SEC Consult
an Eviden business

Pentester’S Windows NTFS Tricks Collection

12.06.2018

In this blog post René Freingruber (@ReneFreingruber) from the SEC Consult Vulnerability Lab shares different filesystem tricks which were collected over the last years from various blog posts or found by himself.

These tricks don’t lead to a directly exploitable condition, however, they can indirectly lead to exploitable flaws in special situations because of the non-intuitive behavior. Please note that only information on how to exploit the problems together with the impact will be explained (without technical low level details on the Windows API), as this would go beyond the scope of the article.

SEC Consult
an Eviden business

Trick 1: Create Folders Without Permissions (Cve-2018-1036/Ntfs Eop)

On Windows you can assign "special permissions" to folders like permissions that a user is allowed to create files in a folder, but is not allowed to create folders.

One example of such a folder is C:\Windows\Tasks\ where users can create files, but are not allowed to create folders:

Moreover, it's possible that an administrator or a program configures such permissions and assumes that users are really not allowed to create folders in it.

This ACL can be bypassed as soon as a user can create files. Adding "::$INDEX_ALLOCATION" to the end of a
will create a folder instead of a file and Windows currently doesn't include a check for this corner case.

As shown above, a directory was successfully created and the user can create arbitrary files or folders in this directory (which can lead to privilege escalation if an administrator/program assumes that this is not possible because of the missing permissions).

Side note: The ::$INDEX_ALLOCATION trick can also be used to delete directories if an application just allows to delete files.

Contact timeline:

	2018-03-01:	Initial contact via secure@microsoft.com. Sending PGP encrypted blogpost.

2018-03-05:	Microsoft asks to extend the disclosure deadline.
2018-03-05:	Agreed upon the extended 90 days deadline.
2018-03-27:	Asked Microsoft if the vulnerabilities could be reproduced and when a patch will be available.
2018-03-28:	Microsoft confirmed that the "trick 1" vulnerability was successfully reproduced and assigned CVE-2018-1036. The vulnerability will be fixed by 5B (May Patch Tuesday).
2018-04-27:	Microsoft contacted us that the proposed fix lead to a regression. Microsoft asked to extend the deadline to 2018-06-12 (June Patch Tuesday).
2018-04-30:	Informed Microsoft that the deadline gets extended to 2018-06-12. Asked Microsoft if the other tricks will also be patched.
2018-04-30:	Microsoft responded that the other tricks "will not receive a downlevel security update".
2018-06-12:	Microsoft releases the patch .
2018-06-13:	Blog post release.

Trick 2: Bypass Path Restrictions With Alternate Data Streams

You maybe wonder why the above technique worked. Basically files on a NTFS volume are stored in the form

<filename>:<stream-name>:<type>

If we create a file named test.txt it will internally be stored as test.txt::$DATA because the stream name is empty and $DATA is the default type. The first trick abused the fact that the type can be changed to INDEX_ALLOCATION which corresponds to the directory type and therefore creates a directory.

write to "test.txt:foo" or to "test.txt:foo:$DATA" (both are equal because $DATA is the default type). Different stream names are for example used to store the origin of a file. If you download a file from the internet (or receive it via e-mail), Windows silently adds a Zone Identifier via a stream name (then it can show an additional warning dialog if you want to execute it). For example, if we download "putty.exe", Windows also creates "putty.exe:Zone.Identifier:$DATA". These stream names can be made visible via the /r switch with the dir command:

As you can see, the Zone Identifier can't be read via the type command (with the more command it would wor
also important to omit the $DATA type if we read the file with notepad. The important message is that we ca
data in an ADS (including applications!). For example, putty can be copied to an ADS and then be invoked via
(direct invocation is not possible):

Side note: This article was written on 2018-03-01 and reported to Microsoft. In the meantime Microsoft Windows Defender was updated to detect WMIC process invocations.

You maybe ask yourself why someone would do this? First of all, ADS can be used to hide data (dir command without the /r switch will not display them; explorer.exe will also not show them; We will later see how we can even hide from the /r dir switch…). However, ADS has another great property – we can add an ADS to a folder. To be allowed to do this we must own the "create folders" permissions on the directory (and the folder name must not be a number). The important fact is that an ADS on a folder looks like a file from the parent folder!

