Breaching Active Directory

we don't focus on the permissions associated with the account; thus, even a low-privileged account would be sufficient. We are just looking for a way to authenticate to AD, allowing us to do further enumeration on AD itself.

Methods that can be used to breach AD:

- NTLM Authenticated Services
- LDAP Bind Credentials
- Authentication Relays
- Microsoft Deployment Toolkit
- Configuration Files

OSINT : used to discover information that has been publicly disclosed.

You can search in Stackoverflow, Github, Haveibeenpwned & Dehashed.

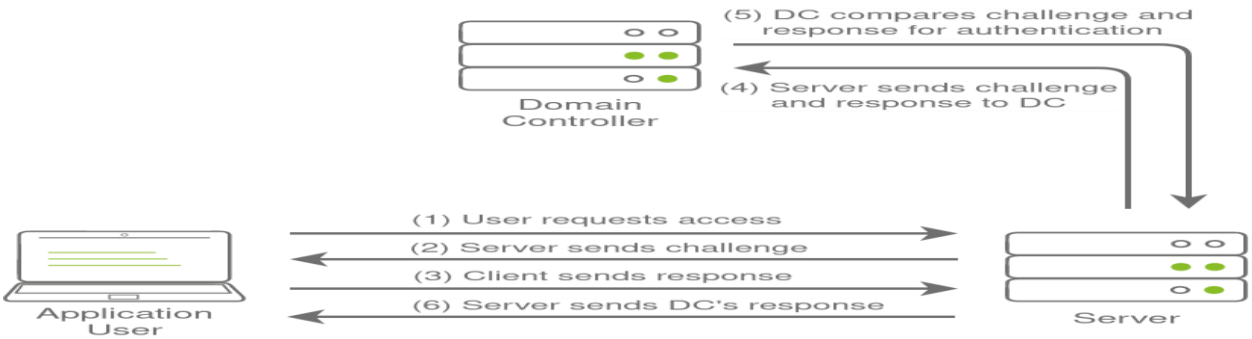
Phishing : excellent method to breach AD. Phishing usually entices users to either provide their credentials. Like malicious web page or ask them to run a specific application that would install a Remote Access Trojan (RAT) in the background.

NetNTLM

- services that use NetNTLM can also be exposed to the internet

- Internally-hosted Exchange (Mail) servers that expose an Outlook Web App (OWA) login portal.
- Remote Desktop Protocol (RDP) service of a server being exposed to the internet.
- Exposed VPN endpoints that were integrated with AD.
- Web applications that are internet-facing and make use of NetNTLM.

All authentication material is forwarded to a Domain Controller, This prevents the application from storing AD credentials, which should only be stored on a Domain Controller



Brute-force Login Attacks : Since most AD environments have account lockout configured, we won't be able to run a full brute-force attack. Instead, we need to perform a password spraying attack. Instead of trying multiple different passwords, which may trigger the account lockout mechanism, we choose and use one password and attempt to authenticate with all the usernames we have acquired.

Password Spraying : we can use python script in this situation.

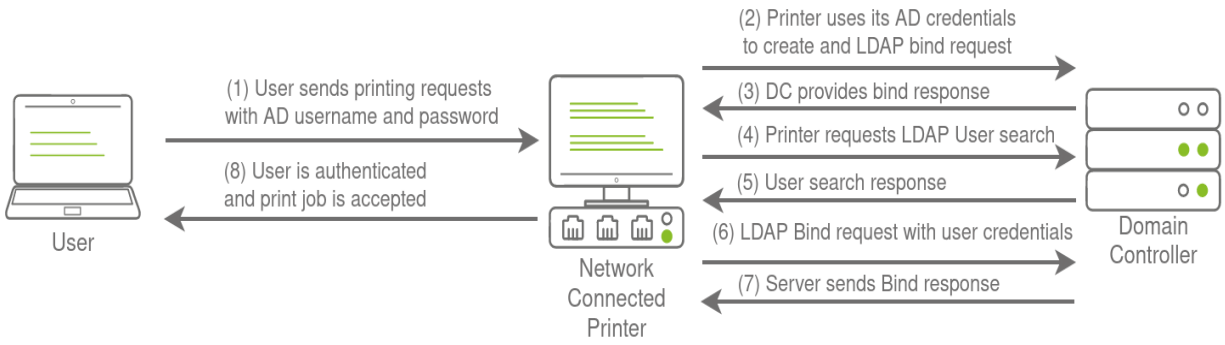
```
python ntlm_passwordspray.py -u <userfile> -f <fqdn> -p <password> -a <attackurl>
```

