Microsoft 365 Security

Everything about Microsoft Security

DFIR – Windows and Active Directory persistence and malicious configurations

Posted on June 23, 2021 by m365guy Leave a comment
This blog post is just meant for me. I've decided to write it, because one day. I'm confident that I will use it during an incident response. In this blog post, we will cover different persistence that are relevant. It does not mean that we will cover every persistence that's out there, which is kind of unrealistic. Due to the fact, that Windows & Active Directory provides various way for an attacker, to remain persistent. I will keep this blog post up-to-date, when I discover new things that are worth to add to the list.

Another thing we will look at is, the malicious configuration(s) that may not by default, indicate a persistence mechanism. However, it can be used to make a system more vulnerable (e.g. enabling WDigest to obtain plain-text password). The definition of "malicious" configuration can be different for anybody, but it may be an IT Admin that doesn't know better and decides to configure something insecurely, which opens the door for an attacker.

This checklist includes well-known persistence mechanism and malicious configurations that has been published on the internet, but also includes additional things from my side as well. Both have been combined together to have a good foundation on the things to look at.

All of this command and screenshots are meant as a reference.

Windows Operating System

We will start with the Windows Operating System itself. This will include a couple of basic checks that are well-known and have been published all over the web. It will also include a few things, that may not be that known to the public. However, it has been relevant to cover.

We'll start with the quick wins first and then go from there further.

- **WDigest enabled?**

WDigest is an authentication protocol that stores passwords in plain-text. By default, this functionality has been deprecated in Windows 10. However, an attacker with local admin can just make a registry flip and still enable it to obtain the plain-text password.

Command

```
reg query HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest /v
UseLogonCredential
```

Result

At the sample result, we can see that WDigest has been enabled. Since the registry value of "UseLogonCredential" has been set to 0x1.

- **Windows Defender AV**

1. Check if there have been any exclusions to being configured. This will exclude a folder from being scanned by AV, which is a simple way to avoid triggers.

Command

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

Result

At the sample result, we can see that the C:\Temp folder has been excluded from Windows Defender AV.

2. The second check will look if Windows Defender AV has been disabled.

Command

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

Result

At the sample result, we can see that the "DisableAntiSpyware" value has been set to 0x1. This means that it is disabled.

- **Sticky keys**

A well-known persistence trick to keep having SYSTEM privileges.

Command

```
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options"
```

Result

At the sample result, we can see that the sethc.exe binary has been set. However, it is not the only binary that can be used during a sticky keys attack. Keep an eye on the following binaries: **sethc.exe**, **utilman.exe**, **osk.exe**, **Magnify.exe**, **Narrator.exe**, **DisplaySwitch.exe**, **AtBroker.exe**

- **Remote Desktop Protocol (RDP)**

1. The first check will be looking if RDP has been enabled on a system. APT3 has enabled RDP on a compromised machine to remain persistent.

Command

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

Result

At the sample result, we can see that the "fDenyTSConnections" registry value has been set to 0x0, which means that it has been enabled.

2. The second check related to RDP is to see if RDP shadow session has been enabled. In Windows, there is a Group Policy setting that can enforce certain options with the likes of viewing user sessions without their permission.

Command

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

Result

At the sample result, we can see that the value has been set to 0x4, which translates to the following option: **View Session without user's permission**

3. In an APT report of FireEye, it was stated that the DarkSide gang enabled multiple RDP session. We can't tell what the exact reason behind this is, but it may perhaps be a trick to lure multiple admins do an interactive logon to a compromised server to be able to get the NT hashes of it. Who knows, but it's worth to check it out.

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 limitation has been removed.

- **Login to a console without a password**

This trick has been seen in different Threat Intel reports whereby an adversary is making a registry flip to login to an machine with a local account without an password.

Command

```
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v LimitBlankPasswordUse
```

Result

At the sample result, we can see that an empty password is allowed to a login console. Because we can see that the "LimitBlankPasswordUse" has been set to 0x0.

- **Hide account from Windows logon screen**

An evasion trick to hide some trace, which won't display the username of a user that has recently logged in.

Command

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

Result

At the sample result, we can see that this setting has been configured.

- **Disable UAC Remote Restriction**

By default, a non RID 500 local admin will not connect as a full administrator, has no elevation potential, and cannot perform administrative tasks. In order to administer the workstation the user must interactively login. However, if HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\LocalAccountTokenFilterPolicy is enabled, all local users connecting remotely are granted full administrative rights.