For example, on Windows a normal user can't create files in C:\Windows\ (only admins can write to this folder). So it's possible that applications assume that files in C:\Windows\ can be trusted because only admins can create such files. However, C:\Windows\Tracing is a folder in which normal users can create files and folders – a normal user can therefore create an ADS on this folder.

Let's say the user writes to the file C:\Windows\Tracing:test.dll. If this path is now passed to a Windows API which calculates the base folder, this API will start at the end of the path and go backward until the first "\" is found. Then everything left from "\" will be returned as the base folder. For C:\Windows\Tracing:test.dll this will return C:\W as base folder, however, as already mentioned, a normal user is not allowed to create files in this folder but us trick we created a file which looks like it is stored in C:\Windows!

Here is the output of different Windows functions which calculate the base folder (we can see that it's always C:\windows):

Side note: The above stored dll can be started with the Windows built-in control.exe application with the command: control.exe C:\Windows\tracing:foobar.dll

This behavior can be used to bypass some application whitelisting solutions but can also bypass security checks in various situations where the programmer assumed that it's enough to check if a file is stored in a specific base folder and assumes that only admins can write to this folder because of the set ACL.

For example, consider that an application allows to upload data and that uploaded data is stored in applicationFolder\uploadedData\. Moreover, the application allows to start scripts / applications from applicationFolder\ but not from applicationFolder\uploadedData\ (with a blacklist approach). If the user uploads a file named ":foo.ps1", the system will create an ADS like applicationFolder\uploadedData:foo.ps1 and this file appears to be stored inside applicationFolder\ and therefore bypassing the security checks.

Another interesting fact is that ADS names can contain symbols which are normally forbidden for filenames like " or * (you have to create these files using the native Windows API; cmd.exe filteres these characters):

This on it's own can lead to several problems (e.g.: if the filename ends with " and the path is enclosed by " as mentioned by James Forshaw in his blog; see the references section). However, another interesting attack vector can be XSS (or command injection). Let's assume that a website runs on IIS and allows to upload files and is prone to CSRF. After uploading the file a success dialog is shown including the filename. If the filename is not sanitized, this could lead to XSS, however, filenames are not allowed to contain symbols such as < or > (and we can therefore not execute JavaScript code with it). However, an ADS is allowed to contain these symbols, therefore an attacker could try to send an upload request for a filename with ADS:

Trick 3: Create Files Which Can'T Be Found By Using The "..." Folder

Every folder contains per default two special entries, namely the directory "." which refers to the current directory and ".." which referes to the parent directory. On Windows it's not possible to create files / folders with just dots in the name, most likely to prevent attacks which confuse the parsers with the dots.

The screenshot above shows that it's not possible to create a "..." or "...." folder. However, this can be bypassed with the above mentioned ::$INDEX_ALLOCATION trick:

The "..." folder was created with the above mentioned trick, however, such folders can also be created by passing the name twice as shown in the "...." example (mkdir "....\....\" creates the directory "....", but also a directory "...." in it. Just passing mkdir "....\xyz\" doesn't work.).

As you can see, you can't enter the folder with just the name (e.g.: "cd …" or "cd …\" or "cd …\…" is not working), so you really have to use the syntax "cd …\…\". After that you can create files in this folder. (Interesting side note: If you enter "cd ." in this folder, you will go one directory up, because path's are confused).

Moreover, it's also not possible to open this directory from the GUI (explorer.exe). I encountered two different situations. In some cases double clicking such a folder has no impact (you stay in the current directory and the path keeps the same), in other cases you stay in the folder but the path in the explorer changes. For example, after the folder 17 times it looks like this (notice the "…" dirs in the path):

You can try to enter the folder as often as you want, you will not see the files in the folder in the GUI. It's also not possible to open the folder by passing "C:\test\...\...\" in the path input field in the above image.

(Side note: If you try to delete this folder from the GUI, explorer.exe will crash; You will see a dialog where windows is counting files in the folder and where it counts "a lot of files"; Maybe it's better if you don't try this on your working system...).

Searching for files in this folder via the GUI (explorer.exe) also doesn't work, for example, if you search for the "123.txt" file with the GUI it will hang/search forever, without actually finding the files.

