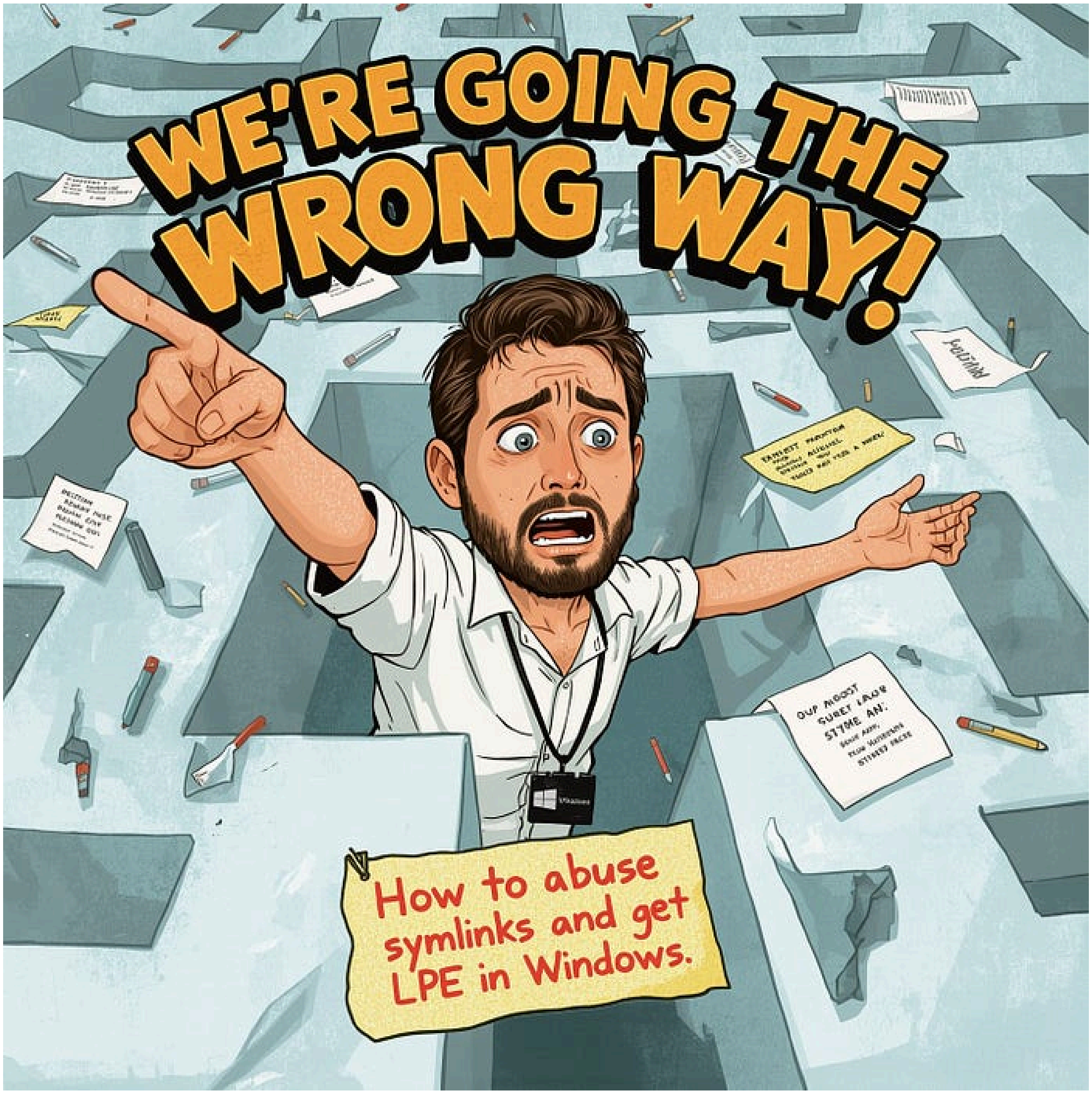
We’re going the wrong way! How to abuse symlinks and get LPE in Windows

cicada-8.medium.com/were-going-the-wrong-way-how-to-abuse-symlinks-and-get-lpe-in-windows-0c598b99125b

CICADA8 June 25, 2025


WE'RE GOING THE WRONG WAY!
How to abuse symlinks and get LPE in Windows.

Hello everybody, my name is Michael Zhmailo and I am a penetration testing expert in the CICADA8 team.

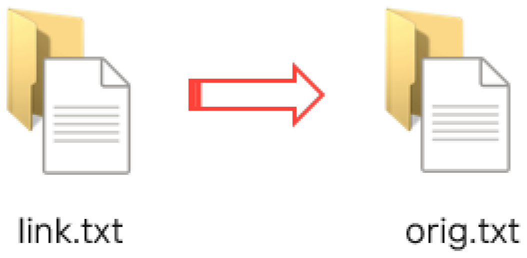
Symbolic links have been present in Windows systems almost since birth. However, few offensive security courses will teach you about them, although symbolic links have great potential, because with luck you can get LPE! My article will tell you in detail about symbolic links, the specifics of working with them, and will also clearly show you the logic of abuse to obtain LPE.

What is a symbolic link?

So, a symbolic link allows you to point from one object to another. Literally: a symbolic link *example* could point to file *1.txt.* There are different types of symbolic links in Windows. Let's take a closer look at them.

> NTFS (Soft Link) — Allows you to link from one file to another. To create, you need administrator rights or the privilege or Windows Developer Mode enabled. U can create such link using:

```
mklink link.txt orig.txt;
```



Soft Link Example

> Hard Link — also allows you to link from one file to another, but only within the same drive. Requires FILE_WRITE_ATTRIBUTES rights to create. You cannot link to directories. U can read more about soft and hard links here. U can create hard link with:

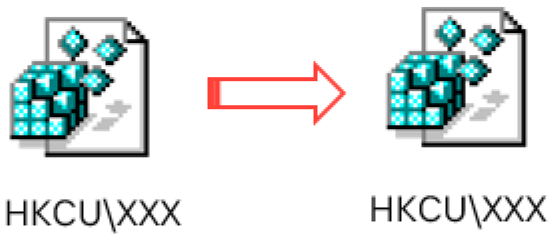
```
mklink /H link.txt orig.txt/CommonUtils/Hardlink.cpp
```

C:\link.txt
C:\hi\orig.txt

Hard link example

> Registry Link — allows you to link from one registry key to another, requires KEY_CREATE_LINK+KEY_CREATE_SUB_KEY rights to create, also you cannot create a symlink between certain hives. For example, you cannot create a symlink from to

```
/CommonUtils/RegistrySymlink.cpp
```



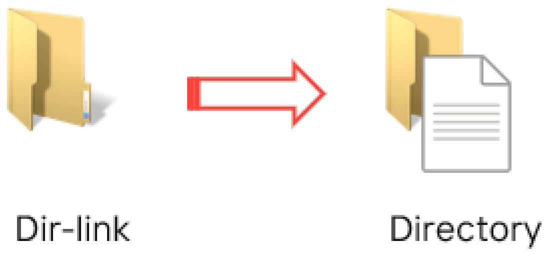
Registry Symlink example

As you can see, almost all symbolic links require a fairly privileged account. How can you elevate your privileges then?

And here we come to two other less popular types of symlinks.

> NTFS Mount Point — This is a symbolic link from one folder to another. Can be created on behalf of a low-privileged account. The directory that becomes a symbolic link (in the examples) must be empty, and we must have write access to it. U can create such link using:

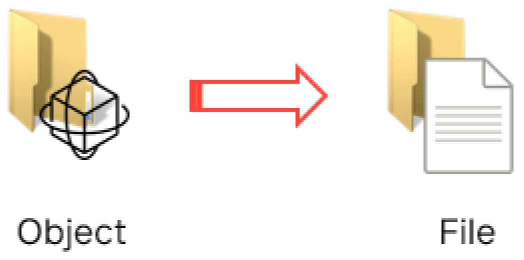
```
mklink /J Dir-link Directory/CreateMountPoint
```



NTFS Mount Point Example

> Object Manager Symlink — allows you to create symbolic links within the Object Manager namespace. As a low-privileged user, you can create symbolic links in the and namespaces. You can also try creating a symbolic link inside if you meet some conditions described here.

```
/NativeSymlink/CreateDosDeviceSymlink
```

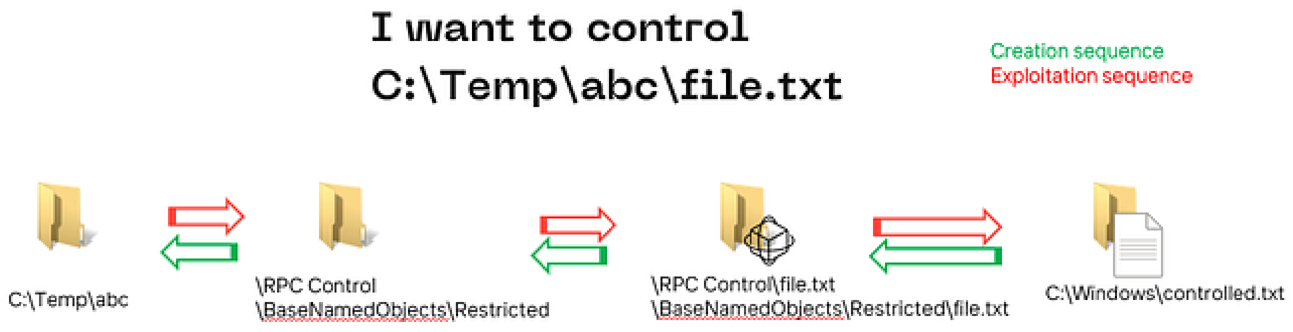


Object manager symlink example

U can read more about Object Manager here. To achieve LPE we will use these two types of symbolic links.

