Active Directory Methodology

Basic overview

Active Directory serves as a foundational technology, enabling **network administrators** to efficiently create and manage **domains**, **users**, and **objects** within a network. It is engineered to scale, facilitating the organization of an extensive number of users into manageable **groups** and **subgroups**, while controlling **access rights** at various levels.

The structure of **Active Directory** is comprised of three primary layers: **domains**, **trees**, and **forests**. A **domain** encompasses a collection of objects, such as **users** or **devices**, sharing a common database. **Trees** are groups of these domains linked by a shared structure, and a **forest** represents the collection of multiple trees, interconnected through **trust relationships**, forming the uppermost layer of the organizational structure. Specific **access** and **communication rights** can be designated at each of these levels.

Key concepts within **Active Directory** include:

1. **Directory** – Houses all information pertaining to Active Directory objects.
2. **Object** – Denotes entities within the directory, including **users**, **groups**, or **shared folders**.
3. **Domain** – Serves as a container for directory objects, with the capability for multiple domains to coexist within a **forest**, each maintaining its own object collection.
4. **Tree** – A grouping of domains that share a common root domain.
5. **Forest** – The pinnacle of organizational structure in Active Directory, composed of several trees with **trust relationships** among them.

Active Directory Domain Services (AD DS) encompasses a range of services critical for the centralized management and communication within a network. These services comprise:

1. **Domain Services** – Centralizes data storage and manages interactions between **users** and **domains**, including **authentication** and **search** functionalities.
2. **Certificate Services** – Oversees the creation, distribution, and management of secure **digital certificates**.
3. **Lightweight Directory Services** – Supports directory-enabled applications through the **LDAP protocol**.
4. **Directory Federation Services** – Provides **single-sign-on** capabilities to authenticate users across multiple web applications in a single session.
5. **Rights Management** – Assists in safeguarding copyright material by regulating its unauthorized distribution and use.
6. **DNS Service** – Crucial for the resolution of **domain names**.

For a more detailed explanation check: **TechTerms - Active Directory Definition**

Kerberos Authentication

To learn how to **attack an AD** you need to **understand** really good the **Kerberos authentication process**. **Read this page if you still don't know how it works.**

Cheat Sheet

You can take a lot to https://wadcoms.github.io/ to have a quick view of which commands you can run to enumerate/exploit an AD.

Recon Active Directory (No creds/sessions)

If you just have access to an AD environment but you don't have any credentials/sessions you could:

- **Pentest the network:**
  - Scan the network, find machines and open ports and try to **exploit vulnerabilities** or **extract credentials** from them (for example, printers could be very interesting targets.
  - Enumerating DNS could give information about key servers in the domain as web, printers, shares, vpn, media, etc.
    - `gobuster dns -d domain.local -t 25 -w /opt/Seclist/Discovery/DNS/subdomain-top2000.txt`
  - Take a look to the General **Pentesting Methodology** to find more information about how to do this.
- **Check for null and Guest access on smb services** (this won't work on modern Windows versions):
  - `enum4linux -a -u "" -p "" <DC IP> && enum4linux -a -u "guest" -p "" <DC IP>`
  - `smbmap -u "" -p "" -P 445 -H <DC IP> && smbmap -u "guest" -p "" -P 445 -H <DC IP>`
  - `smbclient -U '%' -L //<DC IP> && smbclient -U 'guest%' -L //`
  - A more detailed guide on how to enumerate a SMB server can be found here:

139,445 - Pentesting SMB

- **Enumerate Ldap**
  - `nmap -n -sV --script "ldap* and not brute" -p 389 <DC IP>`
  - A more detailed guide on how to enumerate LDAP can be found here (pay **special attention to the anonymous access**):

389, 636, 3268, 3269 - Pentesting LDAP

- **Poison the network**
  - Gather credentials **impersonating services with Responder**
  - Access host by **abusing the relay attack**
  - Gather credentials **exposing fake UPnP services with evil-SSDP**
- **OSINT**:
  - Extract usernames/names from internal documents, social media, services (mainly web) inside the domain environments and also from the publicly available.

