Analysis of the Security of Web Browsers via API Calls and Function Return Values

Lorie M. Liebrock, Ph.D
Assistant Professor
liebrock@cs.nmt.edu

Horst D. Clausen, Ph.D
Research Faculty
kearney@cs.nmt.edu

Kalyan Bondili
bondili@cs.nmt.edu

Department of Computer Science
New Mexico Institute of Mining and Technology
Socorro, NM-87801

Abstract

Web browsers are the primary interface for accessing the information on the World Wide Web. This work examines the relative vulnerability of web browsers. More specifically, this work explores the use of WIN32 DLLs and API calls and how the modification of the return values and out parameters of the dangerous APIs affects browser.

Results suggest that the APIs and DLLs used by the four web browsers (Opera, Mozilla Firefox, Netscape Navigator, and Internet Explorer) are similar with few exceptions. None of the more rarely used APIs have been specifically used by attackers to exploit security holes in the web browsers. The modification of return values and out parameters definitely affects the performance of the web browsers and in some cases crashes the web browsers.

Recommendations for developers include using secure functions, verification of the returned values and out parameters, proper bounds checking for buffer overflow, and more informative error messages. Each of these improvements would make browsers more usable and less vulnerable.

Most importantly, this comparative study will help users make informed decisions about which browser to use based on relative performance in terms of security vulnerabilities rather than simple popularity.

1. Introduction

Web browsers are the interfaces that allow users to access the information on WWW. Different web browsers are available but the most popular application with the users is Internet Explorer (IE). With the increase in vulnerabilities in IE some users are switching browsers. One approach to determining whether the browsers are inherently different is by looking at the functions called and used by them. This study focuses on identifying whether different web browsers use different API (Application Programming Interface) calls and WIN32 DLLs, as well as the effects of manipulating the return values and out parameters after the execution of these functions. Of particular interest are well known functions, called Dangerous Functions, that have been used repeatedly in various attacks.

2. Related Work

Shinder [13] takes a look at what makes Web browsers vulnerable to malicious attackers, how popular Web browsers differ (or don't) in this regard, and what you can do to protect yourself when Web surfing, no matter which browser you choose. This paper essentially states that Internet Explorer users are more vulnerable, but it does not show any experiments that allow users to quantify vulnerability and make informed decisions.

Many security web sites, such as computercops.biz, secunia.com, www.kirupa.com, and sans.org provide information on current vulnerabilities in the various web browsers. These are specific attack announcement that do not address general comparisons.

There are also a number of services that will test your web browser for vulnerabilities, e.g., bcheck.scanit.be, browsercheck.qualys.com, and www.farm9.com. These services generally do not fully specify up front what will be tested, so users may or may not have a good idea of how vulnerable they are after testing. Further, to do a comparison, users would need to install and configure all browsers, then test each one, in order to evaluate relative vulnerabilities of browsers. Here we present the head to head comparison results for four browsers.

3. Objectives

The objectives of this work are as follows.

- Determining whether the win32 DLLs used by all the web browsers are similar or if there are any DLLs being used by one web browser not being used by others.
- Identifying whether the API calls made by different web browsers are significantly different from one another when they are being used to browse the same websites
- Comparing the behavior of different web browsers by causing the API (Application Programming Interface) calls to fail in the Dangerous Functions category.
- Verifying whether modifying the OUT parameter values of Dangerous Functions makes the browser non-functional or causes catastrophic failures by opening up security holes.
- Verifying whether bounds checking is being done when Dangerous Functions are being used.

The examination and manipulation of parameters is done using a powerful application testing tool called Holodeck (Enterprise Edition).

4. Holodeck

Holodeck Enterprise Edition (version 2.5.210, trial version) is a software testing tool which works with the "application under test" by inserting itself as a thin layer between the operating system and the application. HEE can be used to log API calls, modify the parameters of APIs (only the OUT parameters) and observe the behavior of the application in these situations. Dangerous functions (named as such by Holodeck) are a category of APIs that perform string copying, deal with critical sections, and perform various other operations.

Using HEE, the dangerous functions were caused to fail while they were being executed by each of the web browsers and their behavior was observed.