- **<userfile>** - Textfile containing our usernames - *"usernames.txt"*
- **<fqdn>** - Fully qualified domain name associated with the organisation that we are attacking.
- **<password>** - The password we want to use for our spraying attack.
- **<attackurl>** - The URL of the application that supports Windows Authentication.

- LDAP authentication is a popular mechanism with third-party (non-Microsoft) applications. Like,

- Gitlab
- Jenkins
- Custom-developed web applications
- Printers
- VPNs

The process of authentication through LDAP is shown below



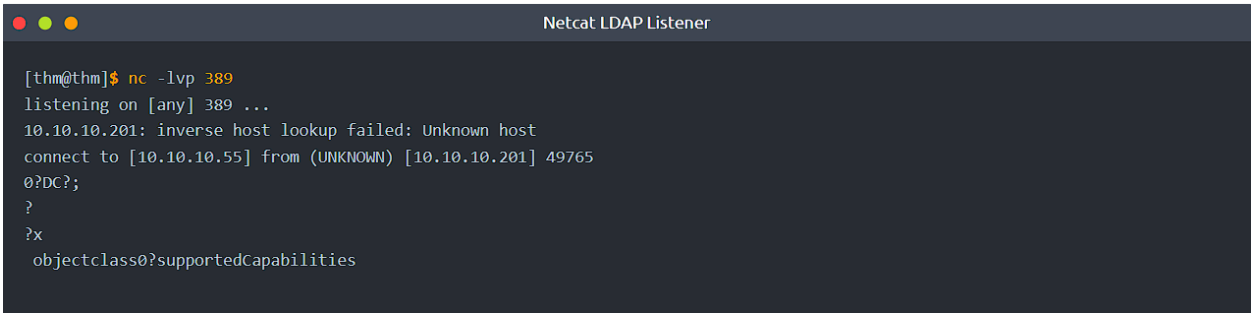
LDAP Pass-back Attacks: This is a common attack against network devices, such as printers, can be performed when we gain access to a device's configuration where the LDAP parameters are specified.

the credentials for these interfaces are kept to the default ones such as `admin:admin` or `admin:password`

we can alter the LDAP configuration, such as the IP or hostname of the LDAP server. In an LDAP Pass-back attack, we can modify this IP to our IP and then test the LDAP configuration.

Performing an LDAP Pass-back:

let's use a simple Netcat listener to test if we can get the printer to connect to us `nc -lvp 389`



Here we found a problem so we will use Rouge LDAP Server.

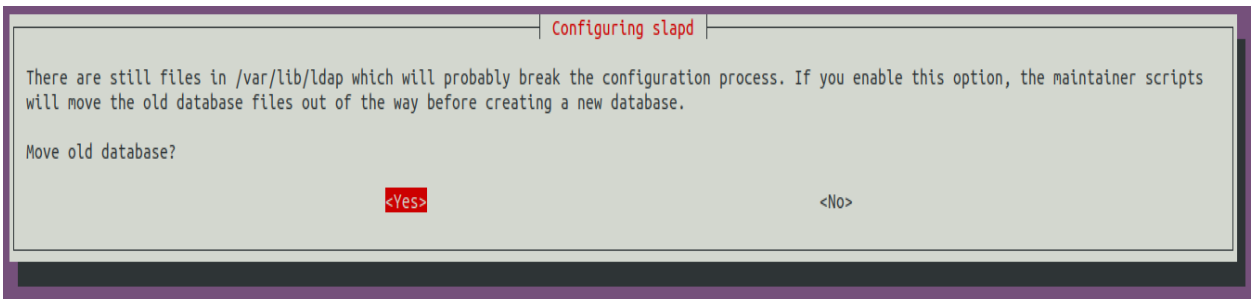
To host a rogue LDAP server, use this command.

