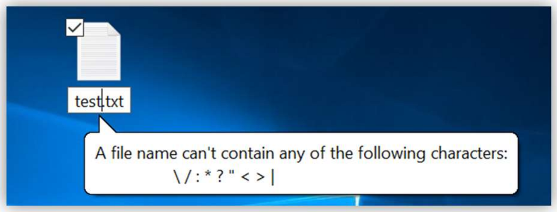
Windows

and the \/:*?"<>| characters

NONE
April 2018

TL;DR

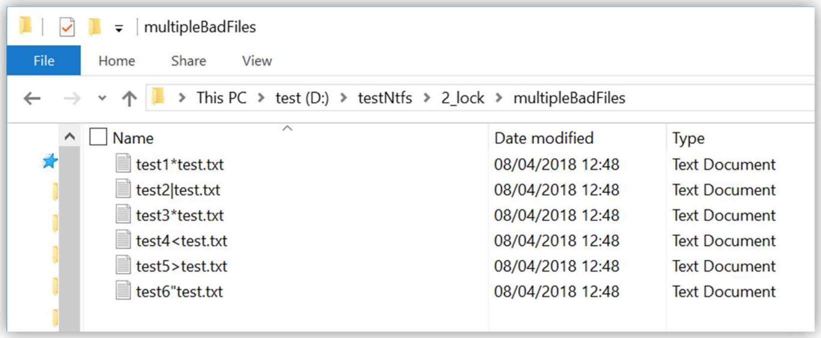


A file name can't contain any of the following characters:
\ / : * ? " < > |

As everybody knows, the characters \/:*?"<>| are prohibited in filenames on the Windows environment. As such characters can be used on Unix, I did some tests to see how it would be handled if I created files with bad names on an external device and mounted it on a Windows workstation.

For this experiment, I used a NTFS formatted USB drive and a Windows 10 operating system.

As shown in the following screenshot, the Windows file explorer is able to open directories with such files:



Name	Date modified	Type
test1*test.txt	08/04/2018 12:48	Text Document
test2|test.txt	08/04/2018 12:48	Text Document
test3*test.txt	08/04/2018 12:48	Text Document
test4<test.txt	08/04/2018 12:48	Text Document
test5>test.txt	08/04/2018 12:48	Text Document
test6"test.txt	08/04/2018 12:48	Text Document

After trying to interact with these documents, I noticed some side effects such as denial of service and parameter injections which are presented in this report.

Directory locking and relative filenames

After creating two documents named « aaa.txt » and « aaa.txt" » in a directory, explorer considered it as corrupted and were not able to open it anymore. It is not a really big deal but deleting this folder or its content was also impossible which may be more problematic:

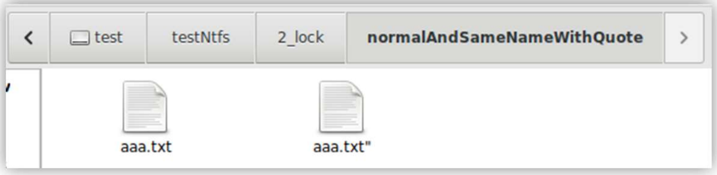
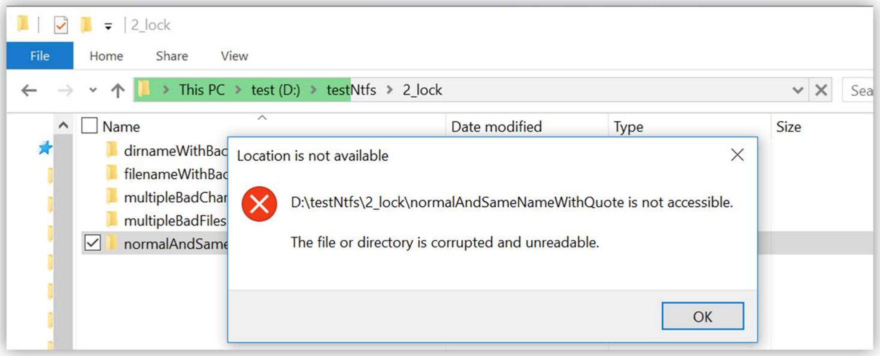