Arbitrary File Deletion Example

So, let's look at an example. Let's say we know that some operation which can lead to LPE is being performed on a file, and we want to replace it with a symbolic link.



Using NTFS Mount Point + Object Manager symlink example

So, let's say there is some Windows service that performs the operation of deleting the file *C:\Temp\abc\file.txt*. And we want to replace this file with *C:\Windows\controlled.txt*, while working on behalf of a low-privileged account. In addition, we have full rights to the directory *C:\Temp\abc* and to the *file.txt* file itself. We have no rights to the file controlled.txt.

In this case, we proceed as follows:

1. Create a Object Manager symbolic link from to

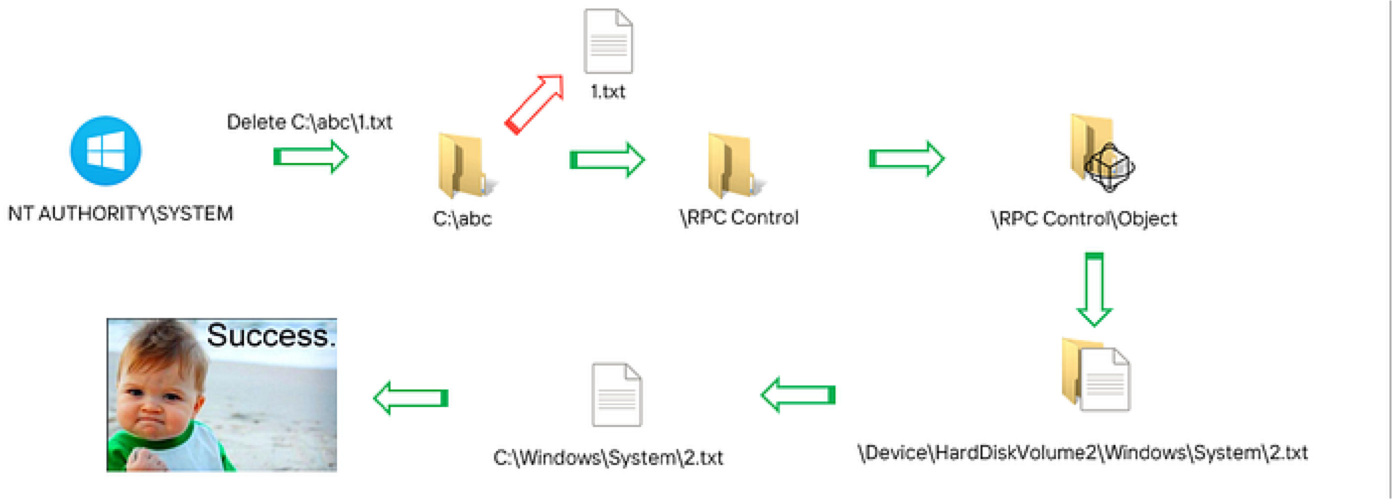
2. After that, we create an NTFS Mount Point from *C:\Temp\abc* (dont forget about deleting all files from *C:\Temp\abc*) to \RPC Control;

3. We see successful deletion of *C:\Windows\controlled.txt*!

How does it work? So, first, the high-privileged service that performs the delete operation calls a method like DeleteFile() on the file *C:\Temp\abc\file.txt*. However, this file does not exist. And *C:\Temp\abc* points to *\RPC Control*. So, *\RPC Control\file.txt* is accessed. This is also a symbolic link. Only it points to *C:\Windows\controlled.txt*. This causes the service to follow two symbolic links and instead of the desired *C:\Temp\abc\file.txt*, it deletes *C:\Windows\controlled.txt*. This is what the vulnerability looks like :) It's called **Arbitrary File Deletion**.

More arbitrary file operations

So, the general logic of exploitation is as follows. We create two symbolic links, after which we redirect the execution flow of the privileged service, forcing it to perform some operations against the file we need.



Abuse logic

You can see examples of Arbitrary File Delete on these CVEs:

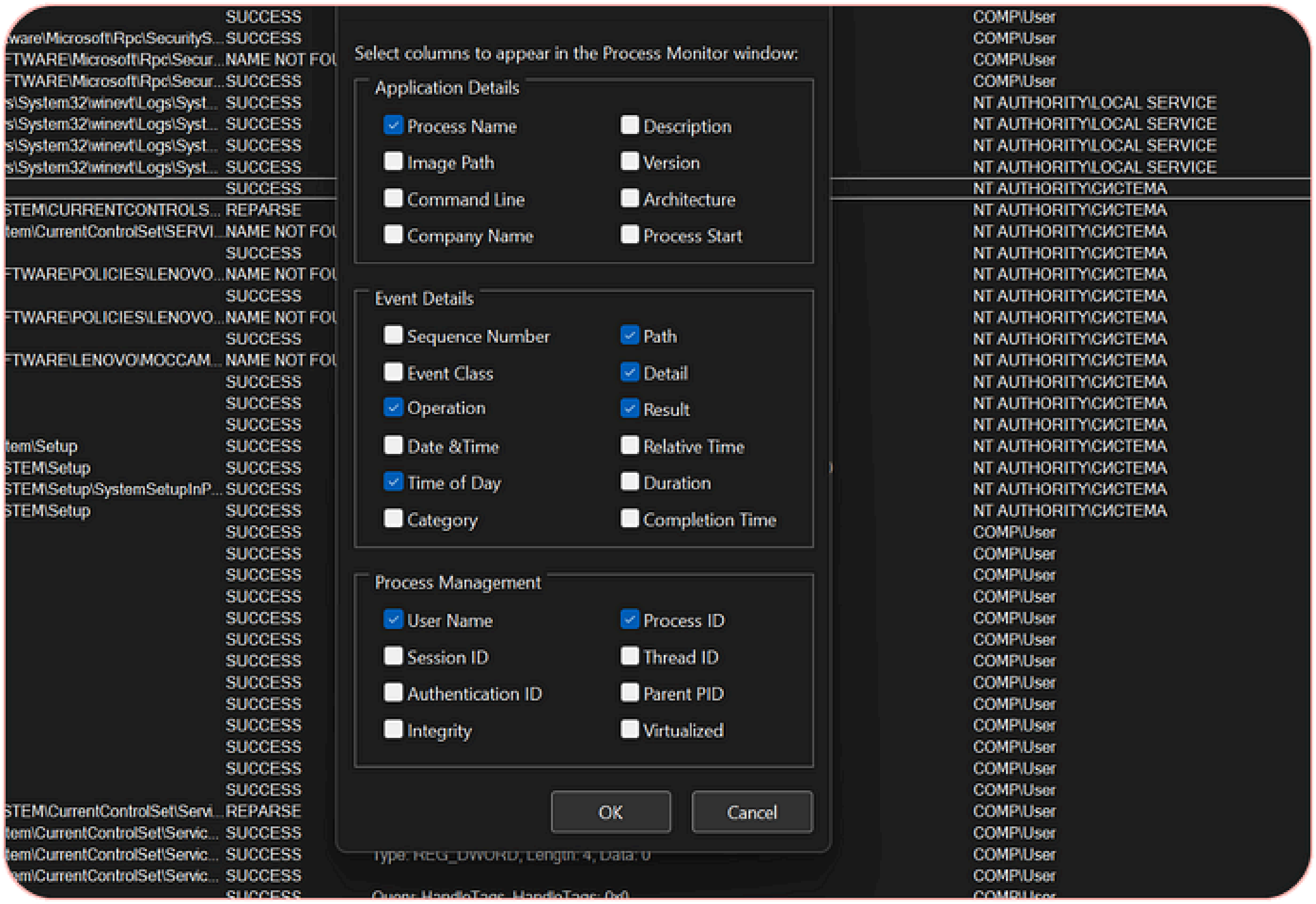
- CVE-2020–0683 — Windows MSI "Installer service" Elevation of Privilege;
- CVE-2023–21800 — Windows Installer EOP;
- CVE-2024–29404 — Razer EOP.

However, in addition to deletion, you may encounter copying operations, creating files, moving files, overwriting. I've put together the following POC to demonstrate these scenarios.

- CVE-2024–26238 — Windows 10 PlugScheduler EOP. Arbitrary Create;
- CVE-2024–12754 — AnyDesk LPE. Arbitrary Copy;
- CVE-2024–37726 — MSI Center Arbitrary File Overwrite Vulnerability;
- CVE-2024–21111 — Oracle VirtualBox LPE;
- CVE-2020–1076 — Arbitrary file write in VaultSvc.

How to find this vulnerability?

The easiest way to find such a vulnerability is to use Process Monitor. To do this, set up the settings as I have, then monitor file operations and check which files you can replace with a symbolic link.