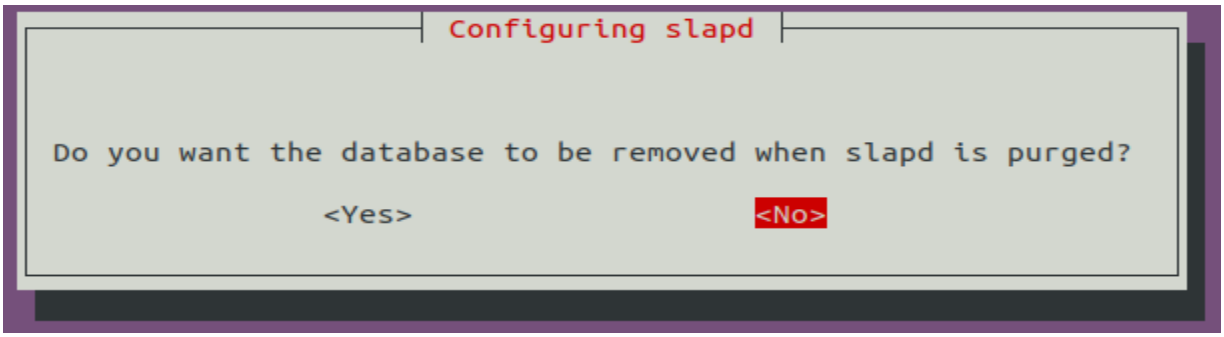
```
sudo apt-get update && sudo apt-get -y install slapd ldap-utils && sudo systemctl enable slapd
```

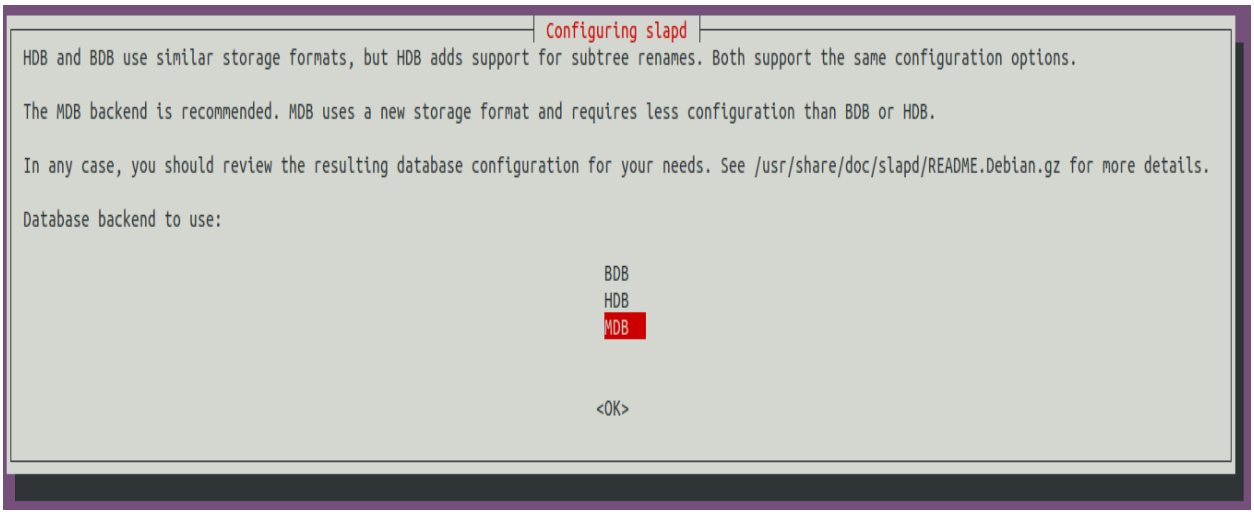
after installation, use this command.

```
sudo dpkg-reconfigure -p low slapd
```