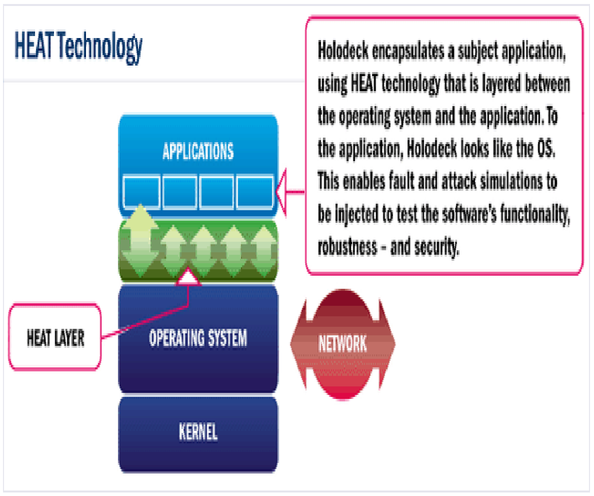
HEAT Technology
Holodeck encapsulates a subject application, using HEAT technology that is layered between the operating system and the application. To the application, Holodeck looks like the OS. This enables fault and attack simulations to be injected to test the software's functionality, robustness - and security.
APPLICATIONS
HEAT LAYER
OPERATING SYSTEM
NETWORK
KERNEL

Figure 1. HEAT Layer

Using the HEAT Layer HEE logs all the functions called by the application under test, the parameters given as input to the functions called (IN parameters) and OUT parameters. For example, when a string copy function is called with one parameter that contains the address of the buffer location to which the string is to be copied, the function copies the string to that buffer location and returns that value; in such a case it is also the OUT-parameter for the function. Besides the above information, the time stamps for each of the API calls, the return value, the name of the API called, and the DLL to which the API call belongs are also available.

Holodeck also helps the tester in creating faults in the application under test and observing the behavior of the application in such conditions. The various faults include network faults, disk faults, and memory faults. Resource constraints can be created by preventing access to particular DLLs, registry values, and files or folders.

The features of Holodeck mainly used in this work include the logging of all the APIs that are called by the web browsers, and testing the use of various functions by modifying the return values and out parameter values. The report generation feature was used to clearly identify the number of times each function was called.

5. Web Browsers

In this work the "application under test" is the web browser. With the aim of testing the current state-of-the-art web browsers the latest versions have been used for the test process. The following four are the most popularly used web browsers in the market today on the windows platform. The abbreviations listed in Table 1, with the specific browser version identification, are used to refer to the browsers through the rest of this paper.

Table 1. Browser Versions

Browser Name	Version	Name
Internet Explorer	6.0	IE
Opera	7.54	OP
Mozilla Firefox	1.0	FF
Netscape	7.2	NS

6. Test Configuration

The environment used to conduct these experiments is as follows.

Operating System:
Microsoft Windows XP Professional, SP2
Processor: Intel Pentium III processor 863 MHz
RAM: 256 MB of RAM
Holodeck Enterprise Edition (version 2.5.210)

7. Method

The following is a basic description used of the process used to analyzed the operation of the web browsers.

- Open HEE
- Start the application to be tested using HEE. Using HEE to start the application will allow HEE to log all the API calls made by the application from the beginning to the end of the use of application.
- Set the test to be executed (the modified return value and modified OUT parameters) when the API is executed by the browser application.
- Browse a website using the application while HEE is logging the APIs and waiting to execute the test created.
- As soon as the API call under test is used by the application, HEE executes the test thereby modifying the return values and other OUT parameter values (as specified earlier).
- Observe the behavior of the browser.

8. Normal functions

The normal functions are the APIs being used by the web browsers and logged by default by HEE. The functions in this category include COM Functions, File Functions, Library Functions, Network Functions, Process Functions, and Registry Functions. HEE logs these functions by default when any application is being tested. Here the normal functions are examined during web browser operation to explore differences between browser operations that may affect vulnerability.