Figure 1 – A directory with « aaa.txt » and « aaa.txt" »



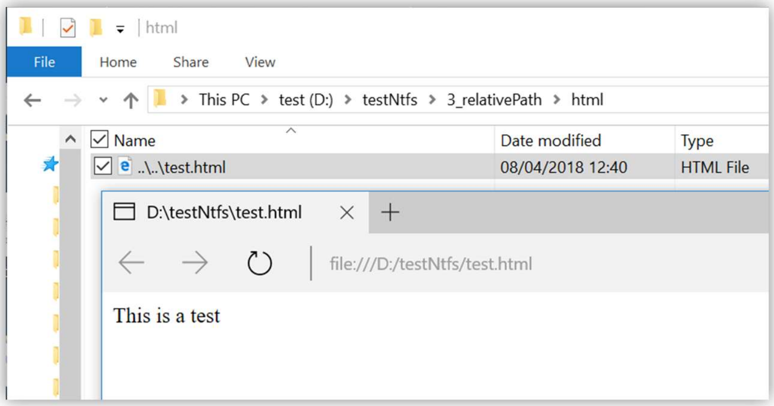
Location is not available

D:\testNtfs\2_lock\normalAndSameNameWithQuote is not accessible.

The file or directory is corrupted and unreadable.

Figure 2 – Cannot access or delete the folder

As it is possible to create filenames with the « \ » characters, an USB drive may contain files with relative path in their names in order to fool software who would try to open them:



Name	Date modified	Type
..\..\test.html	08/04/2018 12:40	HTML File

file:///D:/testNtfs/test.html

This is a test

Figure 3 – The file opened by the browser is D:/testNtfs/test.html

Parameter injections

The most important effect found while doing these tests is related to the « " » character. It is possible to create files with such character in their name and to let another user try to open them.

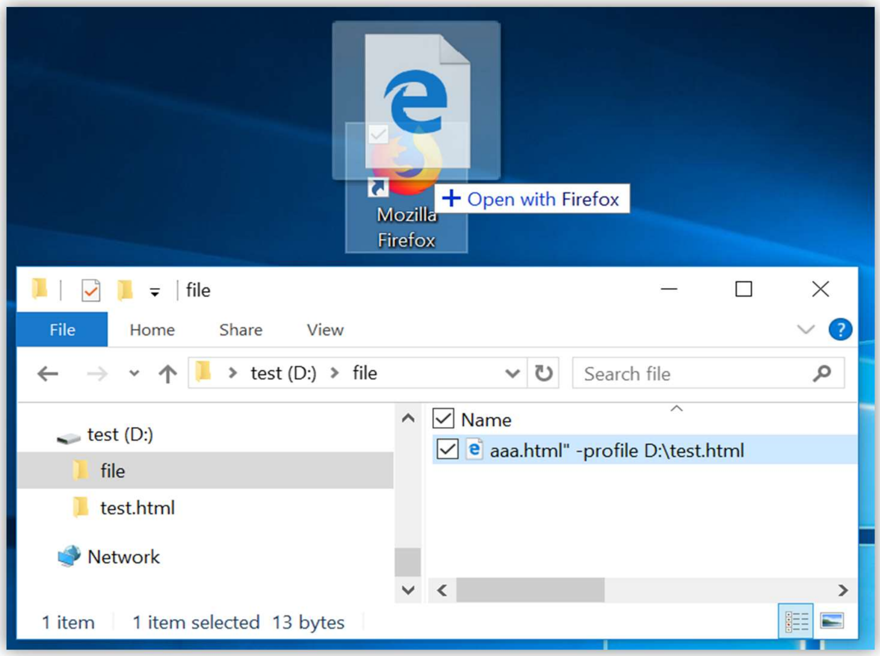
In several case, like in the following example, the software used to open the files seems to be launched by executing the command:

```
"C:\Path\to\the\software.exe" "filename.txt"
```