- If you find the complete names of company workers, you could try different AD **username conventions (read this)**. The most common conventions are: *NameSurname*, *Name.Surname*, *NamSur* (3letters of each), *Nam.Sur*, *NSurname*, *N.Surname*, *SurnameName*, *Surname.Name*, *SurnameN*, *Surname.N*, 3 *random letters and 3 random numbers* (abc123).
- Tools:
  - w0Tx/generate-ad-username
  - urbanadventurer/username-anarchy

User enumeration

- **Anonymous SMB/LDAP enum:** Check the **pentesting SMB** and **pentesting LDAP** pages.
- **Kerbrute enum**: When an **invalid username is requested** the server will respond using the **Kerberos error** code *KRB5KDC_ERR_C_PRINCIPAL_UNKNOWN*, allowing us to determine that the username was invalid. **Valid usernames** will illicit either the **TGT in a AS-REP** response or the error *KRB5KDC_ERR_PREAUTH_REQUIRED*, indicating that the user is required to perform pre-authentication.

Copy

```
./kerbrute_linux_amd64 userenum -d lab.ropnop.com --dc 10.10.10.10 usernames.txt
#From https://github.com/ropnop/kerbrute/releases

nmap -p 88 --script=krb5-enum-users --script-args="krb5-enum-
users.realm='DOMAIN'" <IP>
Nmap -p 88 --script=krb5-enum-users --script-args krb5-enum-
users.realm='<domain>',userdb=/root/Desktop/usernames.txt <IP>

msf> use auxiliary/gather/kerberos_enumusers

crackmapexec smb dominio.es  -u '' -p '' --users | awk '{print $4}' | uniq
```

- **OWA (Outlook Web Access) Server**

If you found one of these servers in the network you can also perform **user enumeration against it**. For example, you could use the tool **MailSniper**:

Copy

```
ipmo C:\Tools\MailSniper\MailSniper.ps1
# Get info about the domain
Invoke-DomainHarvestOWA -ExchHostname [ip]
# Enumerate valid users from a list of potential usernames
Invoke-UsernameHarvestOWA -ExchHostname [ip] -Domain [domain] -UserList
.\possible-usernames.txt -OutFile valid.txt
# Password spraying
Invoke-PasswordSprayOWA -ExchHostname [ip] -UserList .\valid.txt -Password
Summer2021
# Get addresses list from the compromised mail
```

```
Get-GlobalAddressList -ExchHostname [ip] -UserName [domain]\[username] -Password Summer2021 -OutFile gal.txt
```

You can find lists of usernames in **this github repo** **** and this one (**statistically-likely-usernames**).

However, you should have the **name of the people working on the company** from the recon step you should have performed before this. With the name and surname you could used the script **namemash.py** to generate potential valid usernames.

Knowing one or several usernames

Ok, so you know you have already a valid username but no passwords... Then try:

- **ASREPRoast**: If a user **doesn't have** the attribute *DONT_REQ_PREAUTH* you can **request a AS_REP message** for that user that will contain some data encrypted by a derivation of the password of the user.
- **Password Spraying**: Let's try the most **common passwords** with each of the discovered users, maybe some user is using a bad password (keep in mind the password policy!).
    - Note that you can also **spray OWA servers** to try to get access to the users mail servers.

Password Spraying / Brute Force

LLMNR/NBT-NS Poisoning

You might be able to **obtain** some challenge **hashes** to crack **poisoning** some protocols of the **network**:

Spoofing LLMNR, NBT-NS, mDNS/DNS and WPAD and Relay Attacks

NTML Relay

If you have managed to enumerate the active directory you will have **more emails and a better understanding of the network**. You might be able to to force NTML **relay attacks** **** to get access to the AD env.

Steal NTLM Creds

If you can **access other PCs or shares** with the **null or guest user** you could **place files** (like a SCF file) that if somehow accessed will t**rigger an NTML authentication against you** so you can **steal** the **NTLM challenge** to crack it:

Places to steal NTLM creds

Enumerating Active Directory WITH credentials/session

For this phase you need to have **compromised the credentials or a session of a valid domain account.** If you have some valid credentials or a shell as a domain

user, **you should remember that the options given before are still options to compromise other users**.

Before start the authenticated enumeration you should know what is the **Kerberos double hop problem.**

Kerberos Double Hop Problem

Enumeration

Having compromised an account is a **big step to start compromising the whole domain**, because you are going to be able to start the **Active Directory Enumeration:**

Regarding **ASREPRoast** you can now find every possible vulnerable user, and regarding **Password Spraying** you can get a **list of all the usernames** and try the password of the compromised account, empty passwords and new promising passwords.