Please note that searching via cmd works without a problem:

Output is truncated because the command will print the two directories forever…).

A search for "123.txt" (E.g.: with "Get-ChildItem -Path C:\test -Filter 123.txt -Recurse -ErrorAction SilentlyContinue -Force") will therefore never find the file (and will never end).

I also tested this with different AntiVirus products, these seem to work correctly (I placed malware samples in this directory and the tested AntiVirus solutions found them). Some of them were still confused by the path, e.g.: when searching for viruses inside "C:\test\…\" they searched in "C:\test\" instead. Also python code with os.walk() seems to work correctly.

Please note that just creating a directory junction which points to it's own parent folder doesn't lead to an endless loop in cmd or Powershell.

Trick 4: "Hide" The Destination Of A Directory Junction

Directory junctions are a very useful NTFS feature to find security vulnerabilities for attackers. Using it you can create (with normal user privileges) a symbolic link to a destination folder.

that the attacker removes the folder x and replaces it with a directory junction named "x" which points to C:\windows\System32\. If the attacker now clicks on the "restore" button the AntiVirus solution will copy the file to the x folder which now points to system32 with SYSTEM privileges (which directly leads to EoP).

Directory junctions can often be abused if the targeted application contains race condition vulnerabilities (TOCTOU vulnerabilities – time of check time of use).

A directory junction can be created with the mklink utility together with the /J argument. It's possible to combine this with the ::$INDEX_ALLOCATION trick to create a directory junction with the name "…":

The first directory junction "test1" was created with a normal name and therefore the destination is correctly shown in the "dir" output. However, in the case of the "…" directory junction, the target is no longer displayed (instead […] is shown; see the red box). Please also note that you can let junction1 point to junction2 which points to junction3 and so on until the last one points to the actual destination.

The dir command prints files from the system32 folder (red marked; please also note that the first command created the hello.bat file in C:\test8\).

C:\test\hello.bat (green box) and not C:\Windows\System32\hello.bat. I'm not sure if this has a direct impact on the security since you can start files in any folder anyway, however, it could maybe be used to bypass application whitelisting solutions with whitelisted script files.

Trick 5: Hide Alternate Data Streams

As already discussed it's possible to dump ADS via the /r switch in the dir command. Moreover, streams.exe is a tool from Sysinternals which can also dump the streams:

Please note that you can also create an ADS on the drive like "echo 123 > C:\:abc.txt". This will hide from the "dir /r" command inside C:\. However, it will show the ADS inside subfolders of C:\ for the ".." directory. For example:

The red marked ADS was created by the C:\:abc.txt ADS. This ADS is also visible via Sysinternals tool stream it's called directly on C:\. Therefore, to hide from both tools the "..." trick should be used.

There exists a second trick which can be used to hide from the tools. On Windows you can add ".<spaces>." at the end of a file and Windows will automatically remove it (canonicalization removes it).

However, we can create such a file with an ADS! The funny property of such a file is that tools will not be able to open the file, because a path like "xyz. ." will automatically be changed to "xyz" and this file doesn't exist.

Here is the proof:

We can also create a directory with the name ". ." like this:

Then it's not possible to enter this folder:

If we can add spaces in between of a directory name, we can also add it at the end like "b " or ".. " or ". ".

Explanation: There is a "b" and a "b " folder, a file named "a" and a file named "a ", the two default dirs "." and ".." plus the ". " and ".. " ones and the ". ." dir.

Directories with the name ".. " can be entered with our already discussed technique:

Side note 1: This folder can be opened via the GUI if you click twice on the folder, also the content of the folder will be displayed correctly. However, files in it can't be opened because of the wrong path (explorer.exe uses C:\test22\.. \.. \123.txt instead of C:\test22\.. \123.txt). Powershell will again be stuck in an endless loop when searching such folders. Side note 2: You can also create an ADS on a folder with a name such as "abc". Then you can rename the folder to a name just containing numbers (e.g. "1"). After that you can still see the ADS, but you can't open it (ADS on a folder with a number as name doesn't work). To open the ADS data you have to rename the folder first back to for example "abc".