Select columns to appear in the Process Monitor window:

Application Details
- [✓] Process Name
- [ ] Description
- [ ] Image Path
- [ ] Version
- [ ] Command Line
- [ ] Architecture
- [ ] Company Name
- [ ] Process Start

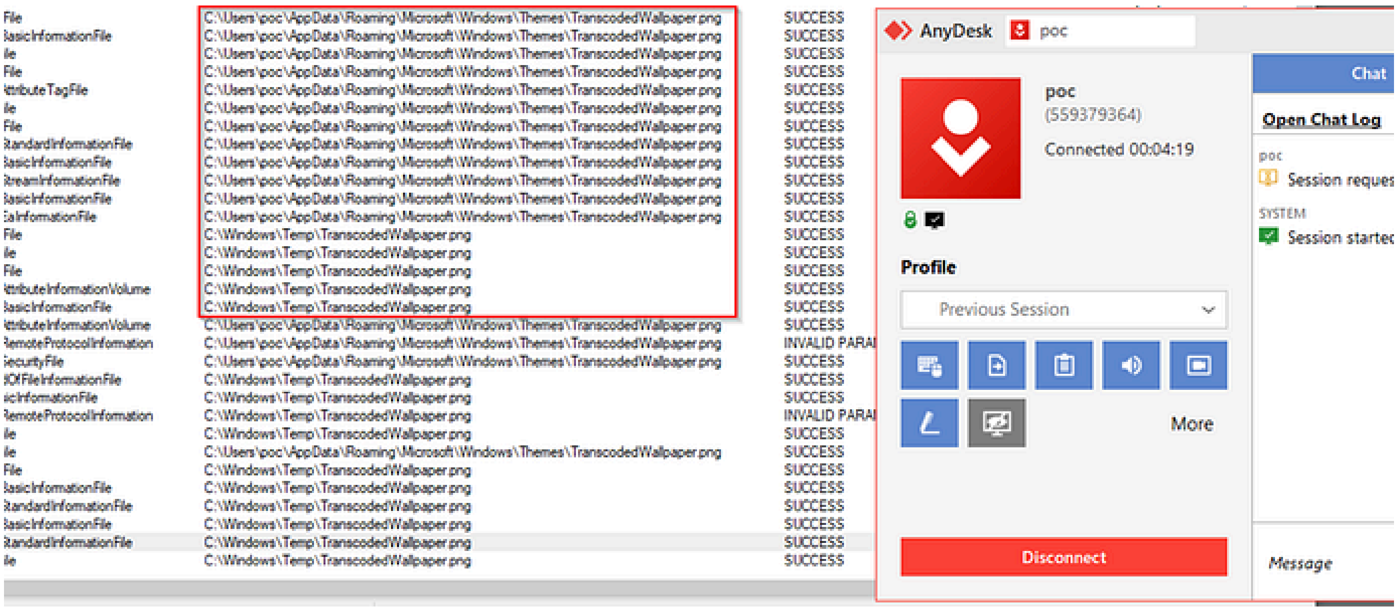
Event Details
- [ ] Sequence Number
- [✓] Path
- [ ] Event Class
- [✓] Detail
- [✓] Operation
- [✓] Result
- [ ] Date &Time
- [ ] Relative Time
- [✓] Time of Day
- [ ] Duration
- [ ] Category
- [ ] Completion Time

Process Management
- [✓] User Name
- [✓] Process ID
- [ ] Session ID
- [ ] Thread ID
- [ ] Authentication ID
- [ ] Parent PID
- [ ] Integrity
- [ ] Virtualized

```
                               SUCCESS       COMP\User
ware\Microsoft\Rpc\SecurityS...SUCCESS       COMP\User
FTWARE\Microsoft\Rpc\Secur...  NAME NOT FOU  COMP\User
FTWARE\Microsoft\Rpc\Secur...  SUCCESS       COMP\User
s\System32\winevt\Logs\Syst... SUCCESS       NT AUTHORITY\LOCAL SERVICE
s\System32\winevt\Logs\Syst... SUCCESS       NT AUTHORITY\LOCAL SERVICE
s\System32\winevt\Logs\Syst... SUCCESS       NT AUTHORITY\LOCAL SERVICE
s\System32\winevt\Logs\Syst... SUCCESS       NT AUTHORITY\LOCAL SERVICE
                               SUCCESS       NT AUTHORITY\СИСТЕМА
STEM\CURRENTCONTROLS...        REPARSE       NT AUTHORITY\СИСТЕМА
tem\CurrentControlSet\SERVI... NAME NOT FOU  NT AUTHORITY\СИСТЕМА
                               SUCCESS       NT AUTHORITY\СИСТЕМА
FTWARE\POLICIES\LENOVO...      NAME NOT FOU  NT AUTHORITY\СИСТЕМА
                               SUCCESS       NT AUTHORITY\СИСТЕМА
FTWARE\POLICIES\LENOVO...      NAME NOT FOU  NT AUTHORITY\СИСТЕМА
                               SUCCESS       NT AUTHORITY\СИСТЕМА
FTWARE\LENOVO\MOCCAM...        NAME NOT FOU  NT AUTHORITY\СИСТЕМА
                               SUCCESS       NT AUTHORITY\СИСТЕМА
                               SUCCESS       NT AUTHORITY\СИСТЕМА
                               SUCCESS       NT AUTHORITY\СИСТЕМА
tem\Setup                      SUCCESS       NT AUTHORITY\СИСТЕМА
STEM\Setup                     SUCCESS       NT AUTHORITY\СИСТЕМА
STEM\Setup\SystemSetupInP...   SUCCESS       NT AUTHORITY\СИСТЕМА
STEM\Setup                     SUCCESS       NT AUTHORITY\СИСТЕМА
                               SUCCESS       COMP\User
                               SUCCESS       COMP\User
                               SUCCESS       COMP\User
                               SUCCESS       COMP\User
                               SUCCESS       COMP\User
                               SUCCESS       COMP\User
                               SUCCESS       COMP\User
                               SUCCESS       COMP\User
                               SUCCESS       COMP\User
                               SUCCESS       COMP\User
                               SUCCESS       COMP\User
                               SUCCESS       COMP\User
                               SUCCESS       COMP\User
STEM\CurrentControlSet\Servi...REPARSE       COMP\User
tem\CurrentControlSet\Servic...SUCCESS       COMP\User
tem\CurrentControlSet\Servic...SUCCESS  Type: REG_DWORD, Length: 4, Data: 0   COMP\User
tem\CurrentControlSet\Servic...SUCCESS       COMP\User
```

Process Monitor Settings (Options -> Select Columns)

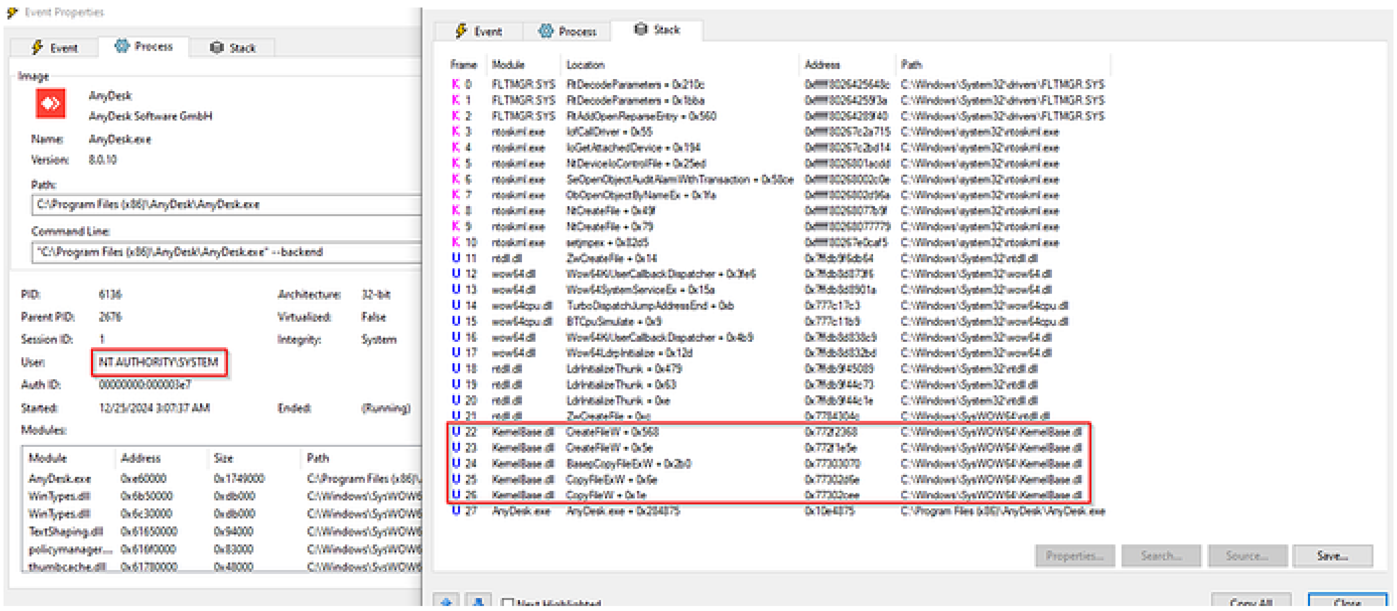
For example, when we investigated LPE in Anydesk, we determined that the operation was being used on the file *C:\users\<username>\AppData\Roaming\Microsoft\Windows\Themes\<wallpaper>.png*, over which we have full control. After which the file was copied to *C:\Windows\Temp\<wallpaper>.png.*



```
File                              C:\Users\poc\AppData\Roaming\Microsoft\Windows\Themes\TranscodedWallpaper.png   SUCCESS
BasicInformationFile              C:\Users\poc\AppData\Roaming\Microsoft\Windows\Themes\TranscodedWallpaper.png   SUCCESS
ile                               C:\Users\poc\AppData\Roaming\Microsoft\Windows\Themes\TranscodedWallpaper.png   SUCCESS
File                              C:\Users\poc\AppData\Roaming\Microsoft\Windows\Themes\TranscodedWallpaper.png   SUCCESS
AttributeTagFile                  C:\Users\poc\AppData\Roaming\Microsoft\Windows\Themes\TranscodedWallpaper.png   SUCCESS
ile                               C:\Users\poc\AppData\Roaming\Microsoft\Windows\Themes\TranscodedWallpaper.png   SUCCESS
File                              C:\Users\poc\AppData\Roaming\Microsoft\Windows\Themes\TranscodedWallpaper.png   SUCCESS
StandardInformationFile           C:\Users\poc\AppData\Roaming\Microsoft\Windows\Themes\TranscodedWallpaper.png   SUCCESS
BasicInformationFile              C:\Users\poc\AppData\Roaming\Microsoft\Windows\Themes\TranscodedWallpaper.png   SUCCESS
StreamInformationFile             C:\Users\poc\AppData\Roaming\Microsoft\Windows\Themes\TranscodedWallpaper.png   SUCCESS
BasicInformationFile              C:\Users\poc\AppData\Roaming\Microsoft\Windows\Themes\TranscodedWallpaper.png   SUCCESS
EaInformationFile                 C:\Users\poc\AppData\Roaming\Microsoft\Windows\Themes\TranscodedWallpaper.png   SUCCESS
File                              C:\Windows\Temp\TranscodedWallpaper.png                                         SUCCESS
ile                               C:\Windows\Temp\TranscodedWallpaper.png                                         SUCCESS
File                              C:\Windows\Temp\TranscodedWallpaper.png                                         SUCCESS
AttributeInformationVolume        C:\Windows\Temp\TranscodedWallpaper.png                                         SUCCESS
BasicInformationFile              C:\Windows\Temp\TranscodedWallpaper.png                                         SUCCESS
AttributeInformationVolume        C:\Users\poc\AppData\Roaming\Microsoft\Windows\Themes\TranscodedWallpaper.png   SUCCESS
RemoteProtocolInformation         C:\Users\poc\AppData\Roaming\Microsoft\Windows\Themes\TranscodedWallpaper.png   INVALID PARAI
SecurityFile                      C:\Users\poc\AppData\Roaming\Microsoft\Windows\Themes\TranscodedWallpaper.png   SUCCESS
IOfFileInformationFile            C:\Windows\Temp\TranscodedWallpaper.png                                         SUCCESS
sicInformationFile                C:\Windows\Temp\TranscodedWallpaper.png                                         SUCCESS
RemoteProtocolInformation         C:\Windows\Temp\TranscodedWallpaper.png                                         INVALID PARAI
ile                               C:\Windows\Temp\TranscodedWallpaper.png                                         SUCCESS
ile                               C:\Users\poc\AppData\Roaming\Microsoft\Windows\Themes\TranscodedWallpaper.png   SUCCESS
File                              C:\Windows\Temp\TranscodedWallpaper.png                                         SUCCESS
BasicInformationFile              C:\Windows\Temp\TranscodedWallpaper.png                                         SUCCESS
StandardInformationFile           C:\Windows\Temp\TranscodedWallpaper.png                                         SUCCESS
BasicInformationFile              C:\Windows\Temp\TranscodedWallpaper.png                                         SUCCESS
StandardInformationFile           C:\Windows\Temp\TranscodedWallpaper.png                                         SUCCESS
ile                               C:\Windows\Temp\TranscodedWallpaper.png                                         SUCCESS
```