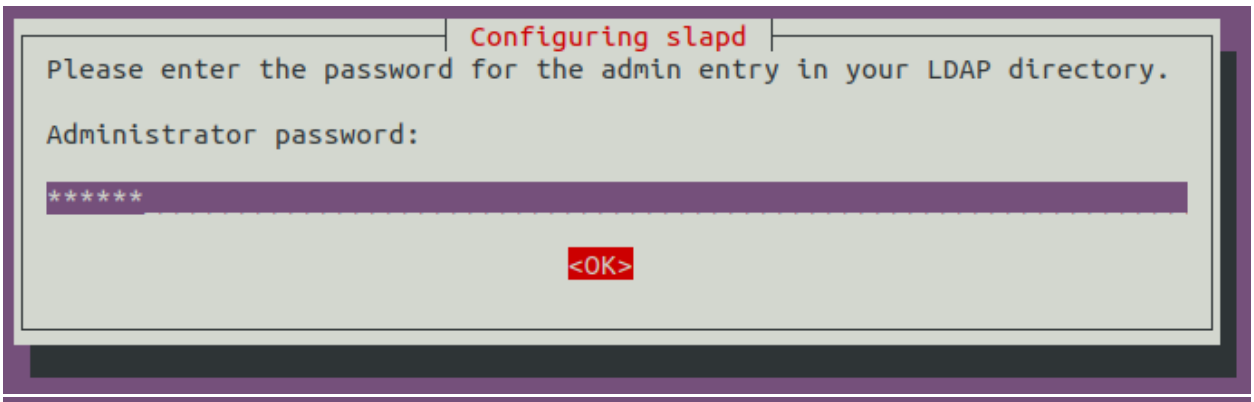
follow the photos bellow,

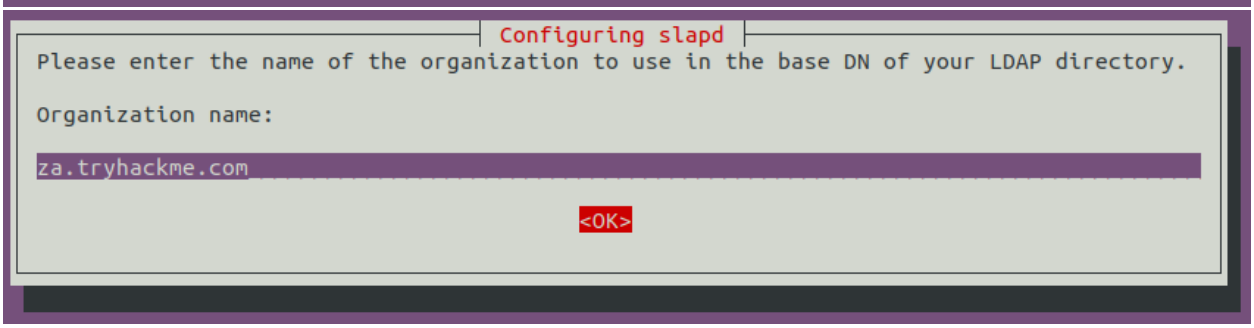


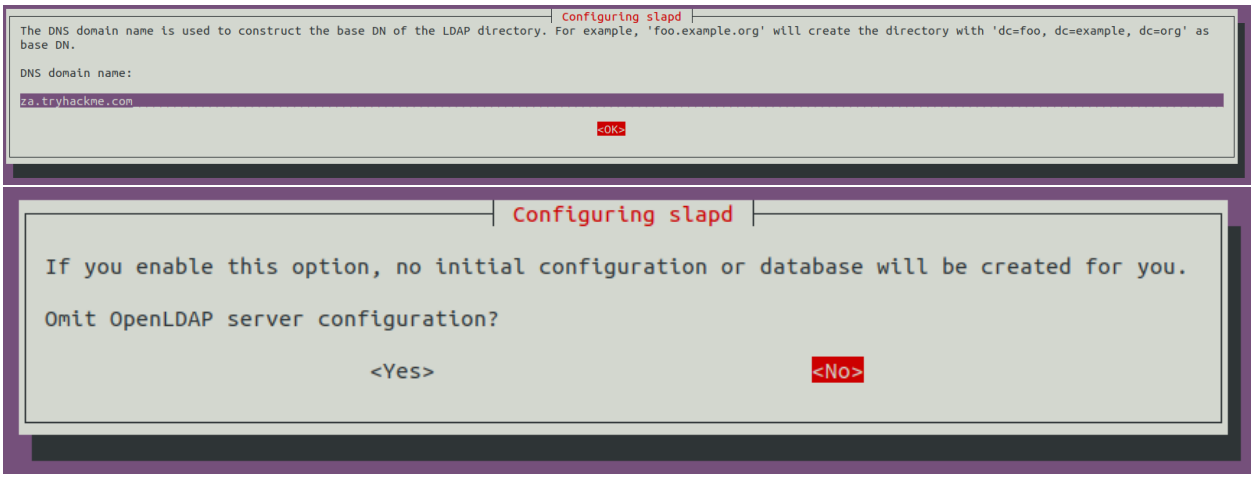




Use password123







Then we will create ldif file to downgrading the supported authentication mechanisms, We want to ensure that our LDAP server only supports PLAIN and LOGIN authentication methods.

olcSaslSecProps.ldif

```
#olcSaslSecProps.ldif
dn: cn=config
replace: olcSaslSecProps
olcSaslSecProps: noanonymous,minssf=0,passcred
```

- **olcSaslSecProps:** Specifies the SASL security properties
- **noanonymous:** Disables mechanisms that support anonymous login
- **minssf:** Specifies the minimum acceptable security strength with 0, meaning no protection.

use the ldif file to patch our LDAP server, use this command.

```
sudo ldapmodify -Y EXTERNAL -H ldapi:// -f ./olcSaslSecProps.ldif && sudo service slapd restart
```

We can verify that our rogue LDAP server's configuration has been applied with this command.

```
ldapsearch -H ldap:// -x -LLL -s base -b ""
```

Our rogue LDAP server has now been configured. When we click the "Test Settings" at http://printer.za.tryhackme.com/settings.aspx, the authentication will occur in clear text. If you configured your