Filesystem tricks vs. AntiVirus products / Forensic software:

I did a quick verification of the above mentioned tricks against AntiVirus products to verify if these can catch malware which abuses the tricks. The most noteworthy finding was with files / folders ending with ". .". For example, I stored the eicar test virus in a folder and copied it with the following commands:

```
echo 123 > "foo. .::INDEX_ALLOCATION"
cd "foo. .::$INDEX_ALLOCATION"
copy ..\eicar.com .
copy ..\eicar.com .\eicar
```

After that I re-enabled the AntiVirus solutions and scanned the folder. All AntiVirus solutions just identified "eicar.com" and "tester" in this folder, but not the eicar virus in "123. ." or in the two files in the "foo. ." folder. However, when this folder is entered and the files are started, the AntiVirus products found them (because the content is loaded from the file system to memory). The "remove" action from Windows Defender could not remove the files and has therefore no impact, however, the "remove" action from for example Emsisoft could remove the test virus in the folder. Emsisoft just removed the "eicar.com" file in the "foo. ." folder, the "eicar" file was not removed and the content can be read without a problem (Emsisoft responded to us that only files which are mapped as executable are scanned with the exception of some specific file extensions like .com. This behavior can be altered in the file guard settings by switching to "Thorough" to also scan at file reads; Windows Defender on the other hand also blocked reading the "eicar" text file). I also conducted a short test against Autopsy 4.6.0 (a free forensic software) by loading "logical files" into the tool (from the running system; not a disk image). The "..." folder can be entered, however, the "foo. ." folder can't. Moreover, I created a file named "valid" with the content "valid" and a file called "valid. ." with the content "secret". Autopsy shows for both files the content "valid" (and never the "secret" content). In addition to that, the ".. " folder (with a space at the end) is intepreted as ".." and therefore goes one directory up at double click. This only applies to the "logical files" mode, in disk image (raw) mode, everything is displayed correctly (in the live mode Autopsy is using the Windows API to access the data and therefore problems occur).

Trick 6: Hiding The Process Binary

As already discussed above: Windows automatically removes ". ." at the end of a file. What if we can somehow start a process with a name like "file1. ."? Well, then it can happen that checks (e.g.: signature checks from AntiVirus products) are maybe performed on "file1" instead. Let's try this:

We created 3 files:

- "file" with the Microsoft signature from taskmgr
- "file. ." which is our "fake malware" which should be hidden but executed
- "filex x" which contains the signature from WinSCP. This file will become important later.

We now need a way to start a process from the "file. ." binary which is not a trivial task because all Microsoft Windows API calls automatically remove the ". ." from the filename and would start "file" (taskmgr) instead. To deal with this problem we use the following code:

The above code just calls CreateProcessA to create a process from "filex x" (WinSCP). If we compile this application and start it, WinSCP will be started. However, we are not going to start it normally. Instead, we start the application inside a debugger (e.g.: WinDbg). Now we set a breakpoint at the function which makes the associated system call: "bp ntdll!NtCreateUserProcess". With "g" (go) we can start our program in the debugger and hit the breakpoint. At the breakpoint the current stack can be dumped ("dq rsp"). The 12th pointer on the stack is important and should be dumped. The 4th value at this address is the pointer to our file name.

normalization already happend, this value can now be modified. Let's overwrite the "x" characters with "." (command "eb" for edit bytes):

After that just continue execution with "g". Guess what will happen?

Correct, "file. ." (the malware) gets executed. However, if a user right clicks the process in the task manager and selects "properties" the properties of "file" (taskmgr) with the valid Microsoft signature will be shown.

But what is with "filex x" (WinSCP)? Yes, this file gets also shown as the running process namely in process explorer (because the path was set before NtCreateUserProcess was called):

And what is with powershell? Yes, also the wrong binary:

Is this a problem? Well, it depends. First of all, an attacker can start a process (the malware), rename / remove it afterwards and then rename a valid file to the same name. Then the above effects in taskmanager and process explorer will also occur. However, the difference is that with the above mentioned trick this happens exactly at the same time when the process gets launched.

For example, consider that the installed endpoint protection checks for every started process if the binary has already known in the cloud. With this trick the endpoint protection may use the wrong binary to verify if the ha already known. Please also note that a debugger is not required to create such processes. An application can just hook the NtCreateUserProcess function and implement the modifications in the hook.