File operation example

Then you need to determine the context, that is, on whose behalf the privileged operation is being performed. In our case, it was *NT AUTHORITY\SYSTEM*, which led to the LPE.



```
Image
    AnyDesk
    AnyDesk Software GmbH
Name:     AnyDesk.exe
Version:  8.0.10
Path:
C:\Program Files (x86)\AnyDesk\AnyDesk.exe
Command Line:
"C:\Program Files (x86)\AnyDesk\AnyDesk.exe" --backend

PID:         6136                      Architecture:  32-bit
Parent PID:  2676                      Virtualized:   False
Session ID:  1                         Integrity:     System
User:        NT AUTHORITY\SYSTEM
Auth ID:     00000000:000003e7
Started:     12/25/2024 3:07:17 AM     Ended:         (Running)
Modules:
Module            Address      Size        Path
AnyDesk.exe       0xe60000     0x1749000   C:\Program Files (x86)\
WinTypes.dll      0x6b50000    0xdb000     C:\Windows\SysWOW6
WinTypes.dll      0x6c30000    0xdb000     C:\Windows\SysWOW6
TextShaping.dll   0x61650000   0x94000     C:\Windows\SysWOW6
policymanager...  0x616f0000   0x83000     C:\Windows\SysWOW6
thumbcache.dll    0x61780000   0x48000     C:\Windows\SysWOW6
```

```
Frame  Module         Location                                          Address              Path
K 0    FLTMGR.SYS     FltDecodeParameters + 0x210c                      0xfffff8026425640c   C:\Windows\System32\drivers\FLTMGR.SYS
K 1    FLTMGR.SYS     FltDecodeParameters + 0x1bba                      0xfffff802642593a    C:\Windows\System32\drivers\FLTMGR.SYS
K 2    FLTMGR.SYS     FltAddOpenReparseEntry + 0x560                    0xfffff80264289f40   C:\Windows\System32\drivers\FLTMGR.SYS
K 3    ntoskrnl.exe   IofCallDriver + 0x55                              0xfffff80267c2a715   C:\Windows\system32\ntoskrnl.exe
K 4    ntoskrnl.exe   IoGetAttachedDevice + 0x194                       0xfffff80267c2bd14   C:\Windows\system32\ntoskrnl.exe
K 5    ntoskrnl.exe   NtDeviceIoControlFile + 0x25ed                    0xfffff8026801acdd   C:\Windows\system32\ntoskrnl.exe
K 6    ntoskrnl.exe   SeOpenObjectAuditAlarmWithTransaction + 0x58ce    0xfffff8026802c0e    C:\Windows\system32\ntoskrnl.exe
K 7    ntoskrnl.exe   ObOpenObjectByNameEx + 0x1fa                      0xfffff802680d96a    C:\Windows\system32\ntoskrnl.exe
K 8    ntoskrnl.exe   NtCreateFile + 0x49f                              0xfffff8026807b9f    C:\Windows\system32\ntoskrnl.exe
K 9    ntoskrnl.exe   NtCreateFile + 0x79                               0xfffff80268077779   C:\Windows\system32\ntoskrnl.exe
K 10   ntoskrnl.exe   setjmpex + 0x82d5                                 0xfffff80267e0caf5   C:\Windows\system32\ntoskrnl.exe
U 11   ntdll.dll      ZwCreateFile + 0x14                               0x7ffdb9f6db64       C:\Windows\System32\ntdll.dll
U 12   wow64.dll      Wow64KiUserCallbackDispatcher + 0x3fe6            0x7ffdb8d0739б       C:\Windows\System32\wow64.dll
U 13   wow64.dll      Wow64SystemServiceEx + 0x15a                      0x7ffdb8d0901a       C:\Windows\System32\wow64.dll
U 14   wow64cpu.dll   TurboDispatchJumpAddressEnd + 0xb                 0x777c17c3           C:\Windows\System32\wow64cpu.dll
U 15   wow64cpu.dll   BTCpuSimulate + 0x9                               0x777c11b9           C:\Windows\System32\wow64cpu.dll
U 16   wow64.dll      Wow64KiUserCallbackDispatcher + 0x4b9             0x7ffdb8d838c9       C:\Windows\System32\wow64.dll
U 17   wow64.dll      Wow64LdrpInitialize + 0x12d                       0x7ffdb8d832bd       C:\Windows\System32\wow64.dll
U 18   ntdll.dll      LdrInitializeThunk + 0x479                        0x7ffdb9f45089       C:\Windows\System32\ntdll.dll
U 19   ntdll.dll      LdrInitializeThunk + 0x63                         0x7ffdb9f44c73       C:\Windows\System32\ntdll.dll
U 20   ntdll.dll      LdrInitializeThunk + 0xe                          0x7ffdb9f44c1e       C:\Windows\System32\ntdll.dll
U 21   ntdll.dll      ZwCreateFile + 0xc                                0x7784304c           C:\Windows\SysWOW64\ntdll.dll
U 22   KernelBase.dll CreateFileW + 0x568                               0x772f2368           C:\Windows\SysWOW64\KernelBase.dll
U 23   KernelBase.dll CreateFileW + 0x5e                                0x772f1e5e           C:\Windows\SysWOW64\KernelBase.dll
U 24   KernelBase.dll BasepCopyFileExW + 0x2b0                          0x77303070           C:\Windows\SysWOW64\KernelBase.dll
U 25   KernelBase.dll CopyFileExW + 0x6e                                0x77302d6e           C:\Windows\SysWOW64\KernelBase.dll
U 26   KernelBase.dll CopyFileW + 0x1e                                  0x77302cee           C:\Windows\SysWOW64\KernelBase.dll
U 27   AnyDesk.exe    AnyDesk.exe + 0x284875                            0x10e4875            C:\Program Files (x86)\AnyDesk\AnyDesk.exe
```