Command

```
reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System" /v
LocalAccountTokenFilterPolicy
```

Result

At the sample result, we can see that UAC remote has been disabled. Because the "LocalAccountTokenFilterPolicy" has been set to 0x1.

- **Image File Execution Options**

A great blog post from Odvar Moe about having persistence with "GlobalFlags" to evade auto runs. In his blog post (https://web.archive.org/web/20210623173110/https://oddvar.moe/2018/04/10/persistence-using-globalflags-in-image-file-execution-options-hidden-from-autoruns-exe/), he explains how to execute any binary file after another application is closed without being detected by Autoruns.exe – In other words, when a user has for example notepad.exe open, but then decides to close notepad. It will automatically launch the executable.

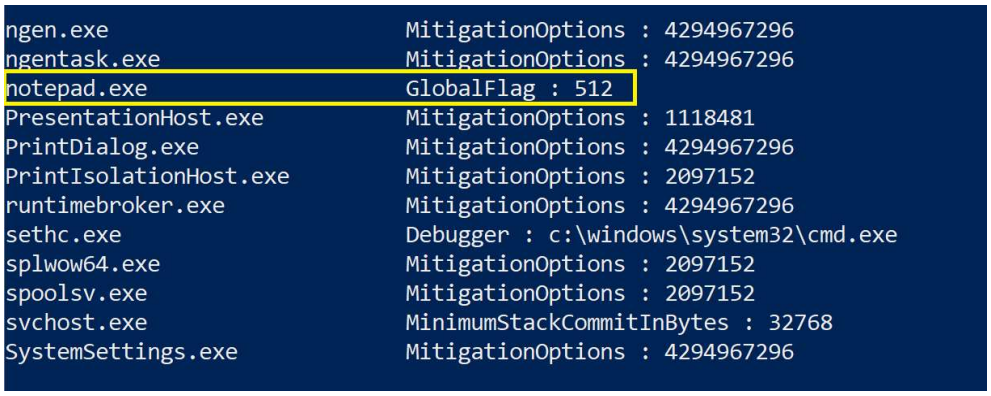
1. Look to see if we see the property "GlobalFlag"

Command

```
Get-ChildItem -Path 'HKLM:SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options'
```

Result

At the sample result, we can see notepad.exe has the "GlobalFlag" property.



2. The second step would be to look what kind of binary would be ran, once notepad.exe has been closed.

Command

```
Get-ChildItem -Path 'HKLM:SOFTWARE\Microsoft\Windows NT\CurrentVersion\SilentProcessExit'
```

Result

At the sample result, we can see that in this example. It would be cmd.exe

- **Registry Run Keys**

Creating registry keys that will execute an arbitrary payload during Windows logon. Very old technique, but still relevant.

Command

```
reg query "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run"
reg query "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce"
reg query "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices"
reg query "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce"

reg add "HKEY_LOCAL_MACHINE\software\Microsoft\Windows\CurrentVersion\RunOnceEx"
reg add "HKEY_LOCAL_MACHINE\software\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit"
reg add "HKEY_LOCAL_MACHINE\software\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell"
reg add "HKEY_LOCAL_MACHINE\software\Microsoft\Windows NT\CurrentVersion\Windows"
reg add "HKEY_LOCAL_MACHINE\software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders"
reg add "HKEY_LOCAL_MACHINE\system\CurrentControlSet\Control\SafeBoot\AlternateShell"
```

Result

At the sample result, we can see a couple of programs. SecurityHealth.exe and RtkAudUService64.exe are the legitimate one's, but the others should be investigated.

- **No logs to Security Event Logs**

@0gtweet has tweeted (https://web.archive.org/web/20210623173110/https://twitter.com/0gtweet/status/1182516740955226112) about a fun technique to stop any logs to the Security Event logs. This means that once an attacker adds the following registry value: CurrentControlSet\Control**MiniNt**. Windows will think it is WinPE and will not log any event to the Security Log.

Command

```
reg query "HKLM\System\CurrentControlSet\Control\MiniNt"
```

Result

At the result, we can see that no logs are available anymore when looking to Security logs.

- **Boot or Logon Initialization Scripts: Logon Script**

Adversaries may use Windows logon scripts automatically executed at logon initialization to establish persistence. This is done via adding a path to a script to the `HKCU\Environment\UserInitMprLogonScript`

Command

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

Result

At the sample result, we can see that the "UserInitMprLogonScript" value has been created.