- You could use the **CMD to perform a basic recon**
- You can also use **powershell for recon** which will be stealthier
- You ca also **use powerview** to extract more detailed information
- Another amazing tool for recon in an active directory is **BloodHound**. It is **not very stealthy** (depending on the collection methods you use), but **if you don't care** about that, you should totally give it a try. Find where users can RDP, find path to other groups, etc.
    - **Other automated AD enumeration tools are: AD Explorer, ADRecon, Group3r, PingCastle.**
- **DNS records of the AD** as they might contain interesting information.
- A **tool with GUI** that you can use to enumerate the directory is **AdExplorer.exe** from **SysInternal** Suite.
- You can also search in the LDAP database with **ldapsearch** to look for credentials in fields *userPassword* & *unixUserPassword*, or even for *Description*. cf. Password in AD User comment on PayloadsAllTheThings for other methods.
- If you are using **Linux**, you could also enumerate the domain using **pywerview**.
- You could also try automated tools as:
    - **tomcarver16/ADSearch**
    - **61106960/adPEAS**
- **Extracting all domain users**

  It's very easy to obtain all the domain usernames from Windows (`net user /domain` ,`Get-DomainUser` or `wmic useraccount get name,sid`). In Linux, you can use: `GetADUsers.py -all -dc-ip 10.10.10.110 domain.com/username` or `enum4linux -a -u "user" -p "password" <DC IP>`

Even if this Enumeration section looks small this is the most important part of all. Access the links (mainly the one of cmd, powershell, powerview and BloodHound), learn how to enumerate a domain and practice until you feel comfortable. During an assessment, this will be the key moment to find your way to DA or to decide that nothing can be done.

Kerberoast

Kerberoasting involves obtaining **TGS tickets** used by services tied to user accounts and cracking their encryption—which is based on user passwords—**offline**.

More about this in:

Kerberoast

Remote connexion (RDP, SSH, FTP, Win-RM, etc)

Once you have obtained some credentials you could check if you have access to any **machine**. For that matter, you could use **CrackMapExec** to attempt connecting on several servers with different protocols, accordingly to your ports scans.

Local Privilege Escalation

If you have compromised credentials or a session as a regular domain user and you have **access** with this user to **any machine in the domain** you should try to find your way to **escalate privileges locally and looting for credentials**. This is because only with local administrator privileges you will be able to **dump hashes of other users** in memory (LSASS) and locally (SAM).

There is a complete page in this book about **local privilege escalation in Windows** and a **checklist**. Also, don't forget to use **WinPEAS**.

Current Session Tickets

It's very **unlikely** that you will find **tickets** in the current user **giving you permission to access** unexpected resources, but you could check:

Copy

```
## List all tickets (if not admin, only current user tickets)
.\Rubeus.exe triage
## Dump the interesting one by luid
.\Rubeus.exe dump /service:krbtgt /luid:<luid> /nowrap
[IO.File]::WriteAllBytes("ticket.kirbi",
[Convert]::FromBase64String("<BASE64_TICKET>"))
```

NTML Relay

If you have managed to enumerate the active directory you will have **more emails and a better understanding of the network**. You might be able to to force NTML **relay attacks.**

Looks for Creds in Computer Shares

Now that you have some basic credentials you should check if you can **find** any **interesting files being shared inside the AD**. You could do that manually but it's a very boring repetitive task (and more if you find hundreds of docs you need to check).

Follow this link to learn about tools you could use.

Steal NTLM Creds

If you can **access other PCs or shares** you could **place files** (like a SCF file) that if somehow accessed will t**rigger an NTML authentication against you** so you can **steal** the **NTLM challenge** to crack it:

Places to steal NTLM creds

CVE-2021-1675/CVE-2021-34527 PrintNightmare

This vulnerability allowed any authenticated user to **compromise the domain controller**.

PrintNightmare

Privilege escalation on Active Directory WITH privileged credentials/session

For the following techniques a regular domain user is not enough, you need some special privileges/credentials to perform these attacks.