Privileged File Operation

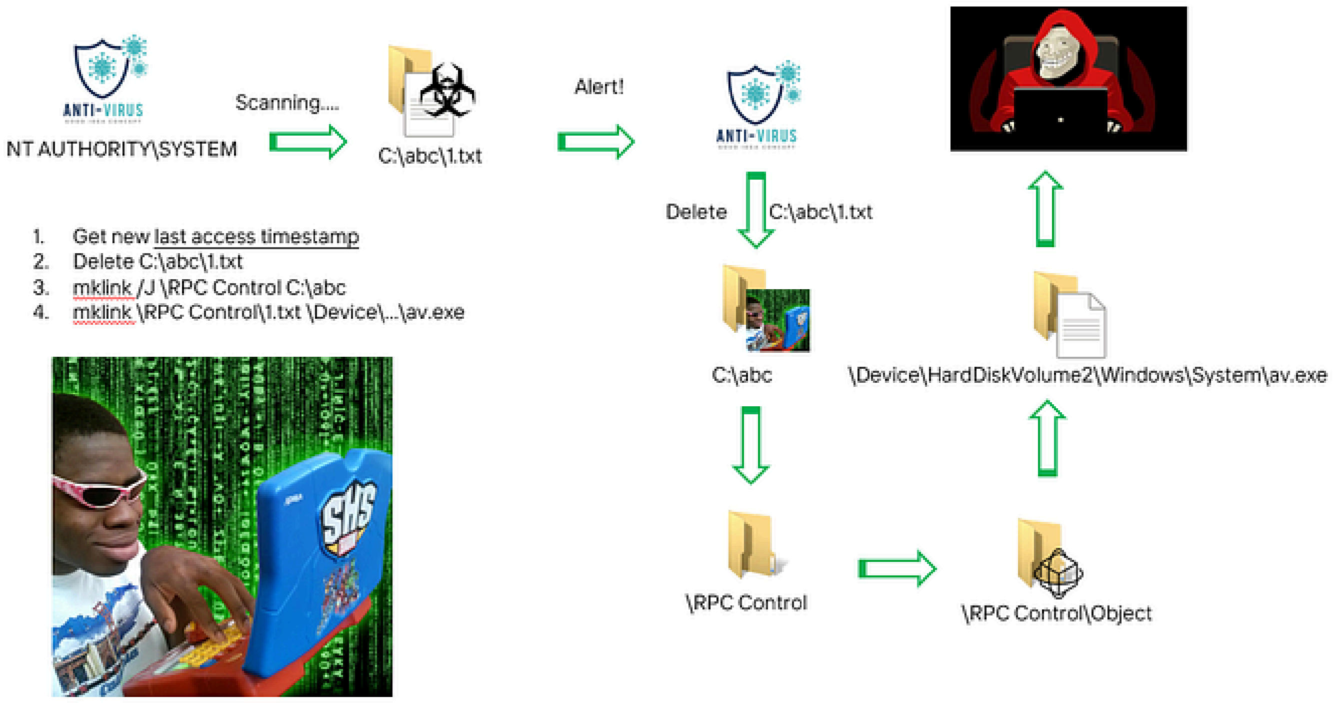
So, we just need to file a privileged file operation on a file we control, and then replace that file with a symbolic link. But how to trigger this privileged file operation? Here I offer several options.

- Sending an IOCTL to the driver — you can try to trigger the driver to perform a file operation on a file under your control;
- Bulk creation of COM objects — When initializing, COM objects can use different files, while running on behalf of the NT AUTHORITY\SYSTEM;
- MSI Packet Installation — different files may be used during the installation of an MSI package;

- GUI/RPC/ALPC — finally, you can try to interact with the target application via the IPC interfaces it provides.

Arbitrary File Delete

Let's take a closer look at the vulnerability. For example, how Arbitrary File Delete was used in the past to abuse antiviruses.


ANTI-VIRUS
NT AUTHORITY\SYSTEM
Scanning....
C:\abc\1.txt
Alert!
ANTI-VIRUS
Delete
C:\abc\1.txt
1. Get new last access timestamp
2. Delete C:\abc\1.txt
3. mklink /J \RPC Control C:\abc
4. mklink \RPC Control\1.txt \Device\...\av.exe
C:\abc
\Device\HardDiskVolume2\Windows\System\av.exe
\RPC Control
\RPC Control\Object

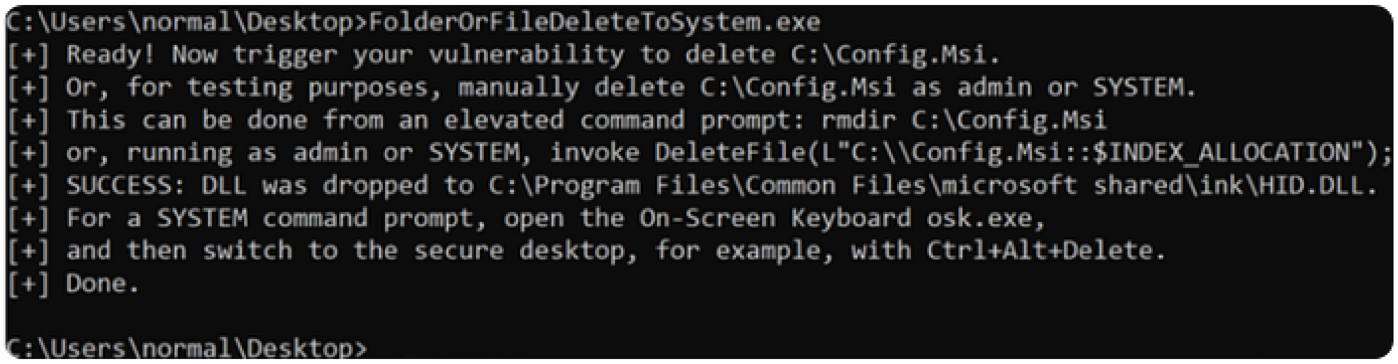
AV Abuse Example

So, we have an antivirus service that operates on behalf of NT AUTHORITY\SYSTEM. The hacker places a malicious EICAR file on the device disk. Then it starts checking the Last Access TimeStamp in a loop in an attempt to detect access to its file. The antivirus accesses the file, identifies malicious file, and then tries to delete it from the disk. However, at this point the hacker replaces his file with a symbolic link, which, by connecting NTFS Mount Point and Object Manager Symlink, leads to a critical file for the antivirus itself. This is how the antivirus deletes itself :)

Join Medium for free to get updates from this writer.

So what can we do once we have achieved arbitrary file deletion?

Here is a more general option — try to delete some DLL library. Then write your own load in its place. Focus on the Search Order Hijacking and DLL Redirection mechanisms. However, this is a slightly more complex case. Consider a simpler method — abuse of Windows Installer Rollback. This method allows you to get the system shell from arbitrary file deletion primitive.



```
C:\Users\normal\Desktop>FolderOrFileDeleteToSystem.exe
[+] Ready! Now trigger your vulnerability to delete C:\Config.Msi.
[+] Or, for testing purposes, manually delete C:\Config.Msi as admin or SYSTEM.
[+] This can be done from an elevated command prompt: rmdir C:\Config.Msi
[+] or, running as admin or SYSTEM, invoke DeleteFile(L"C:\\Config.Msi::$INDEX_ALLOCATION");
[+] SUCCESS: DLL was dropped to C:\Program Files\Common Files\microsoft shared\ink\HID.DLL.
[+] For a SYSTEM command prompt, open the On-Screen Keyboard osk.exe,
[+] and then switch to the secure desktop, for example, with Ctrl+Alt+Delete.
[+] Done.

C:\Users\normal\Desktop>
```

Windows Installer Rollback abuse

The mechanism is based on the fact that we delete the C:\Config.msi folder, write our own malicious MSI files and call the Rollback mechanism, which leads to the execution of these files. You can explore this mechanism in more detail in these CVEs:

- CVE-2023–27470 — Take Control Agent 7.0.41.1141 LPE;
- CVE-2023–20178 — Arbitrary File Delete vulnerability in Cisco Secure Client (tested on 5.0.01242) and Cisco AnyConnect (tested on 4.10.06079);
- CVE-2022–30206 — Windows Print Spooler Elevation of Privilege Vulnerability.

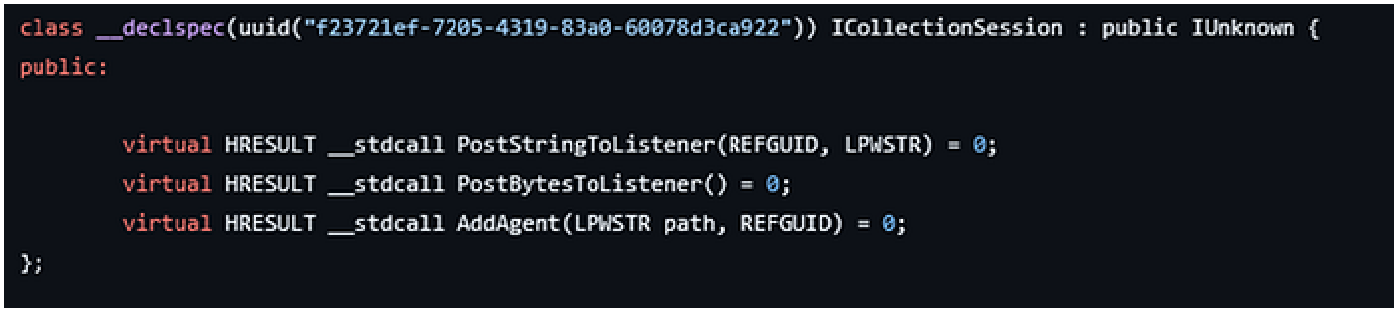
Fun Fact

To be honest, I was quite surprised by the fact that the topic of symbolic link abuse is largely ignored in the leading and most popular offensive security courses. Moreover, this vulnerability is more than relevant. Just a couple of months ago, another CVE was registered, related to the abuse of symbolic links. Read the research here.