rogue LDAP server correctly and it is downgrading the communication, you will receive the following error: "This distinguished name contains invalid syntax". If you receive this error, you can use a tcpdump to capture the credentials.

```
sudo tcpdump -SX -i eth0 tcp port 389
```

```
        0x0000:  4500 0069 b093 4000 8006 20e6 0a0a 0ac9  E..i..@.........
        0x0010:  0a0a 0a39 c2ab 0185 8b05 d64a b818 7e2d  ...9.......J..~-
        0x0020:  5018 2014 3afe 0000 3084 0000 003b 0201  P...:...0....;..
        0x0030:  0560 8400 0000 3202 0102 0418 7a61 2e74  .`....2.....za.t
        0x0040:  7279 6861 636b 6d65 2e63 6f6d 5c73 7663  ryhackme.com\svc
        0x0050:  4c44 4150 8013 7472 7968 6163 6b6d 656c  LDAP..password11
```

NetNTLM authentication used by SMB:

The Server Message Block (**SMB**) protocol allows clients (like workstations) to communicate with a server (like a file share). In networks that use Microsoft AD, SMB governs everything from inter-network file-sharing to remote administration.

use our rogue device to stage a man in the middle attack, relaying the SMB authentication between the client and server, which will provide us with an active authenticated session and access to the target server.