Hash extraction

Hopefully you have managed to **compromise some local admin** account using AsRepRoast, Password Spraying, Kerberoast, Responder including relaying, EvilSSDP, escalating privileges locally. Then, its time to dump all the hashes in memory and locally. **Read this page about different ways to obtain the hashes.**

Pass the Hash

Once you have the hash of a user, you can use it to **impersonate** it. You need to use some **tool** that will **perform** the **NTLM authentication using** that **hash**, **or** you could create a new **sessionlogon** and **inject** that **hash** inside the **LSASS**, so when any **NTLM authentication is performed**, that **hash will be used.** The last option is what mimikatz does. **Read this page for more information.**

Over Pass the Hash/Pass the Key

This attack aims to **use the user NTLM hash to request Kerberos tickets**, as an alternative to the common Pass The Hash over NTLM protocol. Therefore, this could be especially **useful in networks where NTLM protocol is disabled** and only **Kerberos is allowed** as authentication protocol.

Over Pass the Hash/Pass the Key

Pass the Ticket

In the **Pass The Ticket (PTT)** attack method, attackers **steal a user's authentication ticket** instead of their password or hash values. This stolen ticket is then used to **impersonate the user**, gaining unauthorized access to resources and services within a network.

Pass the Ticket

Credentials Reuse

If you have the **hash** or **password** of a **local administrato**r you should try to **login locally** to other **PCs** with it.

Copy

```
# Local Auth Spray (once you found some local admin pass or hash)
## --local-auth flag indicate to only try 1 time per machine
crackmapexec smb --local-auth 10.10.10.10/23 -u administrator -H
10298e182387f9cab376ecd08491764a0 | grep +
```

Note that this is quite **noisy** and **LAPS** would **mitigate** it.

MSSQL Abuse & Trusted Links

If a user has privileges to **access MSSQL instances**, he could be able to use it to **execute commands** in the MSSQL host (if running as SA), **steal** the NetNTLM **hash** or even perform a **relay attack**. Also, if a MSSQL instance is trusted (database link) by a different MSSQL instance. If the user has privileges over the trusted database, he is going to be able to **use the trust relationship to execute queries also in the other instance**. These trusts can be chained and at some point the user might be able to find a misconfigured database where he can execute commands. **The links between databases work even across forest trusts.**

MSSQL AD Abuse

Unconstrained Delegation

If you find any Computer object with the attribute ADS_UF_TRUSTED_FOR_DELEGATION and you have domain privileges in the computer, you will be able to dump TGTs from memory of every users that logins onto the computer. So, if a **Domain Admin logins onto the computer**, you will be able to dump his TGT and impersonate him using Pass the Ticket. Thanks to

constrained delegation you could even **automatically compromise a Print Server** (hopefully it will be a DC).

Unconstrained Delegation

Constrained Delegation

If a user or computer is allowed for "Constrained Delegation" it will be able to **impersonate any user to access some services in a computer**. Then, if you **compromise the hash** of this user/computer you will be able to **impersonate any user** (even domain admins) to access some services.

Constrained Delegation

Resourced-based Constrain Delegation

Having **WRITE** privilege on an Active Directory object of a remote computer enables the attainment of code execution with **elevated privileges**:

Resource-based Constrained Delegation

ACLs Abuse

The compromised user could have some **interesting privileges over some domain objects** that could let you **move** laterally/**escalate** privileges.

Abusing Active Directory ACLs/ACEs

Printer Spooler service abuse

Discovering a **Spool service listening** within the domain can be **abused** to **acquire new credentials** and **escalate privileges**.

Force NTLM Privileged Authentication

Third party sessions abuse

If **other users** **access** the **compromised** machine, it's possible to **gather credentials from memory** and even **inject beacons in their processes** to impersonate them. Usually users will access the system via RDP, so here you have how to performa couple of attacks over third party RDP sessions:

RDP Sessions Abuse

LAPS

LAPS provides a system for managing the **local Administrator password** on domain-joined computers, ensuring it's **randomized**, unique, and frequently **changed**. These passwords are stored in Active Directory and access is controlled

through ACLs to authorized users only. With sufficient permissions to access these passwords, pivoting to other computers becomes possible.

LAPS

Certificate Theft

Gathering certificates from the compromised machine could be a way to escalate privileges inside the environment:

AD CS Certificate Theft

Certificate Templates Abuse

If **vulnerable templates** are configured it's possible to abuse them to escalate privileges:

AD CS Domain Escalation

Post-exploitation with high privilege account

Dumping Domain Credentials

Once you get **Domain Admin** or even better **Enterprise Admin** privileges, you can **dump** the **domain database**: *ntds.dit*.