8.1. Testing method

Four websites were selected to perform the testing. The websites and the browsing activity done in the browsers are described below.

www.yahoo.com Go to the website, click on the link for shopping and go to that page. Wait until the page loads completely and then stop the application (close the web browser)

www.nmt.edu Go to the website, choose catalog in the drop down menu on the left and click on it. After the page opens and completes loading, stop the application (close the web browser).

www.phrack.org Go to the website, and click on the link "unofficial" on the left side of the page. Once the page loads, click on the link. The link would open in a new page with "The page cannot be displayed" error. Stop the application (close the web browser).

www.sans.org Go to the website, and click on the link "Top 20 List" on the right side of the page. Wait for the page to load. After it completely loads, scroll to the bottom of the page and then stop the application (close the web browser).

The test procedure followed to create reports and API logs is as follows for each URL.

- Open HEE
- Select one URL
- For each of the web browsers perform the below steps
  - Create a new project in HEE
  - Select the functions to be logged and Open the Web browser using HEE
  - Visit the URL after the web browser loads
  - Perform the browsing activity
  - Close the browser
  - Analyze the log sequences and reports

8.2. Analysis 1 – Based on DLLs

The objective of this analysis was to determine whether any of the browsers was using a DLL different from any of the others. However, as the following table shows, all the browsers were using a common set of DLLs to access the APIs. No Win32 DLLs other than the ones mentioned below were being used by any of the browsers. All tested web browsers use Advapi32.dll, Kernel32.dll, Msvcrt40.dll, Ole32.dll, Ws2_32.dll.

8.3. Analysis 2 – Based on the APIs

The objective of this analysis was to identify if there are any functions that are used exclusively by one browser but not used by other browsers in browsing the four websites. The table below shows functions from each category called by one web browser but not used by the other web browsers. The functions in the File and Network Tables called by NS and FF are similar.

Table 2. File function category API calls

Browser	Function
IE	GetDiskFreeSpaceA GetDiskFreeSpaceExA GetDiskFreeSpaceExW GetDiskFreeSpaceW GetFileAttributesExA GetFileAttributesExW
OP	CopyFileExW CopyFileW GetCurrentDirectoryA GetDriveTypeA MoveFileW OpenFile SearchPathA
FF	MoveFileA MoveFileWithProgressA RemoveDirectoryA RemoveDirectoryW
NS	MoveFileA MoveFileWithProgressA RemoveDirectoryA RemoveDirectoryW

Table 3. Com, Library, Process function category API calls

Browser	Function
IE	Nil
OP	Nil
FF	Nil
NS	Nil

Table 4. Registry function category API calls

Browser	Function
IE	RegCreateKeyW RegDeleteKeyW RegDeleteValueA RegDeleteValueW RegEnumKeyA RegEnumKeyExW RegEnumKeyW RegEnumValueA RegEnumValueW RegQueryInfoKeyA RegQueryInfoKeyW
OP	Nil
FF	Nil
NS	RegDeleteKeyA

Table 5. Network function category API calls

Browser	Function
IE	Getsockopt
OP	WSACleanup WSASend
FF	Accept Listen WSAAccept
NS	Accept Listen WSAAccept

Based on the analysis above, it is clear that there are a few functions that are used by one browser but not by others. After finding these functions, research was performed using the internet to identify whether any of these functions have been used by hackers or viruses or worms to attack and cause security breaches. The results showed that none of the above functions have been specifically used in attacks to exploit a security hole in the web browsers. However, a trojan that attacked in the past has used functions that are commonly used by the above browsers to hook to various applications. Such an attack in the future might increase chances of system compromise because web browsers are one of the most commonly used applications by many users and a trojan of this sort can take advantage of the browsers execution to activate its malicious code.

Backdoor.HackDefender is a backdoor trojan component that hides processes, services, and files. It was discovered on March 12th, 2003. It hooks various APIs in application processes by first allocating memory area inside a host process. Then, the trojan injects its own code and handler functions into that area. Backdoor.HackDefender will then inventory the list of APIs, placing a jump instruction at the beginning of every API. When a program executes any of the hooked APIs, the jump instruction transfers control to the trojan's handler function (residing in the allocated memory area), which then calls the hooked API through a stub function. When the original API returns the handler function, the trojan performs its processing and filtering processes (for example, excluding a specific file from FindFirstFile/FindNextFile API), and then returns. APIs which have been used by trojans and are used by all of the above browsers include LoadLibraryA, FreeLibrary, LoadLibraryExW, WriteFile, ReadFile, and send. So if this trojan infects these browsers, there is a possibility that the browser will be affected and also the system on which the browser is being used.