LLMNR, NBT-NS, and WPAD

Responder allows us to perform Man-in-the-Middle attacks by poisoning the responses during NetNTLM authentication, tricking the client into talking to you instead of the actual server they wanted to connect to. On a real LAN, Responder will attempt to poison any Link-Local Multicast Name Resolution (LLMNR), NetBIOS Name Service (NBT-NS), and Web Proxy Auto-Discovery (WPAD) requests that are detected.

Since these protocols rely on requests broadcasted on the local network, our rogue device would also receive these requests. Usually, these requests would simply be dropped since they were not meant for our host. However, Responder will actively listen to the requests and send poisoned responses telling the requesting host that our IP is associated with the requested hostname. By poisoning these requests, Responder attempts to force the client to connect to our AttackBox. In the same line, it starts to host several servers such as SMB, HTTP, SQL, and others to capture these requests and force authentication.

To use **Responder**, write this command,

`sudo responder -I tun0` , make sure you specify `tun0` or `tun1` depending on which tunnel has your network IP.

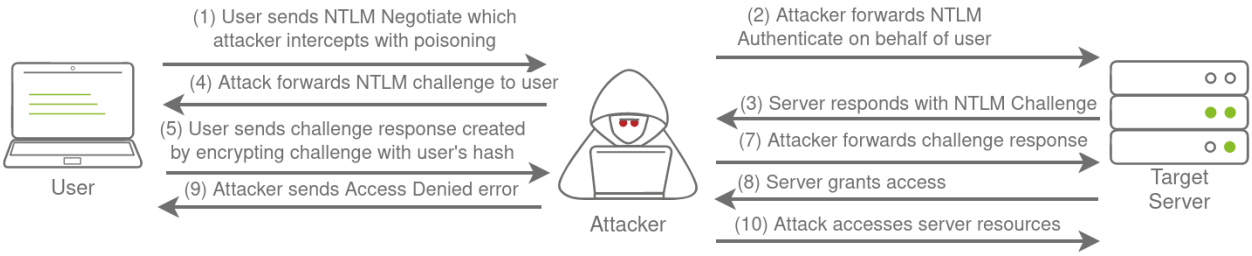
NTLM Password Spraying Attack

```
[+] Listening for events...
[SMBv2] NTLMv2-SSP Client   : <Client IP>
[SMBv2] NTLMv2-SSP Username : ZA\<Service Account Username>
[SMBv2] NTLMv2-SSP Hash     : <Service Account Username>::ZA:<NTLMv2-SSP Hash>
```

Then you will use **hashcat** to crack the password hash sent to u with list of passwords. Use this command

`hashcat -m 5600 <hash file> <password file> --force`

The photo below illustrates how the poisoning happens,



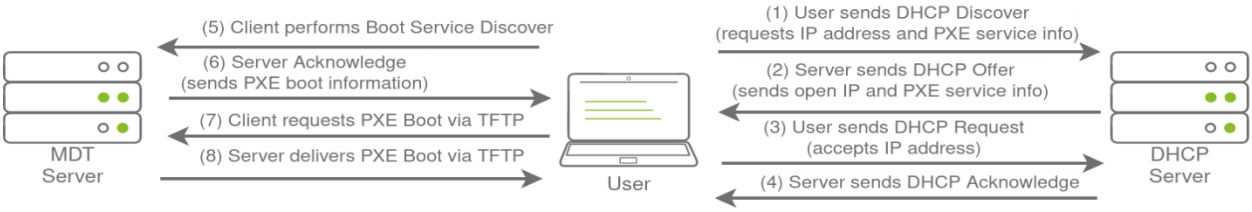
MDT and SCCM:

Microsoft Deployment Toolkit (MDT) is a Microsoft service that assists with automating the deployment of Microsoft Operating Systems (OS). Large organisations use services such as MDT to help deploy new images in their estate more efficiently since the base images can be maintained and updated in a central location.