More information about DCSync attack can be found here.

More information about how to steal the NTDS.dit can be found here

Privesc as Persistence

Some of the techniques discussed before can be used for persistence. For example you could:

- Make users vulnerable to **Kerberoast**

  Copy

  ```
  Set-DomainObject -Identity <username> -Set
  @{serviceprincipalname="fake/NOTHING"}r
  ```

- Make users vulnerable to **ASREPRoast**

  Copy

  ```
  Set-DomainObject -Identity <username> -XOR
  @{UserAccountControl=4194304}
  ```

- Grant **DCSync** privileges to a user

Copy

```
Add-DomainObjectAcl -TargetIdentity "DC=SUB,DC=DOMAIN,DC=LOCAL" -PrincipalIdentity bfarmer -Rights DCSync
```

Silver Ticket

The **Silver Ticket attack** creates a **legitimate Ticket Granting Service (TGS) ticket** for a specific service by using the **NTLM hash** (for instance, the **hash of the PC account**). This method is employed to **access the service privileges**.

Silver Ticket

Golden Ticket

A **Golden Ticket attack** involves an attacker gaining access to the **NTLM hash of the krbtgt account** in an Active Directory (AD) environment. This account is special because it's used to sign all **Ticket Granting Tickets (TGTs)**, which are essential for authenticating within the AD network.

Once the attacker obtains this hash, they can create **TGTs** for any account they choose (Silver ticket attack).

Golden Ticket

Diamond Ticket

These are like golden tickets forged in a way that **bypasses common golden tickets detection mechanisms.**

Diamond Ticket

Certificates Account Persistence

Having certificates of an account or being able to request them is a very good way to be able to persist in the users account (even if he changes the password):

AD CS Account Persistence

Certificates Domain Persistence

Using certificates is also possible to persist with high privileges inside the domain:

AD CS Domain Persistence

AdminSDHolder Group

The **AdminSDHolder** object in Active Directory ensures the security of **privileged groups** (like Domain Admins and Enterprise Admins) by applying a standard **Access Control List (ACL)** across these groups to prevent unauthorized changes. However, this feature can be exploited; if an attacker modifies the AdminSDHolder's ACL to give full access to a regular user, that user gains extensive control over all privileged groups. This security measure, meant to protect, can thus backfire, allowing unwarranted access unless closely monitored.

More information about AdminDSHolder Group here.

DSRM Credentials

Inside every **Domain Controller (DC)**, a **local administrator** account exists. By obtaining admin rights on such a machine, the local Administrator hash can be extracted using **mimikatz**. Following this, a registry modification is necessary to **enable the use of this password**, allowing for remote access to the local Administrator account.

DSRM Credentials

ACL Persistence

You could **give** some **special permissions** to a **user** over some specific domain objects that will let the user **escalate privileges in the future**.

Abusing Active Directory ACLs/ACEs

Security Descriptors

The **security descriptors** are used to **store** the **permissions** an **object** have **over** an **object**. If you can just **make** a **little change** in the **security descriptor** of an object, you can obtain very interesting privileges over that object without needing to be member of a privileged group.

Security Descriptors

Skeleton Key

Alter **LSASS** in memory to establish a **universal password**, granting access to all domain accounts.

Skeleton Key

Custom SSP

Learn what is a SSP (Security Support Provider) here. You can create you **own SSP** to **capture** in **clear text** the **credentials** used to access the machine.\

Custom SSP

DCShadow

It registers a **new Domain Controller** in the AD and uses it to **push attributes** (SIDHistory, SPNs...) on specified objects **without** leaving any **logs** regarding the **modifications**. You **need DA** privileges and be inside the **root domain**. Note that if you use wrong data, pretty ugly logs will appear.

DCShadow

LAPS Persistence

Previously we have discussed about how to escalate privileges if you have **enough permission to read LAPS passwords**. However, these passwords can also be used to **maintain persistence**. Check:

LAPS

Forest Privilege Escalation - Domain Trusts

Microsoft views the **Forest** as the security boundary. This implies that **compromising a single domain could potentially lead to the entire Forest being compromised**.