After obtaining these results, two functions that were commonly used by all the browsers were selected and their execution and return values were manipulated using HEE. The results of browser behavior are shown in the table below.

In Table 8, for the deletecriticalsection function, a file not found error was set so that the function would not execute.

Table 6. Manipulating deletecriticalsection

Browser	Results
IE	Could not load at least the browser window
OP	It loaded the browser window; however loading the browser window took less time for this browser than the other three
FF	Similar to NS
NS	Loaded the browser window but took more time than normal to perform this function

In Table 9, for the closehandle function, the return value was set to an arbitrary value, 123456.

Table 7. Manipulating closehandle

Browser	Results
IE	As above
OP	Loaded the browser window and took more than normal time to perform this
FF	Similar to NS
NS	Loaded the browser window and took more than normal time to perform this

9. Dangerous Functions

Performing the logging of dangerous functions can be done along with the normal functions. However, when such logging is done the API logs become too big and Excel is not able to load the complete file. Hence, the tests mentioned above were re-executed. This time only the Dangerous Functions were logged. The purpose of logging Dangerous Functions is to focus on those functions which have been shown to increase vulnerability.

The same test procedure is used for Dangerous Functions as was used for normal functions.

9.1. Analysis 1 – Dangerous functions used

The tables below contain all of the functions being used by each of the browsers from the Dangerous Functions category. The functions InitializeCriticalSection, LstrcpnA, LstrcpyW, MultiByteToWideChar, and wcsncpy are used by all of the browsers tested.

Table 8. Dangerous functions in IE, OP

IE	OP
InitializeCriticalSection	InitializeCriticalSection
LstrcatA	LstrcpyA
LstrcatW	LstrcpynA
LstrcpyA	LstrcpyW
LstrcpynA	MultiByteToWideChar
LstrcpynW	wcscpy
LstrcpyW	wcsncpy
MultiByteToWideChar	
SetSecurityDescriptorDacl	
strncpy	
wcscat	
wcscpy	
wcsncat	
wcsncpy	

Table 9. Dangerous functions in FF, NS

FF	NS
_mbscpy	_mbscat
InitializeCriticalSection	_mbscpy
LstrcatW	InitializeCriticalSection
LstrcpyA	LstrcatW
LstrcpynA	LstrcpynA
LstrcpynW	LstrcpynW
LstrcpyW	LstrcpyW
memcpy	memcpy
MultiByteToWideChar	MultiByteToWideChar
SetSecurityDescriptorDacl	SetSecurityDescriptorDacl
strlen	strlen
strncpy	strncpy
wcsncpy	wcsncpy

The following table contains all the dangerous functions used exclusively by one browser but not by the other browsers.

Table 10. Exclusive dangerous functions

	IE	OP	FF	NS
Dangerous Functions	lstrcatA Wcscat Wcsncat		_mbscpy memcpy strlen	_mbscat _mbscpy memcpy strlen

10. Buffer Overflow Analysis

The Dangerous Functions category contains functions pertaining to string manipulations such as string copying and concatenation. [1] and [2] have list function calls that commonly lead to buffer overflow errors. They say that attackers often find buffer problems in code by searching for functions that are known to have bounds problems.

Table 11. Buffer oveflow functions

Strcpy	Strcat	Lstrcpy	Lstrcat	lstrcpyA
lstrcatA	lstrcpyW	lstrcatW	Lstrcpyn	Wcscat
lstrcpynA	Strncat	lstrcpynW	Wstrncat	Wstrcpy
Memcpy	Strncpy	Memmove	Wstrncpy	Scanf
Sprintf	Wscanf	Swprintf	Fgets	Gets
Fgetws	Getws			

After testing the web browsers by logging the Dangerous Functions it was obvious that the web browsers were using some of the above functions which can cause buffer overflows. Buffer overflows usually allow the attacker to modify the return address and execute code that results in opening a shell with the root privileges that the attacker can then use to further compromise a machine. Based on the above observations, the next step was to perform tests on the web browsers to identify the possible vulnerabilities that might arise by modifying the return values of the functions that were used for string copying and concatenation.

