## Theory

* Microsoft Active Directory Domain Services, often referred to as Active Directory (AD), is a service that allows system administrators to update and manage operating systems, applications, users, and data access on a large scale.
* Active Directory consists of several components. The most important component is the domain controller (DC). The domain controller is the hub and core of Active Directory because it stores all information about how the specific instance of Active Directory is configured.
* When an instance of Active Directory is configured, a domain is created with a name such as corp.com where corp is the name of the organization.
* System administrators can (and almost always do) organize these objects with the help of Organizational Units (OU)
* Objects in Active Directory are end-points and user object (Employe’s)
* OUs are comparable to file system folders in that they are containers used to store and group other.
* All AD objects contain attributes, which vary according to the type of object. For example, a user object may include attributes such as first name, last name, username, and password.
* An Active Directory environment has a very critical dependency on a Domain Name System (DNS) service. As such, a typical domain controller in an AD will also host a DNS server that is authoritative for a given domain.
* Aim : From a local admin to global domain admin access.
* Another way to gain control of a domain is to successfully compromise a domain controller since it may be used to modify all domain-joined computers or execute applications on them. Additionally.

## Practical

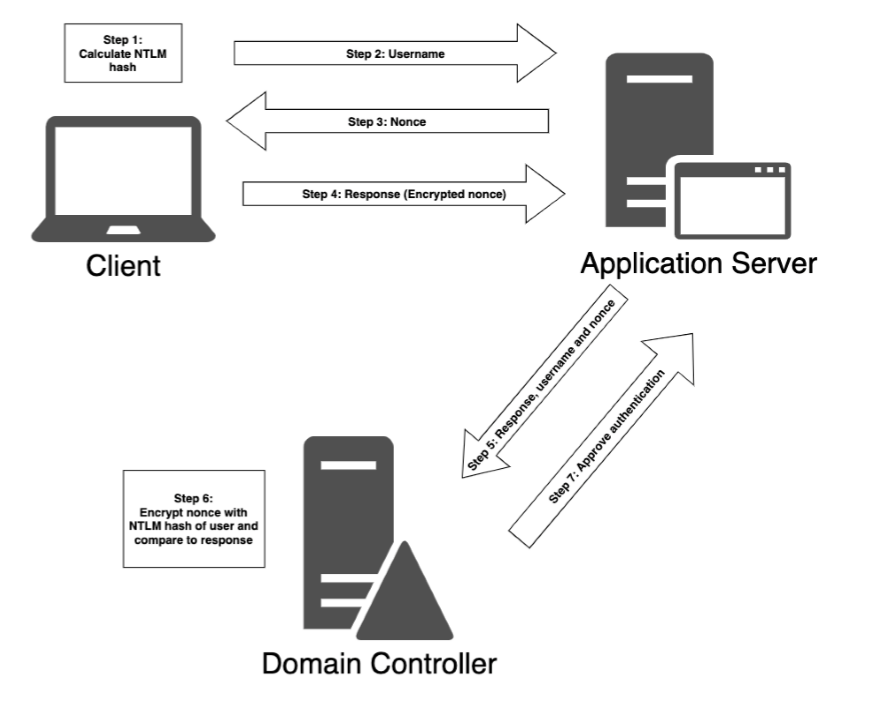
* net user, will enumerate all the user in the local computer.
* Net user /domain ,Adding the /domain flag will enumerate all users in the entire domain.
* net group /domain, prints all the groups in a domain.
* In AD, we can have a group as a member of another group, which means nesting can also happen between groups.
* A script will be created which will have the functionality to query the name of Primary domain controller, search active directory and display user accounts and then clean up the output for readability.
* This script relies on a few components. Specifically, we will use a DirectorySearcher object to query Active Directory using the Lightweight Directory Access Protocol (LDAP). which is a network protocol understood by domain controllers also used for communication with third-party applications. LDAP is an Active Directory Service Interfaces (ADSI) provider (essentially an API) that supports search functionality against an Active Directory. Our script will center around a very specific LDAP provider path614 that will serve as input to the DirectorySearcher .NET class.
* Therefore we will create a object of DirectorySearch .NetClass and use the protocol LDAP with a path to create the above functionality. The prototype path looks like
* LDAP://HostName[:PortNumber][/DistinguishedName] , we need a hostname of domain controller, and Distinguished name is a combination of commonname, organizationunit, domain; example CN=Litware,OU=Docs\, Adatum,DC=Fabrikam,DC=COM.
* **System.DirectoryServices.ActiveDirectory.Domain** is the class using which we can extract current domain information
* [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain() , GetCurrentDomain() is the method.
* Create LDAP path like “LDAP://PrimaryDomainController/DC=corp,DC=com”
* Create a searcher object like

