Kerberos Delegation


bloggingforlogging.com/2021/11/03/kerberos-delegation

November 3, 2021

When authenticating against a server across the network a common problem that people encounter is the inability to access downstream servers like a file share. This is because the network session that is running the code does not have access to the account's secret to regenerate the network tokens required to access that downstream server. This is commonly known as the double-hop problem in WinRM and is common enough to warrant its own docs to demonstrate the problem and mention workarounds. Some of those workarounds are:

- Use CredSSP authentication
- Use Kerberos delegation
- Set up a Just Enough Administration (JEA) endpoint that runs as a domain account
- Pass through the username/password on commands that attempt to replicate how thing would run locally

The Kerberos delegation workaround comes in 3 flavours:

- Resource-based Constrained
- Constrained
- Unconstrained

This doc will explain these 3 types and how to set them up. It will also cover how these can be used from Ansible when using the psrp connection plugin.

The following principals will be referenced in this doc to illustrate the delegation scenarios:

- Client `user@DOMAIN.COM`
- WinRM server `ServerA`
- SMB server `ServerB`

In this scenario a user `user@DOMAIN.COM` will be connecting to `ServerA` using WinRM PSRemoting trying to access files in `ServerB`. In a PowerShell script this will look like:

```
$cred = Get-Credential user@DOMAIN.COM
Invoke-Command ServerA -ScriptBlock {
Get-ChildItem -Path \\ServerB\share\folder
} -Credential $cred
```

In a normal environment `user@DOMAIN.COM` will be able to connect to `ServerA` using Kerberos but due to the double-hop problem it is unable to then access the fileshare on `ServerB`. Running that command without any delegation set up will result in the following error:

Access is denied

+ CategoryInfo : PermissionDenied: (\\ServerB\share\folder:String) [Get-ChildItem], UnauthorizedAccessException
+ FullyQualifiedErrorId : ItemExistsUnauthorizedAccessError,Microsoft.PowerShell.Commands.GetChildItemCommand
+ PSComputerName : ServerA
Cannot find path '\\ServerB\share\folder' because it does not exist.
+ CategoryInfo : ObjectNotFound: (\\ServerB\share\folder:String) [Get-ChildItem], ItemNotFoundE
xception
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.GetChildItemCommand
+ PSComputerName : ServerA

One important note before going into the delegation types are that any account marked as `Account is sensitive and cannot be delegated` cannot use any of these delegation scenarios. Windows AD will not permit a service to delegate these accounts so this is an important option to set for highly privileged accounts like `Domain Admins`.

Resource-based Constrained Delegation

Resource-based constrained delegation was introduced with Server 2012 as a way for the target service to specify what principals can delegate to. This is in contrast with the original constrained delegation method mentioned below where AD designates what target principals a service can delegate to. In a practical sense it means that `ServerB` specifies that `ServerA` can delegate a Kerberos credential to it. Even more importantly it does not require sensitive rights in the domain to set up the trusted source principals allowing the service itself to configure itself.

To set this up for the scenario mentioned above run the following in PowerShell:

```
# This is the host Kerberos authenticates with first
# I.e. the host that is allowed to delegate to the target
$host1 = Get-ADComputer -Identity ServerA
# This is the host where the delegation is configured
# I.e. the host the double-hop scenario is trying to connect to
$host2 = Get-ADComputer -Identity ServerB
# Grant the delegation on the AD Object.
Set-ADComputer $host2 -PrincipalsAllowedToDelegateToAccount $host1
```

Internally these details are stored on the `msDS-AllowedToActOnBehalfOfOtherIdentity` attribute as a security descriptor. The Attribute Editor in `dsa.msc` can show the raw value but it's easier to see existing allowed principals with:

```
$property = 'msDS-AllowedToActOnBehalfOfOtherIdentity'
(Get-ADComputer -Identity ServerB -Properties $property |
Select-Object -ExpandProperty $property
).Access
# ActiveDirectoryRights : GenericAll
```

```
# InheritanceType : None
# ObjectType : 00000000-0000-0000-0000-000000000000
# InheritedObjectType : 00000000-0000-0000-0000-000000000000
# ObjectFlags : None
# AccessControlType : Allow
# IdentityReference : DOMAIN\SERVERA$
# IsInherited : False
# InheritanceFlags : None
# PropagationFlags : None
```