Arbitrary File Create/Copy && Some Tricks

So, what if you can control whether a file is created or copied instead of deleting it? In this case, I suggest using two options for abuse:

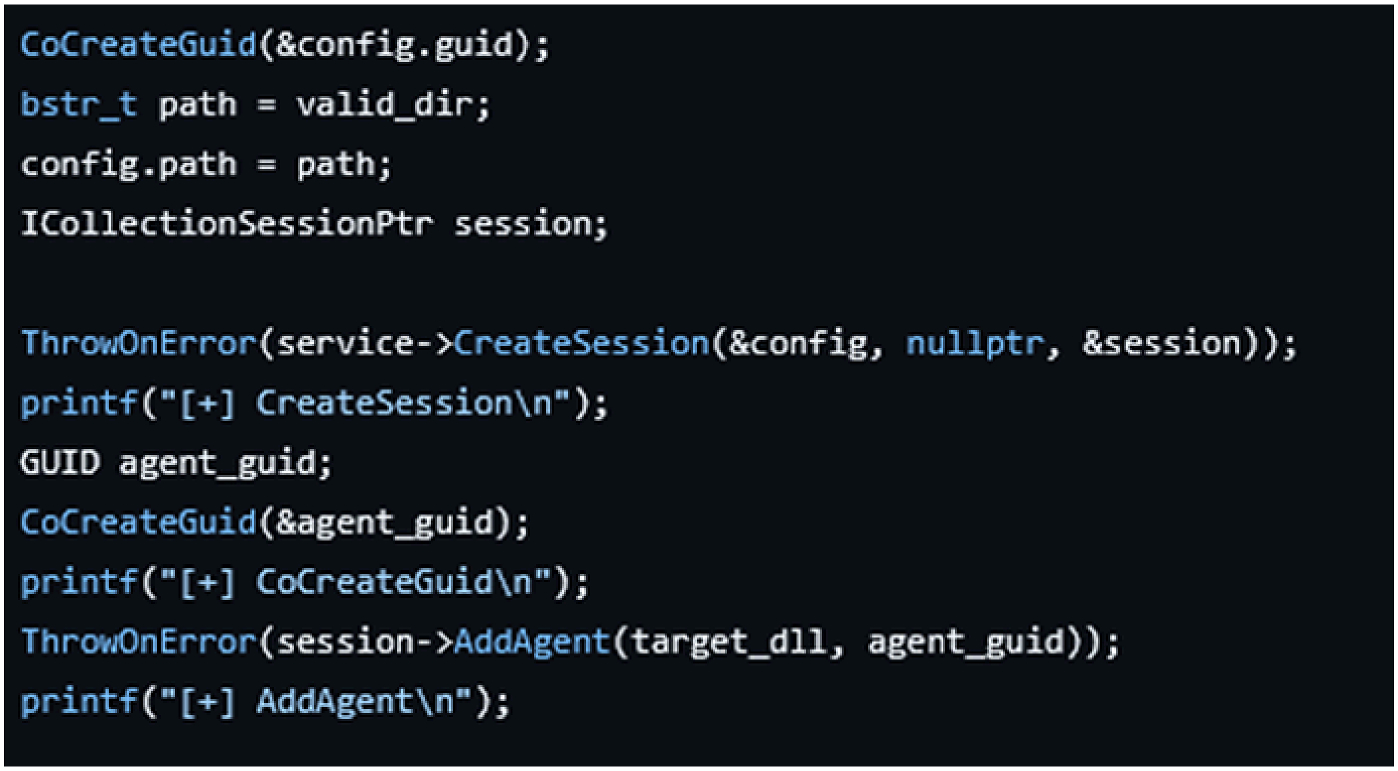
- More general option: Try writing a new configuration file to some service, writing scripts to the system-wide startup or using the DLL Redirection mechanism (create .local files);
- However, there is a potentially vulnerable DiagHub service that will allow us to load an arbitrary library from C:\Windows\System32. So, try to write your library to this path and trigger the service. You can read more here and here.



```
class __declspec(uuid("f23721ef-7205-4319-83a0-60078d3ca922")) ICollectionSession : public IUnknown {
public:

        virtual HRESULT __stdcall PostStringToListener(REFGUID, LPWSTR) = 0;
        virtual HRESULT __stdcall PostBytesToListener() = 0;
        virtual HRESULT __stdcall AddAgent(LPWSTR path, REFGUID) = 0;
};
```

DiagHub vulnerable interface definition



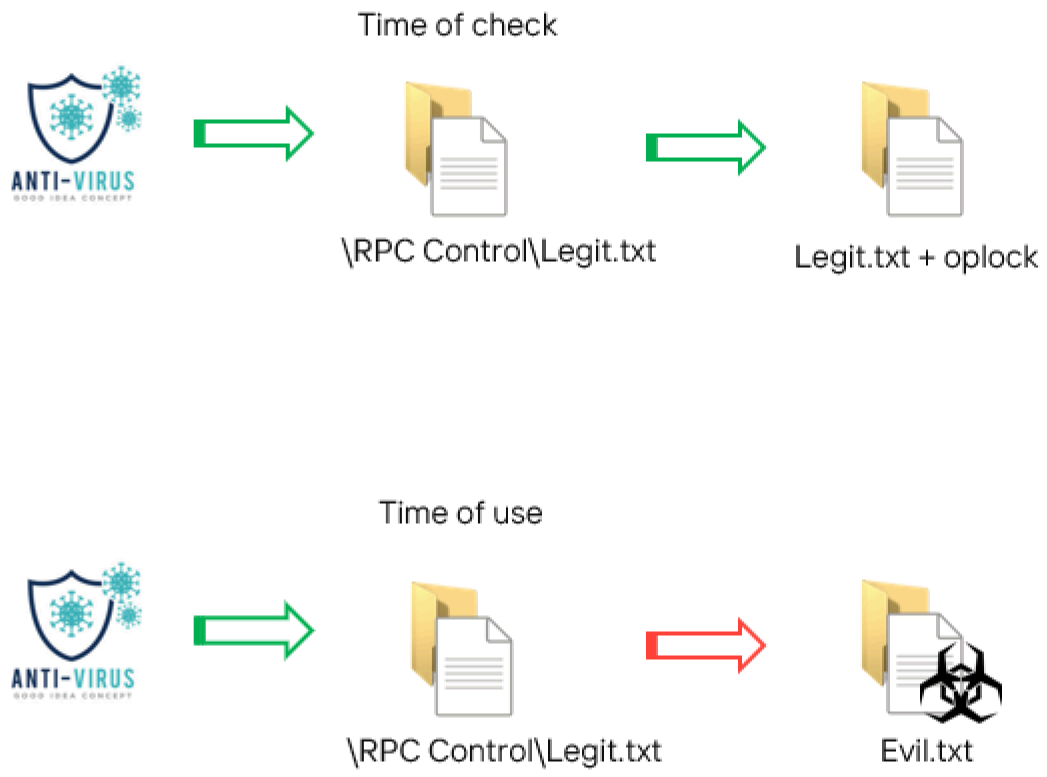
```
CoCreateGuid(&config.guid);
bstr_t path = valid_dir;
config.path = path;
ICollectionSessionPtr session;

ThrowOnError(service->CreateSession(&config, nullptr, &session));
printf("[+] CreateSession\n");
GUID agent_guid;
CoCreateGuid(&agent_guid);
printf("[+] CoCreateGuid\n");
ThrowOnError(session->AddAgent(target_dll, agent_guid));
printf("[+] AddAgent\n");
```

DiagHub initialization and abuse

However, during exploitation you may encounter a problem — the target application checks the file before copying or creating. In this case, look towards the BaitAndSwitch repository. This repository will allow you to perform a Time of check — time of use (TOCTOU) attack.

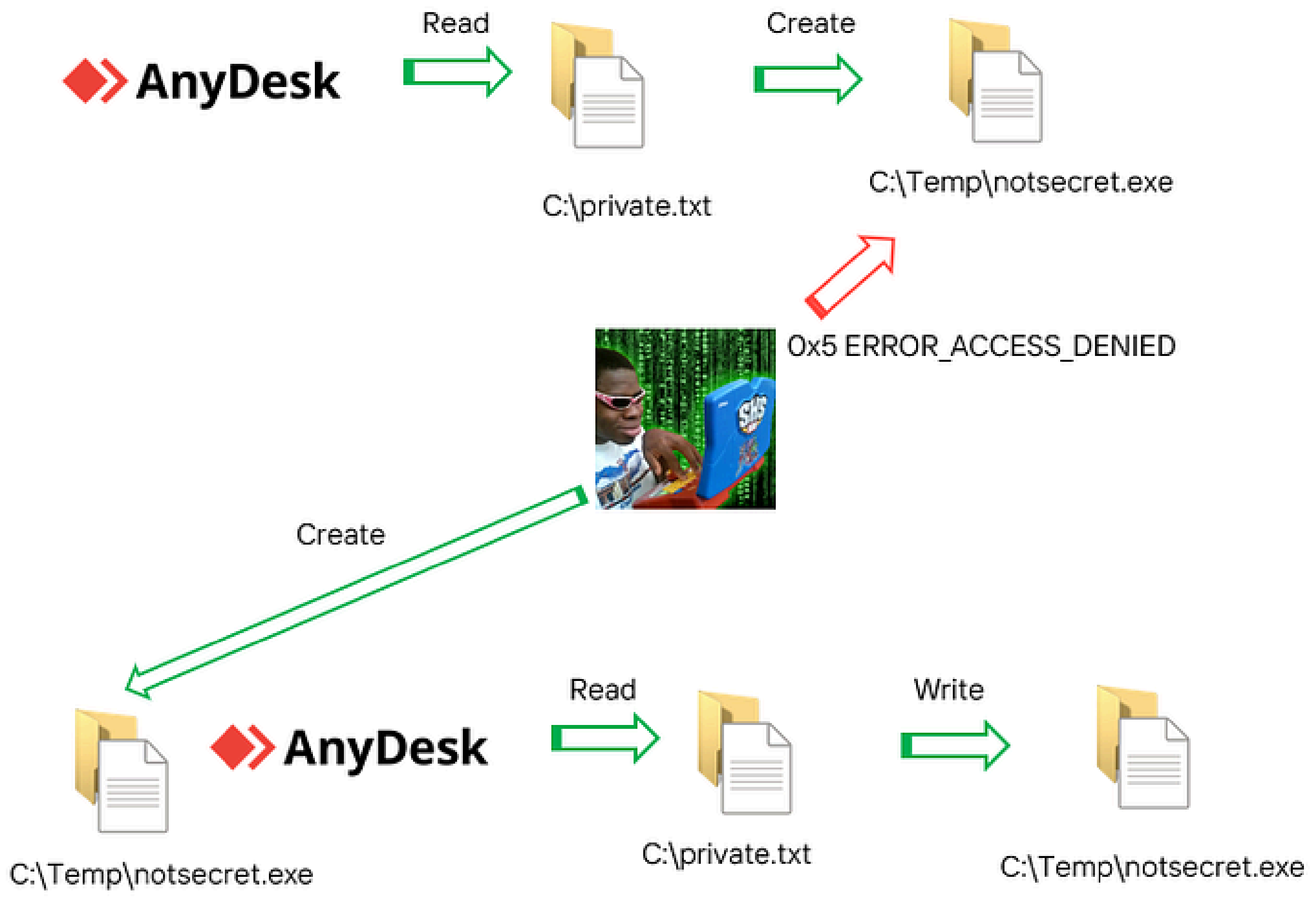
What does it consist of? First, we also create a symbolic link, but it points to a legitimate file that is being checked. In turn, OpLock is installed on this legitimate file. OpLock allows you to track requests to a legitimate file. As soon as the target service requests a legitimate file, OpLock is triggered and our symbolic link changes from a legitimate file to a malicious one.



