function Name** / **Comments** WideCharToMultibyte() MultiByteToWideChar
Method of Attack	BufferOverflow
Exception Criteria	
Solutions	**Solution Applicability** / **Solution Description** / **Solution Efficacy**

1. http://buildsecurityin.us-cert.gov/bsi-rules/35-BSI.html (Barnum, Sean)

	Generally applicable.	To ensure that the size of the lpMultiByteStr (Destination) buffer is large enough to contain the converted string, you should call WideCharToMultiByte twice -- once to determine the size of the translated string and again to translate the string.	Effective.

Signature Details	int WideCharToMultiByte(UINT CodePage, // code page DWORD dwFlags, // performance and mapping flags LPCWSTR lpWideCharStr, // wide-character string int cchWideChar, // number of chars in string. LPSTR lpMultiByteStr, // buffer for new string int cbMultiByte, // size of buffer LPCSTR lpDefaultChar, // default for unmappable chars LPBOOL lpUsedDefaultChar // set when default char used);
Examples of Incorrect Code	

Examples of Corrected Code	`Example: (No Error Checking for brevity)`
	`LPWSTR pwszUserName; // Unicode user name`
	`PSZ pszUserName; // User name which already points to multi-byte string`
	`int nUserNameLenUnicode = lstrlenW( pwszUserName ); // Convert all UNICODE characters`
	`int nUserNameLen = WideCharToMultiByte( CP_ACP, // ANSI Code Page`
	`0, // No special handling of unmapped chars`
	`pwszUserName, // wide-character string to be converted`
	`nUserNameLenUnicode,`

	`NULL, 0, // No output buffer since we are calculating length NULL, NULL); // Unrepresented char replacement - Use Default pszUserName = new char[nUserNameLen]; // nUserNameLen includes the NULL character WideCharToMultiByte(CP_ACP, // ANSI Code Page 0, // No special handling of unmapped chars pszUserName, // wide-character string to be converted nUserNameLenUnicode, pszUserName, nUserNameLen, NULL, NULL); // Unrepresented char replacement - Use Default`
Source References	• Rough Auditing Tool for Security (RATS)[2] • Howard, Michael & LeBlanc, David C. *Writing Secure Code, 2nd ed*. Redmond, WA: Microsoft Press, 2002, ISBN: 0735617228. • http://msdn.microsoft.com/library/default.asp?url=/library/en-us/intl/unicode_2bj9.asp[3]
Recommended Resources	• http://msdn.microsoft.com/library/default.asp?url=/library/en-us/intl/unicode_2bj9.asp[4] • http://blogs.msdn.com/michkap/archive/2005/04/18/409095.aspx
Discriminant Set	**Operating System**: • Windows **Languages**: • C • C++

Cigital, Inc. Copyright

Copyright © Cigital, Inc. 2005-2007. Cigital retains copyrights to this material.

Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and “No Warranty” statements are included with all reproductions and derivative works.

For information regarding external or commercial use of copyrighted materials owned by Cigital, including information about “Fair Use,” contact Cigital at copyright@cigital.com[1].

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

1. mailto:copyright@cigital.com