11. Function Testing

The following table shows all the functions corresponding to string operations and memory operations that are used by the web browsers and might cause buffer overflow problems.

Table 12. Functions to be tested

IE	OP	FF	NS
Lstrcat Lstrcpy Lstrcpyn Strncpy Wcscat Wcscpy	Lstrcpy Lstrcpyn Wcscpy	Lstrcat Lstrcpy Lstrcpyn Memcpy Strncpy	Lstrcat Lstrcpy Lstrcpyn Memcpy Strncpy

From Table 14, functions that were commonly used by all the browsers were selected. Lstrcpy and Lstrcpyn are two functions that are used by all the four browsers. The results from testing these two functions with modified return values are provided below. Next the functions that were used by only one web browser or two of them were selected and tested. Memcpy is one such function used by FF and NS only. Wcscat is used only by IE. Wcscpy is used by IE and OP. Lstrcat is used by all except OP. The test values that were set, the results and error messages are shown below.

11.1. Lstrcpy (used by IE, OP, NS, FF)

The **lstrcpy** function copies a string to a buffer. If the function succeeds, the return value is a pointer to the buffer. If the function fails, the return value is NULL.

The tables below show the attacks performed. Strdestination corresponds to lpString1 and Return Value is the value returned after execution of the function. For each browser the response to the injected value is shown.

Table 13. Lstrcpy attack #1 results

Attack #	1
Strdestination	No Change
Return Value	1
IE	Threw an exception and failed. The browser window was unable to load.
FF	The application threw an exception and failed. The browser window could not be loaded.
NS	Similar to FF.
OP	Browser window loaded, but typing the URL led to an error message: "Network Problem".

Table 14. Lstrcpy attack #2 results

Attack #	2
Strdestination	0
Return Value	1
IE	IE crashed. The browser window was trying to load and the process was terminated before it completed execution.
FF	Though the function failed, the browser opened. It did not load the webpage. The error message was "www.yahoo.com could not be found. Please check the name and try again"
NS	Similar to FF
OP	Browser did not load and it was trying to succeed the above function

Table 15. Lstrcpy attack #3 results

Attack #	3
Strdestination	0
Return Value	0
IE	Behavior similar to Attack #1
FF	Behavior similar to Attack #2
NS	Behavior similar to Attack #2
OP	Behavior similar to Attack #1

Table 16. Lstrcpy attack #4 results

Attack #	4
Strdestination	No Change
Return Value	0
IE	Behavior similar to Attack #1
FF	Behavior similar to Attack #2
NS	Behavior similar to Attack #2
OP	Behavior similar to Attack #1

The browser most affected by the modifications to the Lstrcpy function is IE. It is unable to load even the browser window. This could be a denial of service attack because it prevents the browser application from performing its normal operation. The other browsers at least loaded the browser and then showed an error message. The error messages give an impression to the user that the typed URL might be incorrect or that there is a network problem that is preventing access when the actual problem is with Lstrcpy which has been manipulated. This could be a possible denial of service attack. An attacker can use such a method to force the user to switch to another network or a different service.

The runtime reference library in the MSDN library states that using this function incorrectly can compromise the security of our application. The first argument, lpString1, must be large enough to hold lpString2 and the closing '\0', otherwise a buffer overrun may occur. Buffer overruns may lead to a denial of service attacks against the application if an access violation occurs. In the worst case, a buffer overrun may allow an attacker to inject executable code into our process, especially if lpString1 is a stack-based buffer. The reference library further suggests using alternative functions like StringCbCopy, StringCbCopyEx, StringCbCopyN, StringCbCopyNEx, StringCchCopy, StringCchCopyEx, StringCchCopyN, or StringCchCopyNEx.

The **lstrcpy** function has an undefined behavior if source and destination buffers overlapFunction: lstrcpyW(CLSID\, CLSID\). Clearly web browsers using lstrcpy have a vulnerability that is not present in browsers that don't use lstrcpy.

11.2. Lstrcpyn

The **Lstrcpyn** function copies a specified number of characters from a source string into a buffer. If the function succeeds, the return value is a pointer to the buffer. The function can succeed even if the source string is greater than *iMaxLength* characters. If the function fails, the return value is NULL.