Basic Information

A **domain trust** is a security mechanism that enables a user from one **domain** to access resources in another **domain**. It essentially creates a linkage between the authentication systems of the two domains, allowing authentication verifications to flow seamlessly. When domains set up a trust, they exchange and retain specific **keys** within their **Domain Controllers (DCs)**, which are crucial to the trust's integrity.

In a typical scenario, if a user intends to access a service in a **trusted domain**, they must first request a special ticket known as an **inter-realm TGT** from their own domain's DC. This TGT is encrypted with a shared **key** that both domains have agreed upon. The user then presents this TGT to the **DC of the trusted domain** to get a service ticket (**TGS**). Upon successful validation of the inter-realm TGT by the trusted domain's DC, it issues a TGS, granting the user access to the service.

Steps:

1. A **client computer** in **Domain 1** starts the process by using its **NTLM hash** to request a **Ticket Granting Ticket (TGT)** from its **Domain Controller (DC1)**.
2. DC1 issues a new TGT if the client is authenticated successfully.
3. The client then requests an **inter-realm TGT** from DC1, which is needed to access resources in **Domain 2**.
4. The inter-realm TGT is encrypted with a **trust key** shared between DC1 and DC2 as part of the two-way domain trust.

5. The client takes the inter-realm TGT to **Domain 2's Domain Controller (DC2)**.
6. DC2 verifies the inter-realm TGT using its shared trust key and, if valid, issues a **Ticket Granting Service (TGS)** for the server in Domain 2 the client wants to access.
7. Finally, the client presents this TGS to the server, which is encrypted with the server's account hash, to get access to the service in Domain 2.

Different trusts

It's important to notice that **a trust can be 1 way or 2 ways**. In the 2 ways options, both domains will trust each other, but in the **1 way** trust relation one of the domains will be the **trusted** and the other the **trusting** domain. In the last case, **you will only be able to access resources inside the trusting domain from the trusted one**.

If Domain A trusts Domain B, A is the trusting domain and B ins the trusted one. Moreover, in **Domain A**, this would be an **Outbound trust**; and in **Domain B**, this would be an **Inbound trust**.

Different trusting relationships

- **Parent-Child Trusts**: This is a common setup within the same forest, where a child domain automatically has a two-way transitive trust with its parent domain. Essentially, this means that authentication requests can flow seamlessly between the parent and the child.
- **Cross-link Trusts**: Referred to as "shortcut trusts," these are established between child domains to expedite referral processes. In complex forests, authentication referrals typically have to travel up to the forest root and then down to the target domain. By creating cross-links, the journey is shortened, which is especially beneficial in geographically dispersed environments.
- **External Trusts**: These are set up between different, unrelated domains and are non-transitive by nature. According to Microsoft's documentation, external trusts are useful for accessing resources in a domain outside of the current forest that isn't connected by a forest trust. Security is bolstered through SID filtering with external trusts.
- **Tree-root Trusts**: These trusts are automatically established between the forest root domain and a newly added tree root. While not commonly encountered, tree-root trusts are important for adding new domain trees to a forest, enabling them to maintain a unique domain name and ensuring two-way transitivity. More information can be found in Microsoft's guide.
- **Forest Trusts**: This type of trust is a two-way transitive trust between two forest root domains, also enforcing SID filtering to enhance security measures.
- **MIT Trusts**: These trusts are established with non-Windows, RFC4120-compliant Kerberos domains. MIT trusts are a bit more specialized and cater to environments requiring integration with Kerberos-based systems outside the Windows ecosystem.

Other differences in trusting relationships

- A trust relationship can also be **transitive** (A trust B, B trust C, then A trust C) or **non-transitive**.
- A trust relationship can be set up as **bidirectional trust** (both trust each other) or as **one-way trust** (only one of them trust the other).

Attack Path

1. **Enumerate** the trusting relationships
2. Check if any **security principal** (user/group/computer) has **access** to resources of the **other domain**, maybe by ACE entries or by being in groups of the other domain. Look for **relationships across domains** (the trust was created for this probably).
    1. kerberoast in this case could be another option.
3. **Compromise** the **accounts** which can **pivot** through domains.