In this scenario both the initial host and final destination host are running as the computer account in AD so `Get-ADComputer` is used. If accessing a service that is running under a separate domain account then the cmdlet `Get-ADUser` and `Set-ADUser` should be used instead. The value of `-PrincipalsAllowedToDelegateToAccount` accepts either `$null` to remove all existing principals or a list of accounts. This list will replace any existing value so make sure to retrieve the existing principals before adding new ones. There are more example of this which can be found here.

Once this has been set the Kerberos cache the front facing host should be reset. This will happen naturally after around 15 minutes but it can be forced with a reboot or by using `klist.exe`. In this scenario the following should be run on `ServerA` to reset it's cache `klist purge -li 0x3e7`.

From there things should just work without any further configuration on the client:

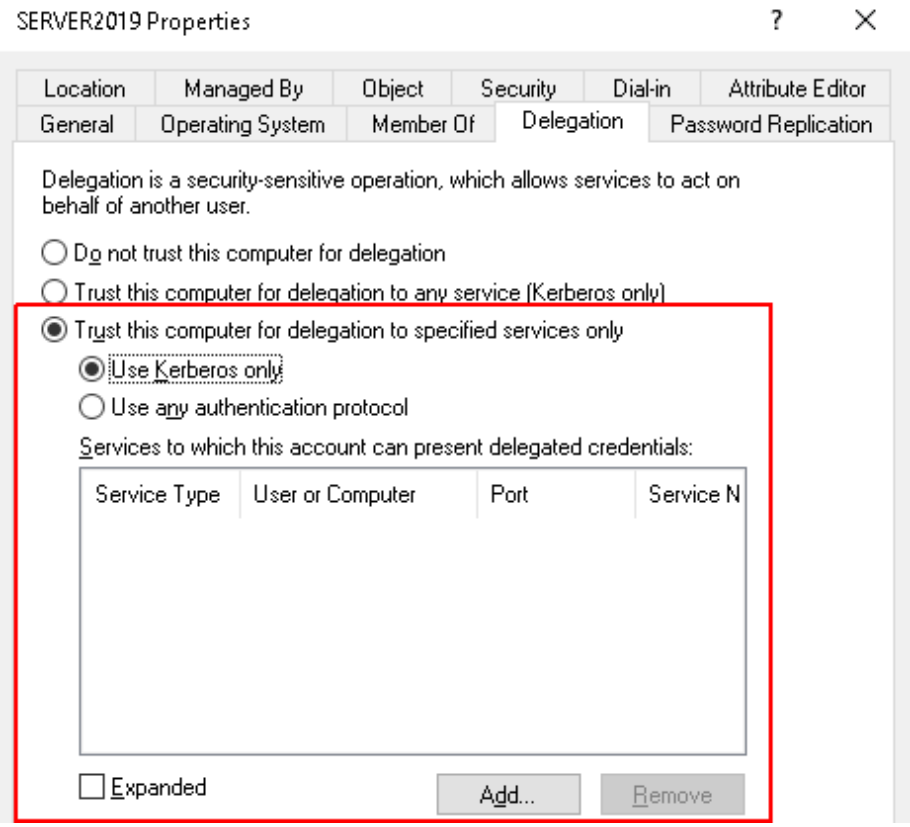
```
$cred = Get-Credential user@DOMAIN.COM
Invoke-Command ServerA -ScriptBlock {
Get-ChildItem -Path \\ServerB\share\folder
} -Credential $cred
# Directory: \\ServerB\share\folder
#
#
# Mode LastWriteTime Length Name PSComputerName
# ---- ------------- ------ ---- --------------
# -a---- 11/3/2021 11:16 AM 0 file.txt ServerB
```

The same applies to Ansible, there's no extra setting that needs to be set, just authenticate with Kerberos and things will work. Resource-based constrained delegation will even work when the client authenticated with another protocol, like NTLM. This means the initial WinRM connection could have been completed with NTLM and the remote session is able to use the delegated context to authenticate to `ServerB`.