An attacker able to create a file with quotes in its name can use them to « close » the filename and to add additional parameters to the command in order to modify its behavior. For example, opening a file named « aaa.html" –profile D:\test.html » with the previous command would execute:

```
"C:\Path\to\the\software.exe" "aaa.html" –profile D:\test.html"
```

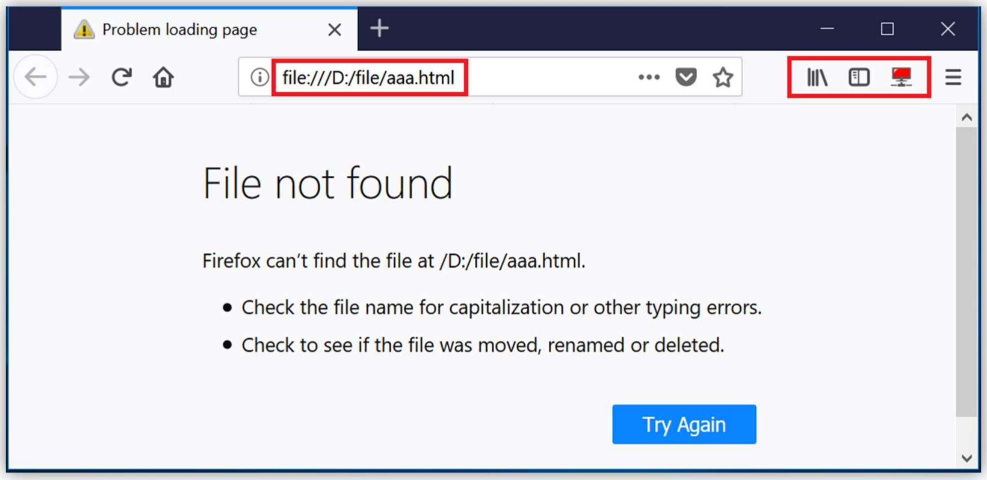
When trying to open this document with a drag&drop on Firefox, the browser tries to open the document « aaa.html » and uses the user profile « test.html » also located on the USB drive. Configuration options and add-ons of this profile are loaded and executed:



aaa.html" -profile D:\test.html

1 item   1 item selected 13 bytes

Figure 4 – Opening the file with Firefox



File not found

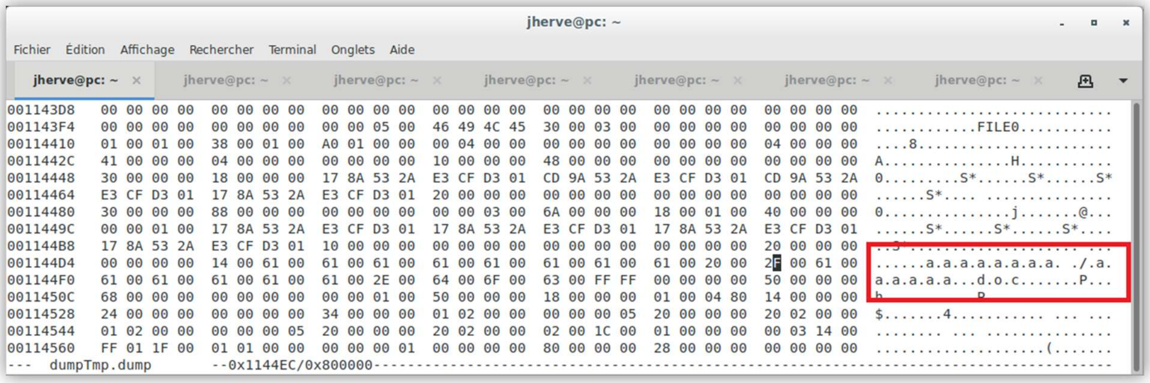
Firefox can't find the file at /D:/file/aaa.html.

- Check the file name for capitalization or other typing errors.
- Check to see if the file was moved, renamed or deleted.