TOCTOU Abuse Example

This is the first trick you can use when abusing arbitrary copying or arbitrary file creation.

The second trick is to override DACL. It was used when writing POC on LPE in AnyDesk. We had the following case: we could perform arbitrary copying of any system files to the *C:\Windows\Temp* folder, but the files were copied together with their original DACL. Thus, we could not read the files. However, we simply recreated the target files in the *C:\Windows\Temp* folder and the vulnerable service involuntarily performed an arbitrary overwrite operation instead of arbitrary copying. And in this case, the original DACL is used. That is, the file that we created ourselves. And we can read our own files without problems.



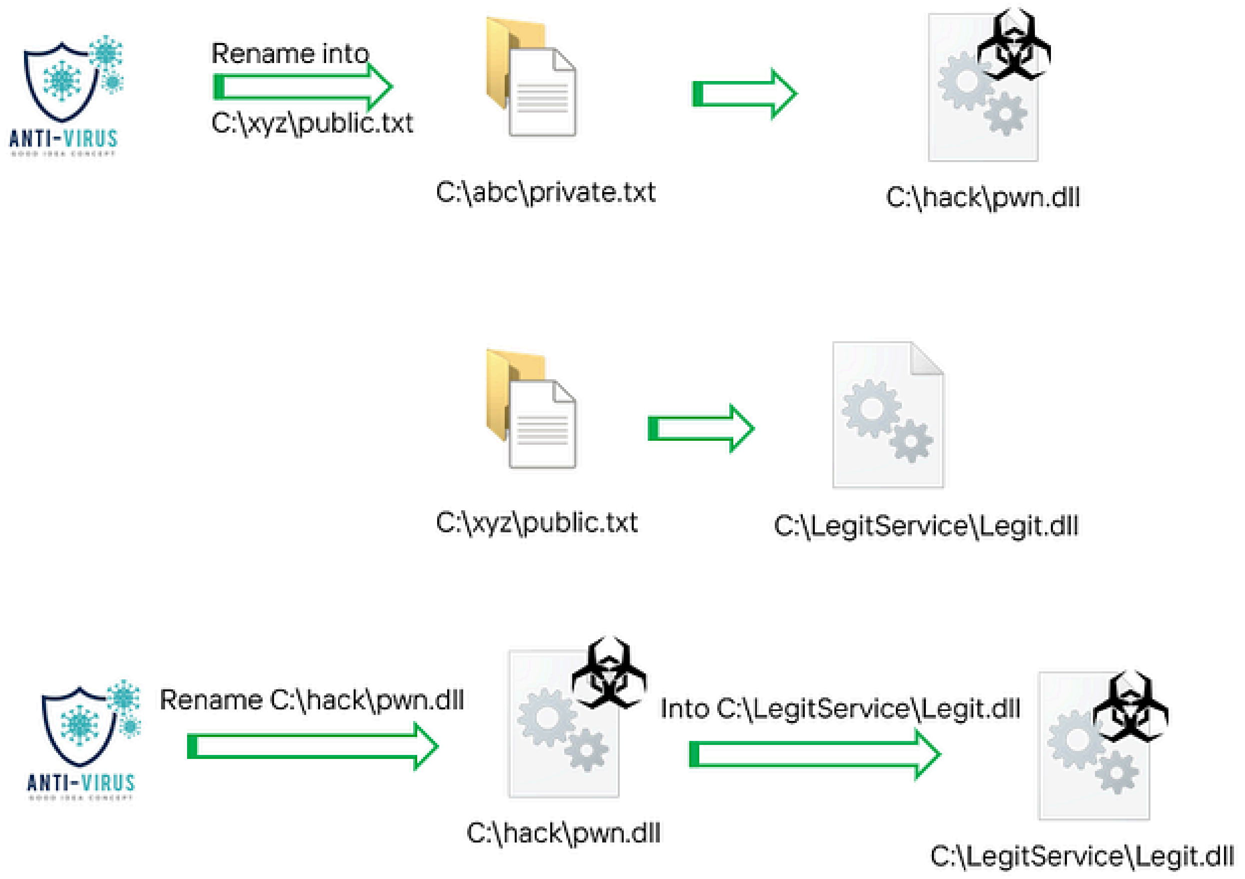
Overriding DACL Example

U can read more about it here.

Arbitrary File Overwrite/Move

It is worth noting that if you have achieved a move or overwrite primitive, the methods of abuse will not differ from the methods used for arbitrary copying or creation. Also look into writing DLL libraries or using DiagHub.

Did you know that renaming can move a file? Let's take a closer look. Let's say we have some privileged service that performs the rename operation from *C:\abc\private.txt* to *C:\xyz\public.txt*. In this case, we control both files: private.txt and public.txt.

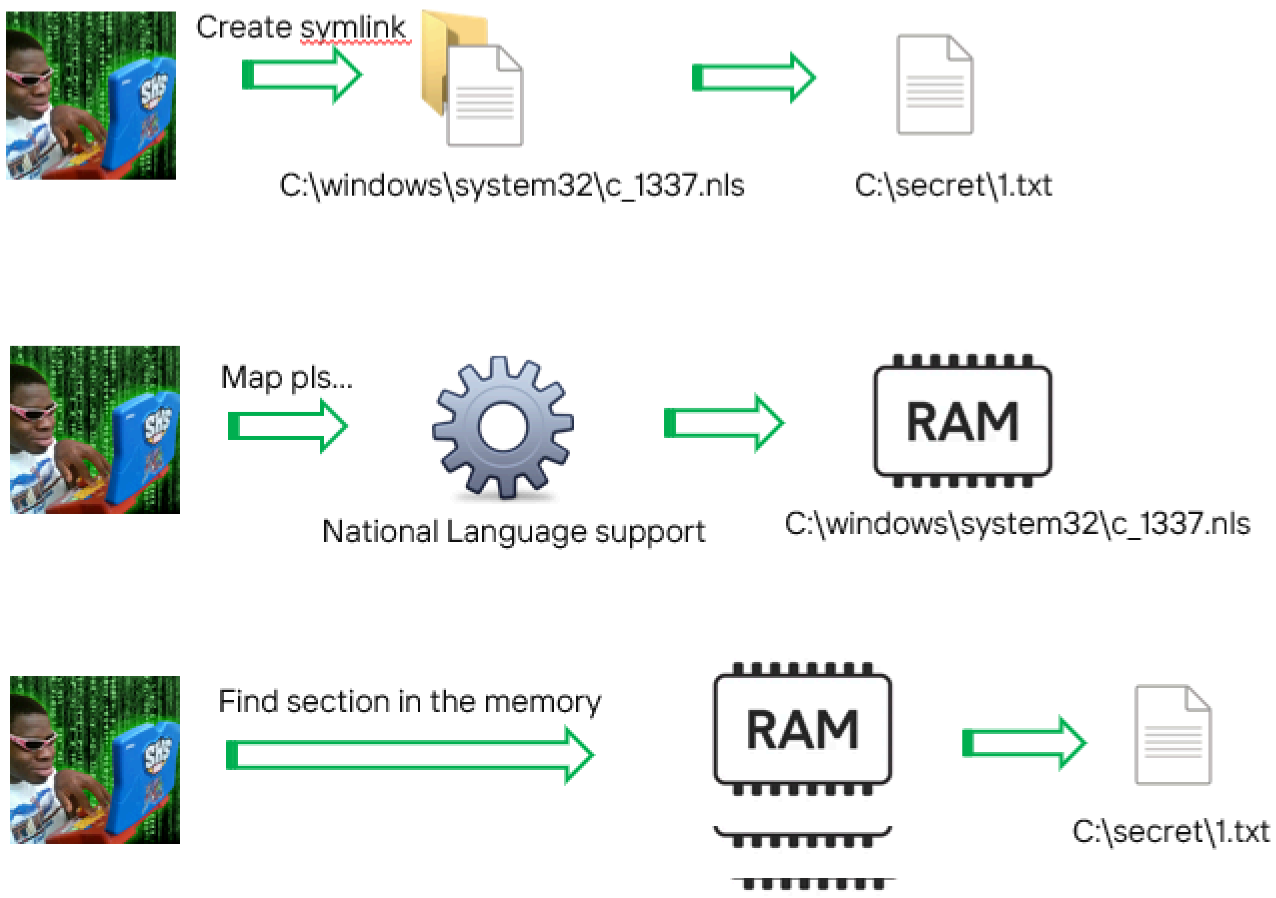


Renaming to moving…

In this case, we can make two symbolic links: one will be called private.txt, and the second public.txt. And they will point to the files we need to move. For example, private.txt will point to the malicious DLL library *C:\hack\pwn.dll*, which we want to write to the path *C:\LegitService\Legit.Dll*. Accordingly, public.txt should point to *C:\LegitService\Legit.dll*. Thus, when the service wants to rename *C:\abc\private.txt* to *C:\xyz\public.txt*, it will rename *C:\hack\pwn.dll* to *C:\LegitService\Legit.dll*. You can read more about this wonderful technique here.