Table 17. Lstrcpyn attack #1 results

Attack #	1
Buffer Pointer	No Change
Return Value	1
IE	Web browser loads. After typing the URL in address bar an error message pops up that says: "Can't access this folder. Path is too long." If we click ok on the error message, after a certain delay the web page loads.
FF	Opens the web browser window. After typing the URL in the address bar it loads the web page immediately without any problem
NS	Similar to FF
OP	Similar to FF

Table 18. Lstrcpyn attack #2 results

Attack #	2
Buffer Pointer	0
Return Value	1
IE	Browser window loaded. After typing the URL it was trying to load the web page. It took a lot of time but finally loaded the webpage
FF	Similar to attack #1 behavior
NS	Similar to attack #1 behavior
OP	Similar to attack #1 behavior

Table 19. Lstrcpyn attack #3 results

Attack #	3
Buffer Pointer	0
Return Value	0
IE	Similar to attack #1 behavior
FF	Similar to attack #1 behavior
NS	Similar to attack #1 behavior
OP	Similar to attack #1 behavior

Table 20. Lstrcpyn attack #4 results

Attack #	4
Buffer Pointer	No Change
Return Value	0
IE	Similar to attack #1 behavior
FF	Similar to attack #1 behavior
NS	Similar to attack #1 behavior
OP	Similar to attack #1 behavior

IE takes more time than any other browser in loading the browser window as well as the web page after typing the URL in the address bar. The reason for this might be that the function under test is used by IE more number of times compared to other browsers. This is one function where manipulating the return value or buffer pointer hasn't crashed IE; it only delayed the loading of webpage even though an error message popped up.

11.3. Lstrcat

The **lstrcat** function appends one string to another. If the function succeeds, the return value is a pointer to the buffer. If the function fails, the return value is NULL.

Table 21. Lstrcat attack #1 results

Attack #	1
Buffer Location	0
Return Value	0
IE	Took time to load the browser. After the URL was typed the web page corresponding to the URL also loaded. But it took a lot of time to perform this
FF	Opened browser window and loaded the webpage without any problem
NS	Similar to FF
OP	No Problem

Table 22. Lstrcat attack #2 results

Attack #	2
Buffer Location	No Change
Return Value	1
IE	Behavior similar to attack #1
FF	Behavior similar to attack #1
NS	Behavior similar to attack #1
OP	Behavior similar to attack #1

Table 23. Lstrcat attack #3 results

Attack #	3
Buffer Location	No Change
Return Value	0
IE	Behavior similar to attack #1
FF	Behavior similar to attack #1
NS	Behavior similar to attack #1
OP	Behavior similar to attack #1

Table 24. Lstrcat attack #4 results

Attack #	4
Buffer Location	0
Return Value	1
IE	Browser window loaded, but after typing the URL in the address bar it threw an exception and crashed
FF	Behavior similar to attack #1
NS	Behavior similar to attack #1
OP	Behavior similar to attack #1

IE was behaving similarly in the first three attacks. However, in attack #4 when the lpstring1 was set to 0 and return value was set to 1 indicating a success, the browser threw an exception and crashed.

FF and NS did not show any sign of speed degradation in either loading the browser window or loading the webpage after the URL was typed in the address bar.

OP was unaffected by the changes in either the buffer location or return value because it was not using the API.

Although the return values were modified, the returned addresses were locations that do not contain valid code. If skilled hackers perform such an attack they would put malicious code in those locations and give return values corresponding to those locations after the API call execution. This can cause catastrophic attacks or security breaches.

MSDN library states that using this function incorrectly can compromise the security of our application. The first argument, lpString1, must be large enough to hold lpString2 and the closing '\0', otherwise a buffer overrun may occur. Buffer overruns may lead to a denial of service attack against the application if an access violation occurs. In the worst case, a buffer overrun may allow an attacker to inject executable code into our process, especially if lpString1 is a stack-based buffer. MSDN suggests using one of the following alternatives: StringCbCat, StringCbCatEx, StringCbCatN, StringCbCatNEx, StringCchCat, StringCchCatEx, StringCchCatN, or StringCchCatNEx.

11.4. Memcpy

Memcpy experiments were only performed on NS and FF as OP and IE don't use this function. The results obtained by modifying the return value are shown below.