MDT is integrated with Microsoft's System Center Configuration Manager (SCCM), which manages all updates for all Microsoft applications, services, and operating systems. MDT is used for new deployments. Essentially it allows the IT team to preconfigure and manage boot images.

PXE Boot

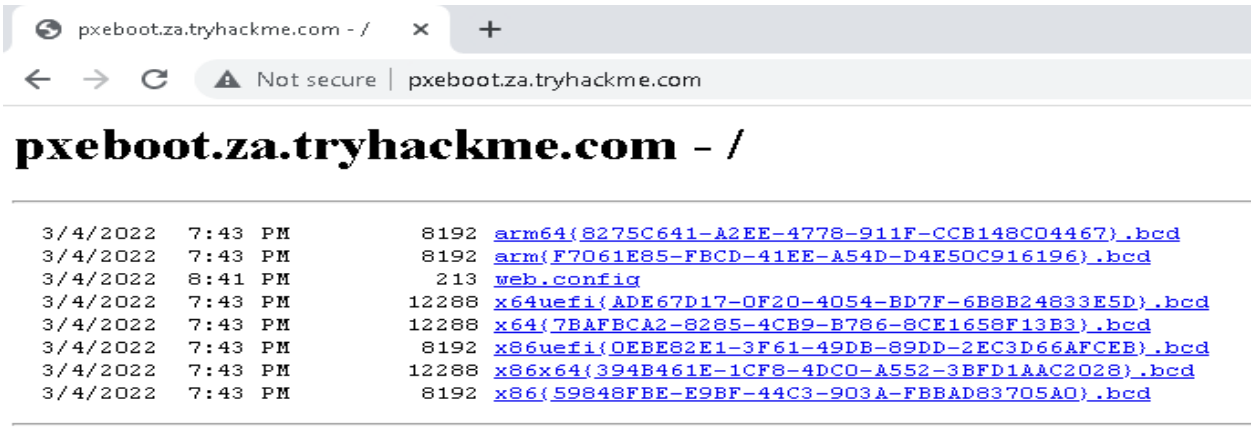
Large organisations use PXE boot to allow new devices that are connected to the network to load and install the OS directly over a network connection. MDT can be used to create, manage, and host PXE boot images. PXE boot is usually integrated with DHCP, which means that if DHCP assigns an IP lease, the host is allowed to request the PXE boot image and start the network OS installation process.



Since DHCP is a bit finicky, we will bypass the initial steps of this attack. We will skip the part where we attempt to request an IP and the PXE boot preconfigure details from DHCP. We will perform the rest of the attack from this step in the process manually.

The second piece of information you would have received was the names of the BCD files. These files store the information relevant to PXE Boots for the different types of architecture.

You will find it at http://pxeboot.za.tryhackme.com/ , It will list various BCD files.



you would use TFTP to request each of these BCD files.

we will focus on the BCD file of the **x64** architecture. Copy and store the full name of this file. For the rest of this exercise, we will be using this name placeholder `x64{7B...B3}.bcd` since the files and their names are regenerated by MDT every day.

We will use SSH to connect to machine, use this command with there credentials.

```
ssh thm@THMJMP1.za.tryhackme.com  :Password1@
```