Attackers with could access to resources in another domain through three primary mechanisms:

- **Local Group Membership**: Principals might be added to local groups on machines, such as the “Administrators” group on a server, granting them significant control over that machine.
- **Foreign Domain Group Membership**: Principals can also be members of groups within the foreign domain. However, the effectiveness of this method depends on the nature of the trust and the scope of the group.
- **Access Control Lists (ACLs)**: Principals might be specified in an **ACL**, particularly as entities in **ACEs** within a **DACL**, providing them access to specific resources. For those looking to dive deeper into the mechanics of ACLs, DACLs, and ACEs, the whitepaper titled “An ACE Up The Sleeve” is an invaluable resource.

Child-to-Parent forest privilege escalation

Copy

```
Get-DomainTrust

SourceName      : sub.domain.local    --> current domain
TargetName      : domain.local        --> foreign domain
TrustType       : WINDOWS_ACTIVE_DIRECTORY
TrustAttributes : WITHIN_FOREST       --> WITHIN_FOREST: Both in the same forest
TrustDirection  : Bidirectional       --> Trust direction (2ways in this case)
WhenCreated     : 2/19/2021 1:28:00 PM
WhenChanged     : 2/19/2021 1:28:00 PM
```

There are **2 trusted keys**, one for *Child --> Parent* and another one for *Parent --> Child*. You can the one used by the current domain them with:

Copy

```
Invoke-Mimikatz -Command '"lsadump::trust /patch"' -ComputerName dc.my.domain.local
Invoke-Mimikatz -Command '"lsadump::dcsync /user:dcorp\mcorp$"'
```

SID-History Injection

Escalate as Enterprise admin to the child/parent domain abusing the trust with SID-History injection:

SID-History Injection

Exploit writeable Configuration NC

Understanding how the Configuration Naming Context (NC) can be exploited is crucial. The Configuration NC serves as a central repository for configuration data across a forest in Active Directory (AD) environments. This data is replicated to every Domain Controller (DC) within the forest, with writable DCs maintaining a writable copy of the Configuration NC. To exploit this, one must have **SYSTEM privileges on a DC**, preferably a child DC.

Link GPO to root DC site

The Configuration NC's Sites container includes information about all domain-joined computers' sites within the AD forest. By operating with SYSTEM privileges on any DC, attackers can link GPOs to the root DC sites. This action potentially compromises the root domain by manipulating policies applied to these sites.

For in-depth information, one might explore research on Bypassing SID Filtering.

Compromise any gMSA in the forest

An attack vector involves targeting privileged gMSAs within the domain. The KDS Root key, essential for calculating gMSAs' passwords, is stored within the Configuration NC. With SYSTEM privileges on any DC, it's possible to access the KDS Root key and compute the passwords for any gMSA across the forest.

Detailed analysis can be found in the discussion on Golden gMSA Trust Attacks.

Schema change attack

This method requires patience, waiting for the creation of new privileged AD objects. With SYSTEM privileges, an attacker can modify the AD Schema to grant any user complete control over all classes. This could lead to unauthorized access and control over newly created AD objects.

Further reading is available on Schema Change Trust Attacks.

From DA to EA with ADCS ESC5

The ADCS ESC5 vulnerability targets control over Public Key Infrastructure (PKI) objects to create a certificate template that enables authentication as any user within the forest. As PKI objects reside in the Configuration NC, compromising a writable child DC enables the execution of ESC5 attacks.

More details on this can be read in From DA to EA with ESC5. In scenarios lacking ADCS, the attacker has the capability to set up the necessary components, as discussed in Escalating from Child Domain Admins to Enterprise Admins.

External Forest Domain - One-Way (Inbound) or bidirectional

Copy

```
Get-DomainTrust
SourceName      : a.domain.local   --> Current domain
TargetName      : domain.external  --> Destination domain
TrustType       : WINDOWS-ACTIVE_DIRECTORY
TrustAttributes :
TrustDirection  : Inbound          --> Inboud trust
WhenCreated     : 2/19/2021 10:50:56 PM
WhenChanged     : 2/19/2021 10:50:56 PM
```

In this scenario **your domain is trusted** by an external one giving you **undetermined permissions** over it. You will need to find **which principals of your domain have which access over the external domain** and then try to exploit it:

External Forest Domain - OneWay (Inbound) or bidirectional

External Forest Domain - One-Way (Outbound)

Copy

```
Get-DomainTrust -Domain current.local

SourceName      : current.local   --> Current domain
TargetName      : external.local  --> Destination domain
TrustType       : WINDOWS_ACTIVE_DIRECTORY
TrustAttributes : FOREST_TRANSITIVE
TrustDirection  : Outbound        --> Outbound trust
WhenCreated     : 2/19/2021 10:15:24 PM
WhenChanged     : 2/19/2021 10:15:24 PM
```

In this scenario **your domain** is **trusting** some **privileges** to principal from a **different domains**.