Table 25. Memcpy results

Return Value	FF	NS
0	The browser window was not loaded. The application terminated after repeatedly trying to use memcpy function and threw an exception	Similar to FF but did not throw an exception
A random integer (2917512)	As above, except that the exception was not thrown	As above, except that the exception was thrown

The above two attacks resulted in a denial of service for both browsers.

11.5. Wcscat

This function appends a string to another string. It is a wide-character version of strcat. The **wcscat** function appends *strSource* to *strDestination* and terminates the resulting string with a null character. The initial character of *strSource* overwrites the terminating null character of *strDestination*.

The table below shows the results obtained when the strDestination was modified to return arbitrary values.

Different values were set for the destination string and the behavior of IE, the only browser that uses wcscat, was observed. The results are shown in the following table.

Table 26. Wcscat results

Attack #	**Destination String**	**IE**
1	0	IE loads and returns with a "page could not be found" error
2	Random string	Crashes
3	Integer	The browser window loads and after typing the URL in the address bar the following message pops up: "Internet Explorer Could not open the search page" error

This function is used only by IE. The attacks used here cause a denial of service by preventing access to the webpage. Attack #2 has the possibility of denial of browser service because the application crashes when such attack is used. No return value is reserved for this function to indicate if there was an error when the function executed. Because the destination string is returned on success, there is a possibility that if the destination string contains code that causes a buffer overflow and it is used by another function, that the code would be executed and cause a security breach.

Another vulnerability that might arise is that the behavior of this function is undefined when the source and destination strings overlap. The runtime library reference on MSDN library has mentioned wcscat's reference section.

11.6 Wcscpy

This function copies a string to another string. It is a wide-character version of strcpy. The **wcscpy** function copies *strSource,* including the terminating null character, to the location specified by *strDestination*. This function returns the destination string. No return value is reserved to indicate an error.

The following table shows the results obtained when the Destination String was set to different values.

Table 27. Wcscpy results

Attack	Destination String	IE	OP
1	0	After typing the URL the pop up "Security Alert: your current settings do not allow this file to be downloaded" is shown a couple of times and then the browser crashes.	After typing the URL the pop up "Connection closed by remote server http://www.yahoo.com" is shown and OP does not crash.
2	"nonstop" (a Random String)	As above	Before loading the web browser itself OP threw an exception and crashed.
3	"323455" (a random number)	Threw an exception and crashed.	Loaded the browser, after typing the URL in the address bar, OP threw an exception and crashed.

IE crashes in all 3 attacks, but opera crashes in only 2 of the 3 attacks. IEs security alert might actually give the user wrong information. A function related to string copying has failed and IE throws an error that conveys that security settings do not allow that file to be downloaded. If the user is knowledgeable enough, he might be able to identify that it is a false Security Alert. However, for many IE users, this message may cause them to lower the security settings on their browser. This might be used as a preliminary attack by the hackers to make the user decrease the security settings and then compromise the machine using other means. This attack prevents the application from working, thus creating a denial of browser service attack.

Another problem that can be associated with this API is that there is no return value reserved to indicate an error when this function fails. So any other functions that might use the value obtained from the execution of this function cannot verify the correctness of execution because no return value is provided. If the returned destination string contains a shell script and it is executed by another function that uses this returned destination string then it might cause catastrophic problems by opening up a shell for the hacker to use with administrative privileges.

A third problem associated with wcscpy is that the behavior is undefined if the source and destination strings overlap.

A fourth problem associated with this function is that this function doesn't check for sufficient space in the string destination before copying the source string and it is a potential cause for buffer overruns. The run time reference library on MSDN library mentions this problem and suggests using a function bounded by the number of characters rather than this function to prevent such overflows.

12. Suggestions – Msdn and others

Security suggestions from other sources include the following.

- [3] mentions that if a program is using some dangerous APIs the need for using those APIs should be fully justified otherwise the API should not be used. Therefore the web browsers using the dangerous functions should justify the need for using the dangerous APIs.

- [5] mentions that APIs should be validated as they might cause potential security problems.

So web browsers can use the set of APIs that have been validated and found to be secure.