- **DLL Load via LSASS**

An adversary may load arbitrary DLLs to extract credentials from memory, which has been blogged here (https://web.archive.org/web/20210623173110/https://blog.xpnsec.com/exploring-mimikatz-part-1/).

Command

```
reg query HKLM\SYSTEM\CurrentControlSet\Services\NTDS /v LsaDbExtPt
reg query HKLM\SYSTEM\CurrentControlSet\Services\NTDS\DirectoryServiceExtPt
```

Result

At the sample result, we can see that an DLL has been loaded.

- **Modifying existing services**

An adversary may modify an existing service to execute arbitrary code.

Command

```
reg query HKLM\SYSTEM\CurrentControlSet\Services /s /v "ImagePath"
reg query HKLM\SYSTEM\CurrentControlSet\Services /s /v "ServiceDLL"
reg query HKLM\SYSTEM\CurrentControlSet\Services /s /v "FailureCommand"
```

Result

The **ImagePath** registry key typically contains the path of the driver's image file. Hijacking this key with an arbitrary executable will have as a result the payload to run during service start.

- **Group Policy Logon Script**

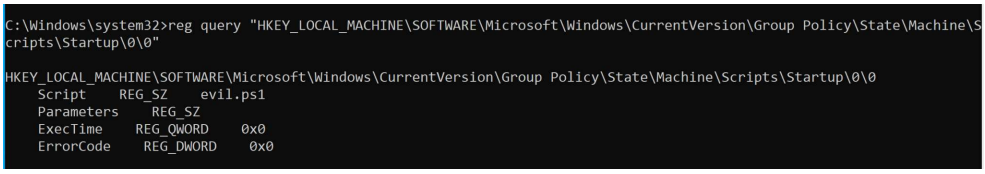
Startup scripts in Group Policy can run scripts before the boot process, which can be persistence trick of an adversary.

Command

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

Result

At the sample result, we can see a logon script that has been attached.



Active Directory Backdoors

Adversaries can create backdoors in Active Directory to remain persistent. However, an suspicious configuration does not immediately indicate that it's an adversary. It can also be an IT Admin that doesn't know better. Well, that's been said. This chapter will cover what AD objects and attributes should be reviewed, and so on. When it comes down to the definition of "backdoors", we actually mean. An backdoor that allows an adversary to get full domain dominance once they are inside a network again. You can think of being a DA or equivalent.

- **AdminSDHolder Container**

Verify that the AdminSDHolder is in a clean state, which means that accounts or groups that are NOT **Domain Admins**, **Enterprise Admins**, and **Administrators** groups. Should be treated as "malicious" and may indicate a potential backdoor in the environment.

Command

```
$ADSI=[ADSI]"LDAP://CN=AdminSDHolder,CN=System,DC=redmond,DC=com"
$ADSI.psbase.get_ObjectSecurity().getAccessRules($true, $true,
[system.security.principal.NtAccount])
```

Result

At the sample result, we can see an ACE with GenericAll on the AdminSDHolder container. This doesn't look, so the permission should be revoked.

Look for any ACE that has GenericWrite or equivalent (e.g. GenericAll, WriteDacl, WriteOwner, ExtendedRight)

- **Review Group Policy permissions linked to Domain Controllers**

Review all the different Access Control Entries that have been granted write permissions to GPOs linked to a Domain Controller. This is not limited to only the Default Domain Controllers Policy, when you have others GPOs linked to it as well.

However, since this is an example. We will pick the Default Domain Controllers Policy as reference and review all the permissions that have been assigned to this GPO.

Any ACE with the rights that has the permissions

Command

```
$ADSI=[ADSI]"LDAP://CN={6AC1786C-016F-11D2-945F-
00C04fB984F9},CN=Policies,CN=System,DC=redmond,DC=com"
$ADSI.psbase.get_ObjectSecurity().getAccessRules($true, $true,
[system.security.principal.NtAccount])
```

Result

At the sample result, we can see an ACE with WriteDacl on the Default Domain Controllers Policy.

We can see this ACE in the GPO Management Console as well.

- **Review User Right Assignment privileges on GPOs linked to DC**

There are sensitive privileges that can be granted via User Right Assignments that can provide escalation paths to domain compromise. One of those permission is SeTakeOwnerShipPrivileges (Take ownership of files or other objects) for example, which allows a user to modify the ownership of every AD object, including the Domain Admins group.