Because the server added to `PrincipalsAllowedToDelegateToAccount` is allowed to delegate any account this can be a very powerful attack vector. It means that if anything compromised `ServerA` it is allowed to present to be any domain account when it talks to `ServerB`. The exception to this rule are accounts that are marked as `sensitive and account be delegated` as they do not allow themselves to be delegated at all.

Constrained Delegation

Constrained delegation was introduced with Server 2003 and it is used to specify what services a service can delegate credentials to. This is the opposite of resource-based delegation which is where the target host specifies who can delegate to it. In the active directory editor constrained delegation is configured with the `Trust this computer for delegation to specified services only` option on the host you are delegating from.



In this panel there are 2 options to further control when delegation can occur

- `Use Kerberos only` – do not allow protocol transition
- `Use any authentication protocol` – allows protocol transition

Protocol transition in this case allows clients to authenticate with any protocol and have the service delegate those credentials to the subsequent host. For example allowing any authentication protocol will allow a client to authenticate using NTLM and the service will then retrieve the Kerberos ticket itself for further delegation. By disabling protocol transition only clients that have authenticated with Kerberos initially will be delegated to the hosts specified.

Theoretically this means that `Use Kerberos only` should work with WinRM when using Kerberos but that does not seem to be the case. This requires further investigation but I've only ever been able to get this to work with the `Use any authentication protocol` option being set. At a guess WinRM is set up in a way that requires protocol transition for using delegation like this.

The values added correlate to the Service Principal Name (SPN) of the hosts that can be delegated to. In the example above, accessing the file share over SMB will use the SPN `cifs/ServerB` so that is what is added to the `ServerA` computer account in Active Directory. The service portion (`cifs`) is highly dependent on the desired delegation target and what SPN that host is registered to.

The settings for constrained delegation are stored in 2 attributes in AD:

- `userAccountControl`
- Bitmask field that contains multiple settings
- `UF_TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION 0x1000000` is set when `Use any authentication protocol is set
- `msDS-AllowedToDelegateTo`
- List of SPNs that the account is allowed to delegate to
- The SPN is in the form `service/hostname` and typically contains an entry for both the netbios and DNS name
- If the attribute is not set then constrained delegation is not enabled

To allow `ServerA` to delegate to `ServerB` for SMB using any authentication protocol the following PowerShell script can be used:

```
$spnListProp = 'msDS-AllowedToDelegateTo'
# This is the initial host the client connects to
$host1 = Get-ADComputer -Identity ServerA -Properties $spnListProp
# Adds the services ServerA is allowed to delegate to
Set-ADComputer $host1 -Add @{
"$spnListProp" = @('cifs/ServerB', 'cifs/ServerB.domain.com')
}
# Enabled protocol transition on the constrained delegation settings
$adControlParams = @{
# Sets the "Use any authentication protocol"
TrustedToAuthForDelegation = $true
# Ensure unconstrained delegation is not enabled - will override the
# constrained settings
TrustedForDelegation = $false
}
Set-ADAccountControl $host1 @adControlParams
```

Like with resource-based constrained delegation once the settings have been applied the initial service principal needs to update it's cache. This can be done by waiting for 15 minutes, rebooting the host, or running `klist purge -li 0x3e7`.