Some of the API calls (Lstrcpy, Lstrcat) used by web browsers might cause buffer overflow problems if incorrectly used. MSDN library suggests using alternate functions to prevent these problems. Web browser developers should make sure that the function is correctly used and when they think it might be incorrectly used, they should use the alternate API calls provided in the reference library.

13. Recommendations

A useful mechanism that can prevent such denial of service attacks for browsers is that once a function starts failing, the application can try using the function a few more times and if it continues to fail, it can revert to the secure functions that implement the required functionality using a secure method. This would prevent denial of browser service attacks, but not other vulnerabilities. The programmer can create a counter to verify how many times a particular function has failed, and if it increases beyond a certain threshold, the control flow can be modified and sent to another function that handles such problems or prints out useful and appropriate error messages. This would prevent confusion arising from false error message. One problem with this approach is that it would increase the code size.

The approach in the experiments was to modify the return values and OUT parameter values and return them after execution of dangerous functions. Each of the web browsers tried to use those new values and had problems. When using the dangerous APIs, if the values of buffer addresses are copied into another variable or location, the web browser can compare both the values once they are returned. If the values have been modified the application can make another call to the function and discard currently returned value.

Gracefully failing the application with appropriate error messages should be incorporated into the web browsers.

HEE should be enhanced to permit changing the IN parameter values. If this is allowed, a lot of valid and invalid strings can be given as input to the API calls and the robustness and security of the application can be tested.

14. Conclusions

After a careful study and comparison of the web browsers using Holodeck Enterprise Edition it can be concluded that all the web browsers are using similar DLLs, similar API calls excepting a few, and do not run into problems when long strings are given as part of the URL. Manipulating the return values and out parameter values has affected the functionality of the browser but has not opened up direct security holes. Also none of the web browsers has been attacked by specifically affecting the APIs used by them. The tests performed also convey that IE uses more dangerous functions than other browsers. So if bounds' checking is not being done for every use of the dangerous functions it might be targeted more by attackers.

By doing comparative studies such as this, users can make informed decisions about which browser (or other software) to use based on relative merits rather than simple popularity. In addition, studies such as this one reinforce the programming practices that should be used for secure software development.

15. References

[1] Ron Brandis, Common system security risks and vulnerabilities and How malicious attackers exploit them: A Bridgepoint Whitepaper, V1.0 9th August 2001

[2] Secure Coding – Design and Building Secure Applications by Foundstone, net-services.ufl.edu/security/ public/webinar_secure_coding.pdf, viewed 2/01/05

[3] Security Code Review, Microsoft Corporation, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/secmod/html/secmod94.asp, January 2004

[4] Backdoor.HackDefender, Symantec Security Update, http://www.sarc.com/avcenter/venc/data/backdoor.hackdefender.html, March 12, 2003

[5] Asp.Net.Vulnerability, Straight Talk for the Developers, http://rtfm.atrax.co.uk/boards/asp.net.asp?action=viewPost&postID=768, November 10, 2004

[6] lstrcpy Fucntion, Microsoft Developers Network, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winui/winui/windowsuserinterface/resources/strings/stringreference/stringfunctions/lstrcpy.asp, viewed 2/01/05

[7] strcat, wcscat, _mbscat, Microsoft Developers Network,

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vclib/html/_crt_strcat.2c_.wcscat.2c_._mbscat.asp, viewed 2/01/05
[8] Holodeck Enterprise Edition, Security Innovation, http://www.securityinnovation.com/holodeck/index.shtml, viewed 2/01/05
[9] Avoiding Buffer Overruns, Microsoft Developers Network, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/secbp/security/avoiding_buffer_overruns.asp, viewed 2/01/05
[10] Mozilla Security Reviewers' Guide, Mozzilla, http://www.mozilla.org/projects/security/components/reviewguide.html, May 2002, viewed 2/01/05
[11] Browser Statistics, Network Solutions http://www.w3schools.com/browsers/browsers_stats.asp, viewed 2/01/05
[12] Vulnerability and Virus Information – Netscape, Secunia, http://www.secunia.com, viewed 2/01/05
[13] Deb Shinder, Web Browser Vulnerabilities: Is Safe Surfing Possible?, Windows Security, http://www.windowsecurity.com/articles/Web-Browser-Vulnerabilities.html , Aug 05, 2004