**New-Object System.DirectoryServices.DirectorySearcher([ADSI]$SearchString)**

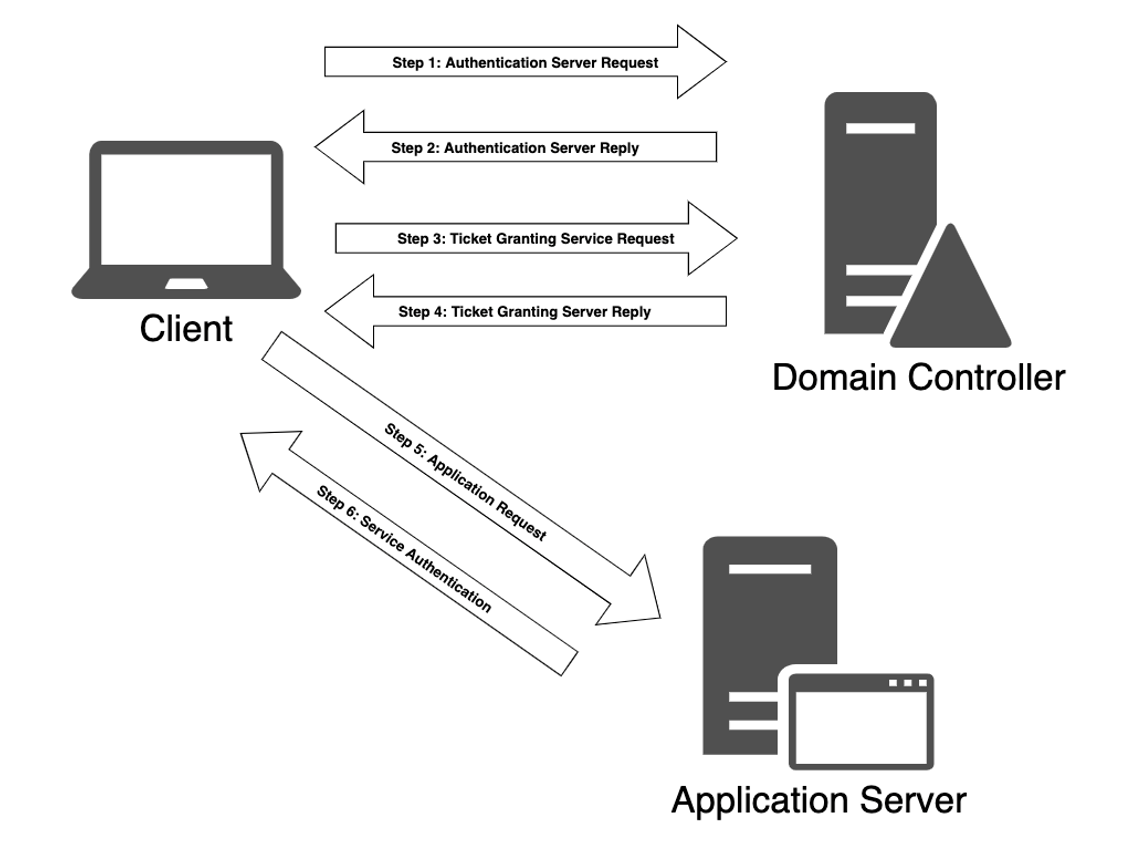
* To find out user accounts which are active, we can use two Windows functions –
* **NetWkstaUserEnum(admin priv) returns the list of all users logged on to a target workstation**
* **NetSessionEnum API(non-admin) returns a list of active user sessions on servers such as fileservers or domain controllers.**
* Aim: to find high value service accounts. We can do that using filters in the old script. "serviceprincipalname=\*http\*"
* **Cool.**

## Active Directory Authentication

#### NTLM Authentication

* NTLM authentication is used when a client authenticates to a server by IP address (instead of by hostname),
* if the user attempts to authenticate to a hostname that is not registered on the Active Directory integrated DNS server.
* Likewise, third-party applications may choose to use NTLM authentication instead of Kerberos authentication.
* 