However, when a **domain is trusted** by the trusting domain, the trusted domain **creates a user** with a **predictable name** that uses as **password the trusted password**. Which means that it's possible to **access a user from the trusting domain to get inside the trusted one** to enumerate it and try to escalate more privileges:

External Forest Domain - One-Way (Outbound)

Another way to compromise the trusted domain is to find a **SQL trusted link** created in the **opposite direction** of the domain trust (which isn't very common).

Another way to compromise the trusted domain is to wait in a machine where a **user from the trusted domain can access** to login via **RDP**. Then, the attacker could inject code in the RDP session process and **access the origin domain of the victim** from there. Moreover, if the **victim mounted his hard drive**, from the **RDP session** process the attacker could store **backdoors** in the **startup folder of the hard drive**. This technique is called **RDPInception.**

RDP Sessions Abuse

Domain trust abuse mitigation

SID Filtering:

- The risk of attacks leveraging the SID history attribute across forest trusts is mitigated by SID Filtering, which is activated by default on all inter-forest trusts. This is underpinned by the assumption that intra-forest trusts are secure, considering the forest, rather than the domain, as the security boundary as per Microsoft's stance.
- However, there's a catch: SID filtering might disrupt applications and user access, leading to its occasional deactivation.

Selective Authentication:

- For inter-forest trusts, employing Selective Authentication ensures that users from the two forests are not automatically authenticated. Instead, explicit permissions are required for users to access domains and servers within the trusting domain or forest.
- It's important to note that these measures do not safeguard against the exploitation of the writable Configuration Naming Context (NC) or attacks on the trust account.

More information about domain trusts in ired.team.

AD -> Azure & Azure -> AD

Azure AD Connect - Hybrid IdentityHackTricks Cloud

Some General Defenses

Learn more about how to protect credentials here.\

Defensive Measures for Credential Protection

- **Domain Admins Restrictions**: It is recommended that Domain Admins should only be allowed to login to Domain Controllers, avoiding their use on other hosts.
- **Service Account Privileges**: Services should not be run with Domain Admin (DA) privileges to maintain security.

- **Temporal Privilege Limitation**: For tasks requiring DA privileges, their duration should be limited. This can be achieved by: `Add-ADGroupMember -Identity 'Domain Admins' -Members newDA -MemberTimeToLive (New-TimeSpan -Minutes 20)`

Implementing Deception Techniques

- Implementing deception involves setting traps, like decoy users or computers, with features such as passwords that do not expire or are marked as Trusted for Delegation. A detailed approach includes creating users with specific rights or adding them to high privilege groups.
- A practical example involves using tools like: `Create-DecoyUser -UserFirstName user -UserLastName manager-uncommon -Password Pass@123 | DeployUserDeception -UserFlag PasswordNeverExpires -GUID d07da11f-8a3d-42b6-b0aa-76c962be719a -Verbose`
- More on deploying deception techniques can be found at Deploy-Deception on GitHub.

Identifying Deception

- **For User Objects**: Suspicious indicators include atypical ObjectSID, infrequent logons, creation dates, and low bad password counts.
- **General Indicators**: Comparing attributes of potential decoy objects with those of genuine ones can reveal inconsistencies. Tools like HoneypotBuster can assist in identifying such deceptions.

Bypassing Detection Systems

- **Microsoft ATA Detection Bypass**:
  - **User Enumeration**: Avoiding session enumeration on Domain Controllers to prevent ATA detection.
  - **Ticket Impersonation**: Utilizing **aes** keys for ticket creation helps evade detection by not downgrading to NTLM.
  - **DCSync Attacks**: Executing from a non-Domain Controller to avoid ATA detection is advised, as direct execution from a Domain Controller will trigger alerts.

References

- http://www.harmj0y.net/blog/redteaming/a-guide-to-attacking-domain-trusts/
- https://www.labofapenetrationtester.com/2018/10/deploy-deception.html
- https://ired.team/offensive-security-experiments/active-directory-kerberos-abuse/child-domain-da-to-ea-in-parent-domain