Once the changes have propagated the command is not able to succeed:

```
$cred = Get-Credential user@DOMAIN.COM
Invoke-Command ServerA -ScriptBlock {
Get-ChildItem -Path \\ServerB\share\folder
} -Credential $cred
```

```
# Directory: \\ServerB\share\folder
#
#
# Mode LastWriteTime Length Name PSComputerName
# ---- ------------- ------ ---- --------------
# -a---- 11/3/2021 11:16 AM 0 file.txt ServerB
```

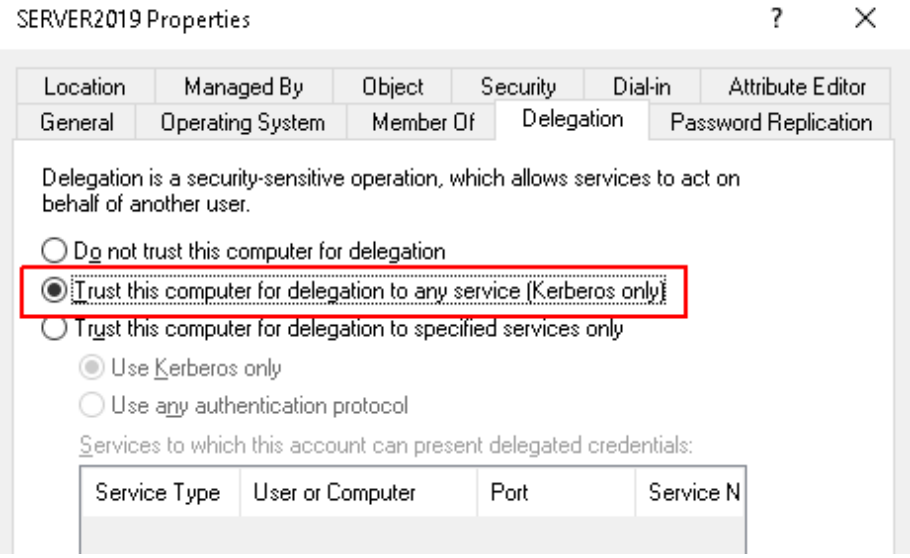
Like with resource-based constrained delegation there are no client settings that need to be configured. This means that Ansible is able to automatically delegate its credentials without further changes needed on its end. Because the `Use any authentication protocol` seems to be required for WinRM the client can authenticate using another protocol, like NTLM, and still be able to delegate to the next host.

This type of constrained delegation can be dangerous as it allows the host with the delegation settings to authenticate as any user (except the sensitive marked accounts) to the SPNs specified. This means if `ServerA` was compromised it will be able to present to be any user using SMB authentication on the host specified. Using the SMB example a compromised host will be able to use a PSExec style method to run a process as any domain computer on the delegated hosts specified. If allowing delegation to an `LDAP` SPN then the compromised host is able to do a DC sync and retrieve further credentials from there.

Unconstrained Delegation

Unconstrained delegation is the last type of Kerberos delegation that can be used. This type of delegation will allow the host to use the same Kerberos Ticket Granting Ticket (TGT) to re-authenticate itself against any subsequent hosts. Because it involves forwarding the Kerberos TGT from the client to the server it can only work when the client authenticates itself with Kerberos in the first place.

In the active directory editor constrained delegation is configured with the `Trust this computer for delegation to any service (Kerberos only)` option on the host you are delegating from.



This option is stored under the `userAccountControl` property under the bit mask `UF_TRUSTED_FOR_DELEGATION 0x80000`.
Setting this in PowerShell is done like so:

```
$host1 = Get-ADComputer -Identity ServerA -Properties $spnListProp
# Clear constrained delegation rules if any are set
Set-ADComputer $host1 -Clear 'msDS-AllowedToDelegateTo'
# Enable unconstrained delegation
Set-ADAccountControl -Identity $host1 -TrustedForDelegation $true
```

Once the settings have been enabled, the cache of the initial service principal needs to be cleared. The client's cache should also be cleared with `klist purge` on Windows or `kdestroy` on Linux. This is because unconstrained delegation changes the service ticket the client sends to the service and it cannot use whatever was cached beforehand.
Once cleared the PowerShell command will work just like the other scenarios.

```
$cred = Get-Credential user@DOMAIN.COM
Invoke-Command ServerA -ScriptBlock {
Get-ChildItem -Path \\ServerB\share\folder
} -Credential $cred
# Directory: \\ServerB\share\folder
#
#
# Mode LastWriteTime Length Name PSComputerName
# ---- ------------- ------ ---- --------------
# -a---- 11/3/2021 11:16 AM 0 file.txt ServerB
```