At the sample result, we can see that StandardUser has the rights to take ownership of every AD object.

The following privileges should be closely reviewed on every GPO that is linked to a DC:

1. Backup files and directories (SeBackupPrivilege) – Backup Operators, Server Operators
2. Restore files and directories (SeRestorePrivilege) – Backup Operators, Server Operators
3. Load and unload device drivers (SeLoadDriverPrivilege) – Print Operators
4. Take ownership of files and objects (SeTakeOwnerShipPrivilege)
5. Enable this computer and user account to be trusted for delegation (SeEnableDelegationPrivilege)

SeEnableDelegationPrivilege requires to have at least GenericWrite or equivalent on a security principal to modify the userAccountControl in order to exploit the privileges.

In order for an attacker to modify the User Right Assignments. The attacker would have to go to the SYSVOL folder on a Domain Controller and lately ending up in a location that looks similar to this:

\DC02\sysvol\contoso.local\Policies{6AC1786C-016F-11D2-945F-00C04fB984F9}\MACHINE\Microsoft\Windows NT\SecEdit

The second would be to modify the GptTmpl file and grant themselves SeTakeOwnerShipPrivileges for example.

- **Scheduled Task to add user to the local Administrators group**

An adversary can create an immediate task on a Domain Controller to add an account automatically to the local Administrators group of a DC, which means that the account is a DA or equivalent.

In this example, we can see a scheduled task that adds a user to the local Administrators group of a Domain Controller.

When we look at the SYSVOL folder on a Domain Controller and look at the Default Domain Controllers Policy. We can see a folder that's called ScheduledTask, when we visit the following path:

\NYK-DC01\sysvol\redmond.com\Policies{6AC1786C-016F-11D2-945F-00C04fB984F9}\MACHINE\Preferences\ScheduledTasks

Verify and see what actions this scheduled task is doing.

- **Malicious service to add user to the local Administrators group**

An adversary that managed to obtain admin rights on a Domain Controller for instance. Can create a malicious service to run automatically in the background to add a user to the local Administrators group. This can be done with the sc.exe utility for instance. Where an adversary decides to create a remote service on a DC.

- **Adding user to local Administrators group via Restricted Groups**

Accounts that are part of groups like Backup Operators and Server Operators have the ability to copy a GPO and restore it reverse, which means that once they have a GPO copied, let's say the Default Domain Controllers Policy. They can start modify it, and then store it reverse. This allows an attacker for example elevate their privileges from Backup/Server Operators to a Domain Admin. Another option for an attacker is of course is to directly modify the GPO.

Review the following path:

\dc02\SYSVOL\contoso.local\Policies{6AC1786C-016F-11D2-945F-00C04fB984F9}\MACHINE\Microsoft\Windows NT\SecEdit

The GptTmpl file is similar to the User Right Assignments of a DC.

Result

At the sample result, we can see that we added a user via restricted groups to the local Administrators group of a Domain Controller.

- **Permissions delegated on the Domain Naming Context**

Effective permissions that have been delegated on the Domain Object can allow an adversary to replicate credentials from every account in a domain. Every ACL that allows an adversary to grant the following two permissions:

1. DS-Replicate-Get-Changes
2. DS-Replicate-Get-Changes-All

Should be treated as a DA or equivalent.

Command

```
$DNC= [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
$DNC = [adsi]"LDAP://$DNC"
$DNC.psbase.get_ObjectSecurity().getAccessRules($true, $true,
[system.security.principal.NtAccount])
```

Result

At the sample result, we can see “EvilUser” with the two mentioned permissions. Every ACE that is not a Domain Admins, Enterprise Admins, Administrators, and the AAD Connect Sync account should be excluded from the results. Once you see a different ACE that has effective permissions to modify the Domain Object. It may indicate that a malicious configuration has been configured on the Domain Root object.

- **Permissions delegated on the MicrosoftDNS container**

By default, DnsAdmins has GenericAll on the MicrosoftDNS container. It is publicly known that DnsAdmins can escalate their privileges to Domain Admin. Not going into the details, but the blog post can be found here (https://web.archive.org/web/20210623173110/https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/from-dnsadmins-to-system-to-domain-compromise).