Kerbrose authentication



1. Step 1 is “**AS\_REQ**”, it contains a timestamp encrypted with user’s password and username.

Encrpt(timestamp,user’s hash) + username

1. When the DC receives the request, it checks by attempting to decrpt the thingy and if the timestamp is unquie and not used before then it considers it a SUCCESS or a correct creds.
2. DC replies with Authentication Server Reply (**AS\_REP**), that contains a **session key** and a Ticket Granting Ticket (**TGT**). The session key is encrypted using the user’s password hash, and may be decrypted by the client and reused. The TGT contains information regarding the user, including **group\_memberships**, **the domain**, a **time stamp**, the **IP address** of the client, and the **session\_key**. And these are encrypted so that user can’t see them.
3. By default, the TGT will be valid for 10 hours, after which a renewal occurs. This renewal does not require the user to re-enter the password.
4. When the user wishes to access resources of the domain, such as a network share, an Exchange mailbox, or some other application with a registered service principal name, it must again contact the KDC.
5. When the client wants to access resources in the domain from a SPN (Service Principle Name) , a **TGS\_REQ** request is generated. It consist of **current user** and a **timestamp** (encrypted using the **session\_key**), the **SPN** of the resource, and the encrypted **TGT**.
6. **Ticket Granting Service Request (or TGS\_REQ) current user** , **timestamp (encrypted using the session key),** the **SPN** of the resource, and the **encrypted TGT.**
7. Next, the ticket granting service on the KDC receives the TGS\_REQ, and if the SPN exists in the domain, the TGT is decrypted using the secret key known only to the KDC. The session key is then extracted from the TGT and used to decrypt the username and timestamp of the request.
8. KDC performs several checks –
   1. The TGT must have a valid timestamp (no replay detected and the request has not expired).
   2. The username from the TGS\_REQ has to match the username from the TGT.
   3. The client IP address needs to coincide with the TGT IP address.
9. If this verification process succeeds, the ticket granting service responds to the client with a Ticket Granting Server Reply or TGS\_REP. This packet contains three parts:
   1. The SPN to which access has been granted.
   2. A session key to be used between the client and the SPN.
   3. A service ticket containing the username and group memberships along with the newly created session key.
10. The first two parts (SPN and session key) are encrypted using the session key associated with the creation of the TGT and the service ticket is encrypted using the password hash of the service account registered with the SPN in question.
11. Now, the Authentication process by DC is complete.
12. First, the client sends to the application server an application request or AP\_REQ , which includes the username and a timestamp encrypted with the session key associated with the service ticket along with the service ticket itself.
13. The application server decrypts the service ticket using the service account password hash and extracts the username and the session key. It then uses the latter to decrypt the username from the AP\_REQ.
14. If the AP\_REQ username matches the one decrypted from the service ticket, the request is accepted. Before access is granted, the service inspects the supplied group memberships in the service ticket and assigns appropriate permissions to the user, after which the user may access the requested service.

**Cached Credential Storage and Retrieval**

1. We can have TGT, TGS, password hashes extracted from the cached dumps.

**Service Account Attacks**

1. When requesting the service ticket from the domain controller, no checks are performed on whether the user has any permissions to access the service hosted by the service principal name. These checks are performed as a second step only when connecting to the service itself. This means that if we know the SPN we want to target, we can request a service ticket for it from the domain controller. Then, since it is our own ticket, we can extract it from local memory and save it to disk.
2. For example, we know that the registered SPN for the Internet Information Services web server in the domain is HTTP/CorpWebServer.corp.com. From PowerShell, we can use the KerberosRequestorSecurityToken class to request the service ticket.
3. The code segment we need is located inside the System.IdentityModel namespace, which is not loaded into a PowerShell instance by default. To load it, we use the Add-Type cmdlet with the AssemblyName argument.

**Low and Slow Password Guessing**

1. **Basic password guessing read form manual.**

**Active Directory Lateral Movement**

**Pass the Hash**

1. You know already
2. Command : pth-winexe -U offsec%aad3b435b51404eeaad3b435b51404ee:2892d26cdf84d7a70e2 eb3b9f05c425e //10.11.0.22 cmd

**Over Pass the Hash**

1. **Almost same as pass the hash**
2. **Just get hashes of all accounts in cache and use mimikatz to extract them.**
3. **GET access to TGT**

**Pass the Ticket**

1. **TGT many times not enough. We need to get access to service accounts. Using TGS we can do that. Create a forged TGS ticket which has groups of domain server.**

**Distributed Component Object Model**

The Microsoft Component Object Model (COM) is a system for creating software components that interact with each other. While COM was created for either same-process or cross-process interaction, it was extended to Distributed Component Object Model (DCOM) for interaction between multiple computers over a network. Specifically, we will leverage the Excel.Application DCOM object.

netsh -r **XOR-APP23** -u XOR\Adam -p shantewhite