Getting this to work from Ansible requires 2 things:

- The TGT must be marked as forwardable
- The variable `ansible_psrp_negotiate_delegate` to be `True`

When using Ansible to get the TGT (explicit credentials defined) then the TGT will always be requested as forwardable. If calling `kinit` manually to get the TGT then either `-f` must be used or `forwardable = true` is set in `/etc/krb5.conf` under `[libdefault]`.

Once both conditions are met then Ansible will be able to utilise unconstrained delegation. One important note to call out is that the AD account does not need to be marked as `Trusted for delegation` for Ansible to utilise unconstrained delegation. Even if the AD computer account of the initial target (`ServerA`) is not trusted for delegation or has enabled any other delegation types Ansible will still send the TGT to the server for unconstrained delegation if it's requested by Ansible.

The reason for this is that Ansible uses GSSAPI and `ansible_psrp_negotiate_delegate=True` will ensure the `GSS_C_DELEG_FLAG` is set when exchanging the Kerberos tokens. Due to historical behaviour `GSS_C_DELEG_FLAG` will ignore the delegation settings in the AD computer account of the service. There does exist

`GSS_C_DELEG_POLICY_FLAG` which acts like `GSS_C_DELEG_FLAG` but honours the delegate policy set in AD but the libraries that Ansible uses do not expose this option. The only way to have Ansible honour the AD side policy is to set the following in the `/etc/krb5.conf` file:

[libdefault]
enforce_ok_as_delegate = true
This is a relatively recent option added in `1.18` of MIT Kerberos and what it does is treats `GSS_C_DELEG_FLAG` as `GSS_C_DELEG_POLICY_FLAG`. Heimdal implementations of GSSAPI have also added this setting but at present there is no released version that includes this option.

With unconstrained delegation, the server principal is able to use the forwarded TGT to authenticate with any service downstream which is very similar to how `CredSSP` works. It is generally recommended to not use unconstrained delegation as if the target host is compromised it is able to pretend to be the client when connecting to any other host.

Security Risks

While briefly mentioned in each section delegation does have it's own security risks. Any highly privileged account should have the `Account is sensitive and cannot be delegated` option set as these can never be delegated. Some key security risks of each delegation type are:

Unconstrained

- A compromised service principal can use the credentials of any user that has connected to it
- There are no limitations with what it can re-authenticate with
- For computer accounts this means any local admin can impersonate a non-sensitive account that has authenticated with it
- Because of this risk unconstrained delegation should generally be avoided

Constrained – Use any authentication protocol

- A compromised service principal can use the credentials of any domain user
- The user doesn't even need to have authenticated with the host
- The amount of damage that can be done dependends on the SPNs allowed for delegation
- `cifs` can be used for SMB and can access files and even start processes on hosts
- `host` essentially full control of the host in question
- `ldap` can be used to sync a DC and retrieve sensitive information
- `MSSQL` authenticate with an MS SQL database

Constrained – Use Kerberos only

- Same considerations as the above but,

- Instead of allowing it to delegate any domain account without authentication the account must first authenticate using Kerberos

Resource-based Constrained

- Users with write access to the `msDS-AllowedToActOnBehalfOfOtherIdentity` on a service principal can connect to that host as any user
- This allows malicious actors to become any domain account when authenticating against the compromised host
- Further damage is dependent on what the target service principal can do
- This attribute is not as locked down as `msDS-AllowedToDelegateTo` which could lead to easier infiltration methods
- One method, known as `Wagging the dog` is one to watch out for

A final note, I am not a security expert in these matters so please do your own research before trying one of these delegation models. There are numerous guides that go into a lot of detail around delegation and cover possible exploits as well as mitigations to use.