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

Command

```
$ADSI=[ADSI]"LDAP://CN=MicrosoftDNS,CN=System,DC=contoso,DC=local"
$ADSI.psbase.get_ObjectSecurity().getAccessRules($true, $true,[system.security.principal.
NtAccount])
```

Result

At the sample result, we can see an ACE with GenericAll on the MicrosoftDNS container. Besides of DnsAdmins, Domain Admins, Enterprise Admins, SYSTEM, Administrators. No ACE should have GenericWrite or equivalent on this container, but if you do discover an ACE. There is a configuration that shouldn't be there in the first place.

- **Fake Domain Controller machine account**

An adversary can remain persistent by creating a "fake" Domain Controller machine account in AD, and then modify the userAccountControl value to **8192**, to let AD think it's a legitimate Domain Controller. Accounts with DS-Install-Replica on the Domain Root Object can add this value to a machine account.

If we now run a simple LDAP query to gather a list of all the Domain Controllers. It will return 3 results, but one machine account is not a legitimate DC.

Once the adversary decides to authenticate as that fake Domain Controller machine account. We can start executing a DCSync attack and get all the credentials as the machine account.

How to investigate this?

The first thing we have to do is run a simple LDAP query to gather a list of all the Domain Controllers. In order to do so, we can use the ADSI accelerator that is available on every domain machine.

Command

```
([adsisearcher]'(&(objectCategory=computer)(primaryGroupID=516))').FindAll()
```

Result

At the sample result, we can see 3 results.

Every Domain Controller must have a DRS RPC Interface GUID to function properly, which contains the following string: **E3514235-4B06-11D1-AB04-00C04FC2DCD2**

If we now run the following another LDAP query…

Command

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

Result

At the sample result, we now can see only 2 machine accounts. This is not a silver bullet way to discover fake DC machine accounts, because once an adversary already has DA or equivalent. They can just add that SPN. However, the chances may be low that they are doing it.

- **Resource Based Constrained Delegation (RBCD) on DC machine accounts**

An adversary 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.

A Domain Controller should NEVER have a value at the **msDs-AllowedToActOnBehalfOfOtherIdentity** attribute, so if you discover this. It is likely a backdoor.

Command

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

Result

At the sample result, we can see a machine that has a value at the **msDS-AllowedToActOnBehalfOfOtherIdentity** attribute.

- **Shadow Credentials – Key Trust Account**

Elad Shamir 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 blog post can be read here. (https://web.archive.org/web/20210623173110/https://posts.specterops.io/shadow-credentials-abusing-key-trust-account-mapping-for-takeover-8ee1a53566ab) 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:

> **Source:** https://posts.specterops.io/shadow-credentials-abusing-key-trust-account-mapping-for-takeover-8ee1a53566ab (https://web.archive.org/web/20210623173110/https://posts.specterops.io/shadow-credentials-abusing-key-trust-account-mapping-for-takeover-8ee1a53566ab)

Here is an example where we are executing this attack on a DC machine account.

Command

This LDAP query will look if there are any accounts that have a value in the msDS-KeyCredentialLink attribute.

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

Result

At the sample result, we can see that one machine account that has a value being set 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.

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

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, which looks like this:

In this example, we have now got the NT hash of the DC03$ machine account. We can do a DCSync attack with the DC03$ account, which probably will evade detection. Because it is a legitimate Domain Controller.

An adversary can for example disable automatically password change by setting a registry value. This value needs to be set on a Domain Controller.

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

Command

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 after 30 days. 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.

- **Golden Ticket Attack**

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).

An adversary already needs to be a DA or equivalent to execute this attack.

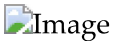
Command

Once an organization got hit. It is important to rotate the KRBTGT account password twice. This command will see when the last time, the KRBTGT account was rotated.

```
$user = [ADSI]'LDAP://CN=krbtgt,CN=Users,DC=yourdomainname,DC=com';
[datetime]::FromFileTime($user.ConvertLargeIntegerToInt64($user.pwdLastSet.value))
```

Result

At the sample result, we can see the date of when the password was rotated.



Summary

This blog post contains a list of Windows & Active Directory backdoors, that adversaries can use to remain persistent in an environment. It is good to be aware of the different backdoors that may exists in an environment, when doing incident response. It doesn't matter if you were able to remove two backdoors, when there was still one left.

An important disclaimer is that this list does not include all the potential backdoors that are out there. However, it does provide a great foundation for DFIR analysts to look out for certain things.

- Windows OS
- Active Directory