Windows CMD Tricks:

is the same as "ca^l^c". It's just important that ^ is not the last symbol and that two ^ symbols are not used after each other. Instead of ^ the double quote can also be used and this has no restrictions (it can be the last character or used multiple times). For example, ^ca^"^"^lc^" will start the calculator.

The same applies to zero-length environment variables. An environment variable can be accessed via %name%. If the environment variable has a length of zero, "cal%name%c" would be the same as "calc". Since environment variables don't have per default a length of zero, this can't be used directly. However, it's possible to call substring on the environment variable with a special syntax (:~start,end). The following figure shows the "windir" environment variable and how substring can be used with negativ values to get a zero-length variable returned:

The following figure shows a combination of these techniques to hide that Powershell is started in version 2 (which was a long time helpful but should not be done anymore on latest Windows 10):

You can see the use of ^ and the environment variable trick (%os:~0,-56%), but also that the version "00000000002.0000" (instead of just 2) is used and that the argument is "?ver" and not "-ver" (note, this is not a normal ? symbol, it's U+2015; just using ? would not work).

On Windows "/" can also be used in paths instead of "\". For example, C:\Windows/\//\system32\calc.exe is the same as C:\Windows\system32\calc.exe. Moreover, you can also access the binary via the UNC path to avoide the "C:\" pattern: \\127.0.0.1\C$\windows\system32\calc.exe

Similiar tricks can often be used to defeat blacklist approaches (e.g. if powershell.exe is forbidden, an attacker can call power^shell.exe to bypass the restriction. Or if calc is forbidden, you can execute:

^"%Localappdata:~-3%^%SystemRoot:~0,1%^"

to start calc.exe and so on).

This blog post was written by René Freingruber (@ReneFreingruber) on behalf of SEC Consult Vulnerability Lab. Some of the listed techniques were already documented by James Forshaw (@tiraniddo). Alex Inführ (@insertScript) documented some of the ADS hiding tricks in his blog post. Please see the following references-section for a full list of previous work. SEC Consult is always searching for talented security professionals to work in our team. More information can be found ***here****.*

REFERENCES:

https://msdn.microsoft.com/en-us/library/dn393272.aspx

https://tyranidslair.blogspot.co.at/2014/05/abusive-directory-syndrome.html

https://tyranidslair.blogspot.co.at/2014/06/addictive-double-quoting-sickness.html

https://googleprojectzero.blogspot.co.at/2016/02/the-definitive-guide-on-win32-to-nt.html

https://googleprojectzero.blogspot.co.at/2015/12/between-rock-and-hard-link.html

https://googleprojectzero.blogspot.co.at/2015/08/windows-10hh-symbolic-link-mitigations.html

http://insert-script.blogspot.co.at/2012/11/hidden-alternative-data-streams.html

Edit history:

2018-06-13: Soroush Dalili pointed out on Twitter that he used the INDEX_ALLOCATION trick already in 2010 to bypass authentication checks in IIS5.1 and a similiar trick with ".." folders. More information can be found in the following two references:

https://soroush.secproject.com/blog/2010/07/iis5-1-directory-authentication-bypass-by-using-i30index_allocation/

https://soroush.secproject.com/blog/2010/12/a-dotty-salty-directory-a-secret-place-in-ntfs-for-secret-files/

2018-06-14: Daniel Bohannon (@danielhbohannon) developed Invoke-DOSfucation, a tool which can be used for obfuscation in cmd.exe.

https://github.com/danielbohannon/Invoke-DOSfuscation

https://www.fireeye.com/blog/threat-research/2018/03/dosfuscation-exploring-obfuscation-and-detection-techniques.html

2018-06-14: Oddvar Moe (@Oddvarmoe) listed more techniques to execute applications / scripts from ADS:

https://gist.github.com/api0cradle/cdd2d0d0ec9abb686f0e89306e277b8f

https://oddvar.moe/2018/01/14/putting-data-in-alternate-data-streams-and-how-to-execute-it/

https://oddvar.moe/2018/04/11/putting-data-in-alternate-data-streams-and-how-to-execute-it-part-2/

Back

SEC Consult
an Eviden business

Jobs

security. The company specializes in information security management, NIS security audits, penetration testing, ISO 27001 certification support, Cyber Defence and secure software certification. SEC Consult is part of Eviden.

Incident?