Figure 5 – Firefox load the HTML page and user profile

Depending of the options accepted by the software, it may be possible to modify its configuration, the load malicious files, to execute commands…

As Unix systems also forbid the use of « / » in filenames, inserting such character is a bit more complex but can be done by modifying the filesystem using an hexadecimal editor. Combining this character with « " » also to inject options at « Windows » format (/h), to use URLs…



```
001143D8   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ............................
001143F4   00 00 00 00  00 00 00 00  00 00 05 00  46 49 4C 45  30 00 03 00  00 00 00 00  00 00 00 00  ............FILE0...........
00114410   01 00 01 00  38 00 01 00  A0 01 00 00  00 04 00 00  00 00 00 00  00 00 00 00  04 00 00 00  ....8.......................
0011442C   41 00 00 00  04 00 00 00  00 00 00 00  10 00 00 00  48 00 00 00  00 00 00 00  00 00 00 00  A...............H...........
00114448   30 00 00 00  18 00 00 00  17 8A 53 2A  E3 CF D3 01  CD 9A 53 2A  E3 CF D3 01  CD 9A 53 2A  0.........S*......S*......S*
00114464   E3 CF D3 01  17 8A 53 2A  E3 CF D3 01  20 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ......S*.... ...............
00114480   30 00 00 00  88 00 00 00  00 00 00 00  00 00 03 00  6A 00 00 00  18 00 01 00  40 00 00 00  0...............j.......@...
0011449C   00 00 01 00  17 8A 53 2A  E3 CF D3 01  17 8A 53 2A  E3 CF D3 01  17 8A 53 2A  E3 CF D3 01  ......S*......S*......S*....
001144B8   17 8A 53 2A  E3 CF D3 01  10 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  20 00 00 00  ..S*.................... ...
001144D4   00 00 00 00  14 00 61 00  61 00 61 00  61 00 61 00  61 00 61 00  61 00 20 00  2F 00 61 00  ......a.a.a.a.a.a.a.a. ./.a.
001144F0   61 00 61 00  61 00 61 00  61 00 2E 00  64 00 6F 00  63 00 FF FF  00 00 00 00  50 00 00 00  a.a.a.a.a...d.o.c.......P...
0011450C   68 00 00 00  00 00 00 00  00 00 01 00  50 00 00 00  18 00 00 00  01 00 04 80  14 00 00 00  h.........P.................
00114528   24 00 00 00  00 00 00 00  34 00 00 00  01 02 00 00  00 00 00 05  20 00 00 00  20 02 00 00  $.......4........... ... ...
00114544   01 02 00 00  00 00 00 05  20 00 00 00  20 02 00 00  02 00 1C 00  01 00 00 00  00 03 14 00  ........ ... ...............
00114560   FF 01 1F 00  01 01 00 00  00 00 00 01  00 00 00 00  80 00 00 00  28 00 00 00  00 00 00 00  ....................(.......
---  dumpTmp.dump       --0x1144EC/0x800000---------------------------------------------------------------------------
```

Figure 6 – Modifying a filename with an hexadecimal editor

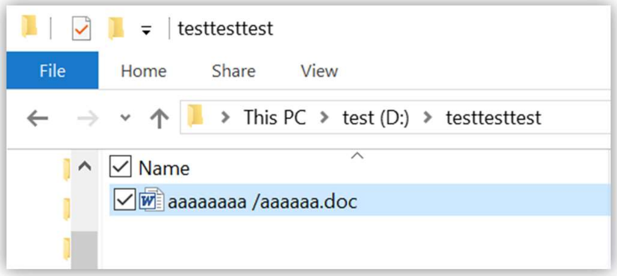


Figure 7 – Filename with a « / » character

As a conclusion, Microsoft acknowledged this finding but decided not to address the issue:

> The engineering team has finished their investigation and determined that due to the many steps required, it does not represent a vulnerability which would be easily exploited by an attacker. As such it doesn't meet the bar for servicing in a security update, although the team may decide to address it in a future version of Windows.
>
> We consider the matter resolved and appreciate you bringing it to our attention; if you have any information which could affect our assessment please let me know and I will be happy to have the team take a second look.