Arbitrary Directory Creation

There are more sophisticated methods. For example, when you can control the folder being created. In this case, you can try to untwist it to arbitrary reading of system files.



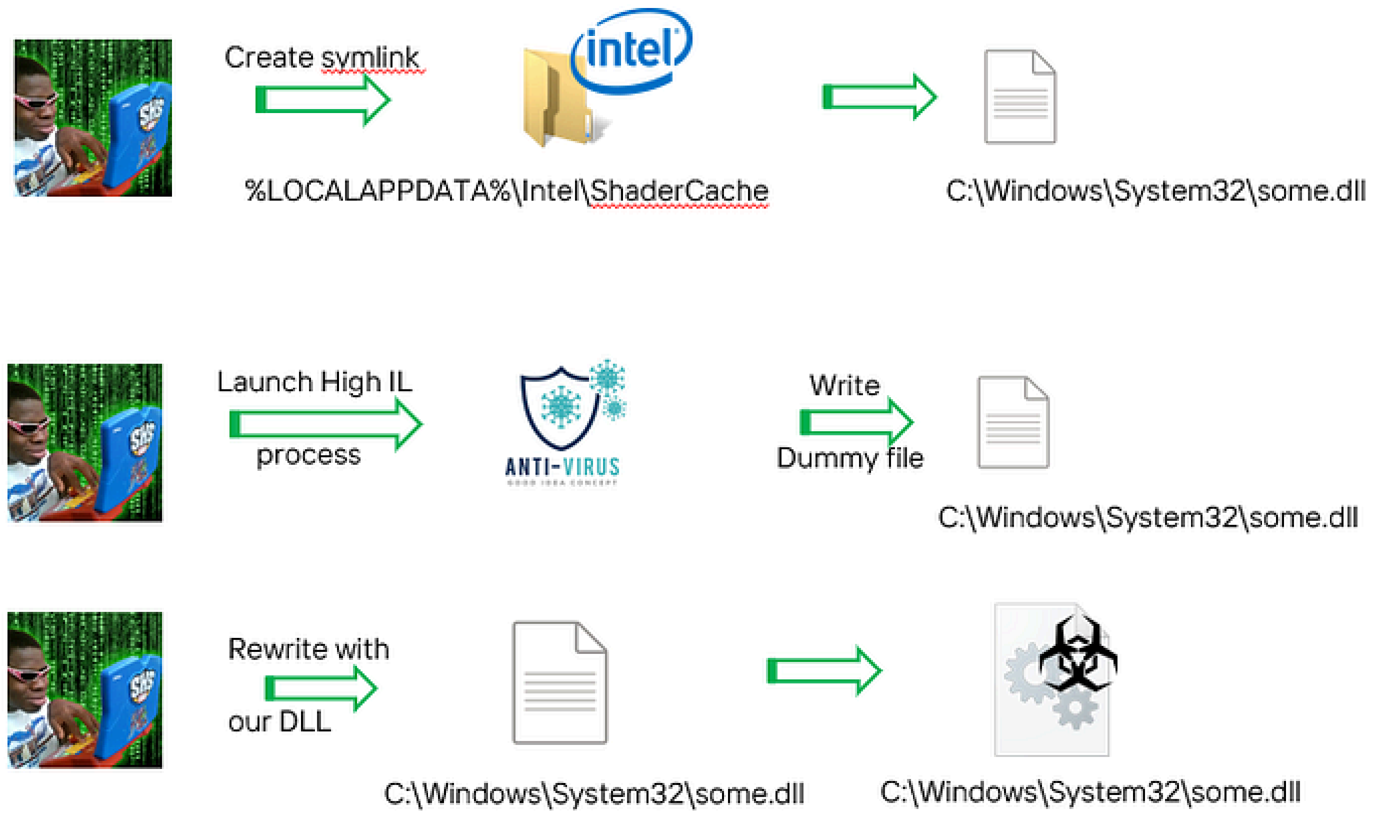
Create symlink
C:\windows\system32\c_1337.nls
C:\secret\1.txt
Map pls...
National Language support
RAM
C:\windows\system32\c_1337.nls
Find section in the memory
RAM
C:\secret\1.txt

From Arbitrary Directory Creation to Arbitrary File Read

In this case, you should create a folder with a special name, for example, *c_1337.nls*, convert it to NTFS Mount Point to the desired file. Yes, to the file, not to the folder, we will work with the National Language Support service, which uses a special system call that allows you to convert a symbolic link NTFS Mount Point to a file. After that, we trigger the National Language Support service, the target file is mapped to shared memory. All we have to do is find the desired file in memory by its characteristic name. You can find the POC here, and the original research here.

Symlinks to UAC Abuse

You can use symbolic links to bypass UAC! It is worth noting that this method only works on systems running Intel processors. To abuse it, you need to locate the ShaderCache folder, delete all files from there, create an NTFS Mount Point, achieve a random overwrite primitive, and write your own DLL.


Create symlink
%LOCALAPPDATA%\Intel\ShaderCache
C:\Windows\System32\some.dll
Launch High IL process
ANTI-VIRUS
Write Dummy file
C:\Windows\System32\some.dll
Rewrite with our DLL
C:\Windows\System32\some.dll
C:\Windows\System32\some.dll

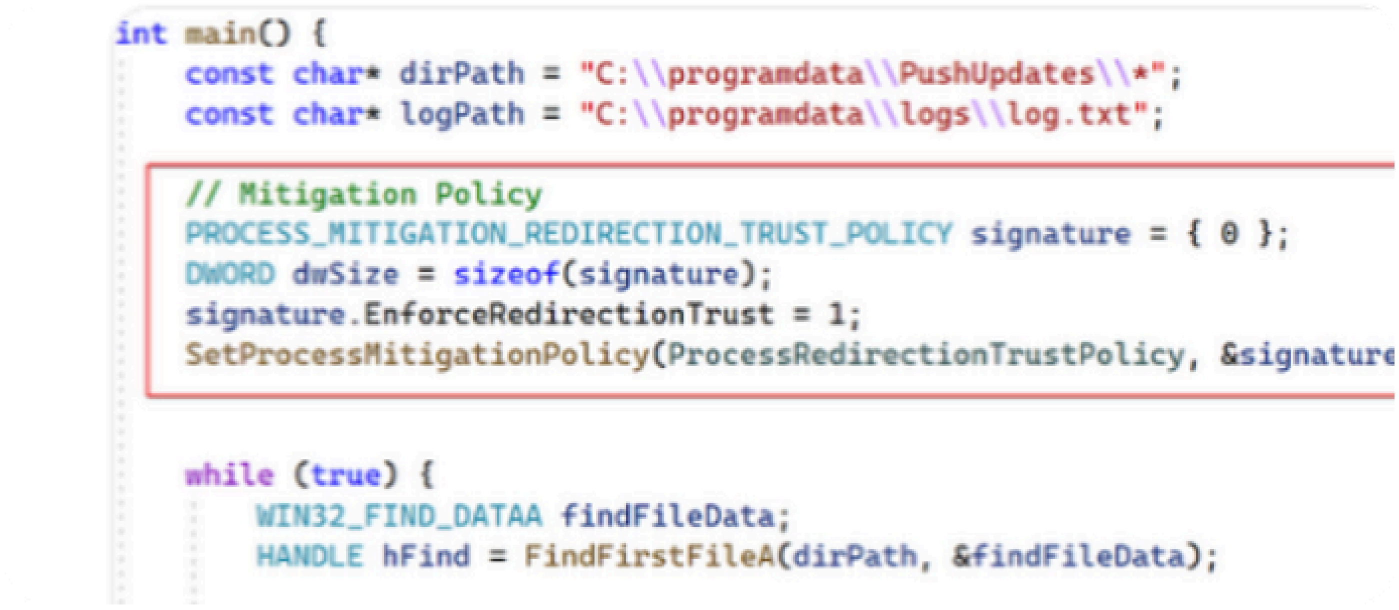
UAC Bypass using symlinks

You can find the original research here.

Defense notes

So how can you protect yourself from symbolic link abuse? I see two options:

> RedirectionTrust — the system does not follow symbolic links that are created by a user with a lower IL;



```
int main() {
    const char* dirPath = "C:\\programdata\\PushUpdates\\*";
    const char* logPath = "C:\\programdata\\logs\\log.txt";

    // Mitigation Policy
    PROCESS_MITIGATION_REDIRECTION_TRUST_POLICY signature = { 0 };
    DWORD dwSize = sizeof(signature);
    signature.EnforceRedirectionTrust = 1;
    SetProcessMitigationPolicy(ProcessRedirectionTrustPolicy, &signature

    while (true) {
        WIN32_FIND_DATAA findFileData;
        HANDLE hFind = FindFirstFileA(dirPath, &findFileData);
```

Enabling RedirectionTrust example

POC to find the RedirectionTrust mechanism in running processes.

> Impersonation — Your service must use a low-privilege user token for file operations. You can use someone else's token using special functions, such as ImpersonateLoggedOnUser().

Also, check out this exploit blocker.

Conclusion

Thank you for reading my article to the end. I hope the material will help you in studying the abuse of symbolic links. Subscribe to us in X!