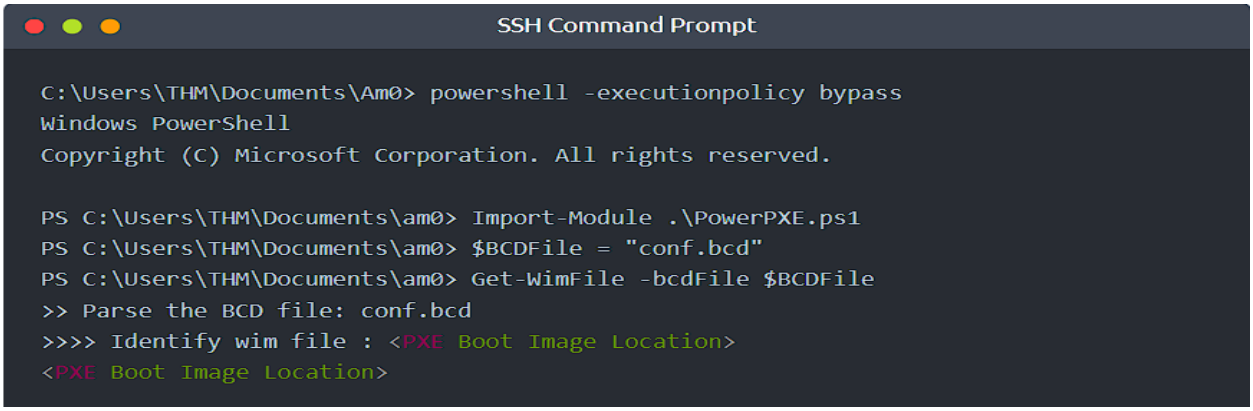
The first step we need to perform is using TFTP and downloading our BCD file to read the configuration of the MDT server. TFTP is a bit trickier than FTP since we can't list files. Instead, we send a file request, and the server will connect back to us via UDP to transfer the file.

to download bcd files use this command.

```
tftp -i <THMMDT IP> GET "\Tmp\x64{39...28}.bcd" conf.bcd
```

then we will use **Powerpxe** tool to read its contents.

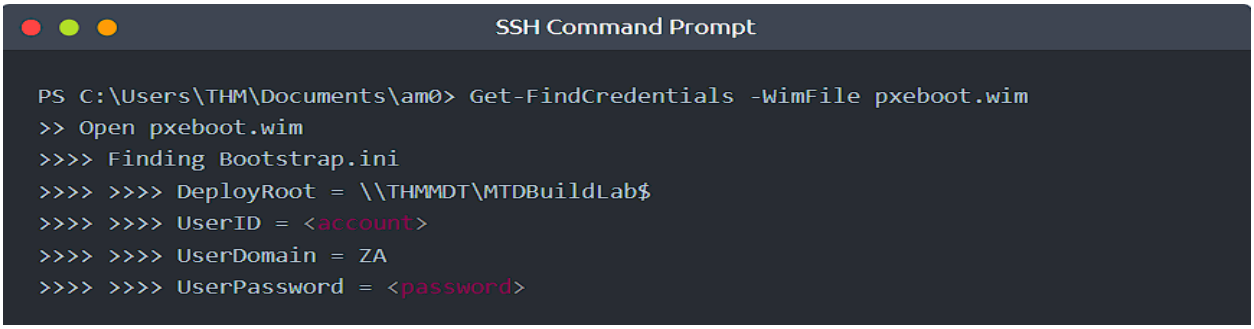
Powerpxe is a PowerShell script that automatically performs this type of attack but usually with varying results, so it is better to perform a manual approach. We will use the Get-WimFile function of **Powerpxe** to recover the locations of the **PXE Boot images** from the BCD file.



Then write this command with image location to download it.

```
tftp -i <THMMDT IP> GET "<PXE Boot Image Location>" pxeboot.wim
```

after downloading pxeboot , use this command to retrieve credentials.



It should be noted that there are various attacks that we could stage. We could inject a local administrator user, so we have admin access as soon as the image boots, we could install the image to have a domain-joined machine.

Configuration Files

configuration files are an excellent avenue to explore in an attempt to recover AD credentials. Depending on the host that was breached, various configuration files may be of value for enumeration:

- Web application config files
- Service configuration files
- Registry keys
- Centrally deployed applications

An example of such as application is McAfee Enterprise Endpoint Security, which organisations can use as the endpoint detection and response tool for security.

McAfee embeds the credentials used during installation to connect back to the orchestrator in a file called ma.db. This database file can be retrieved and read with local access to the host to recover the associated AD service account.

We will be using the SSH access on THMJMP1 again for this exercise. The ma.db file is stored in a fixed location

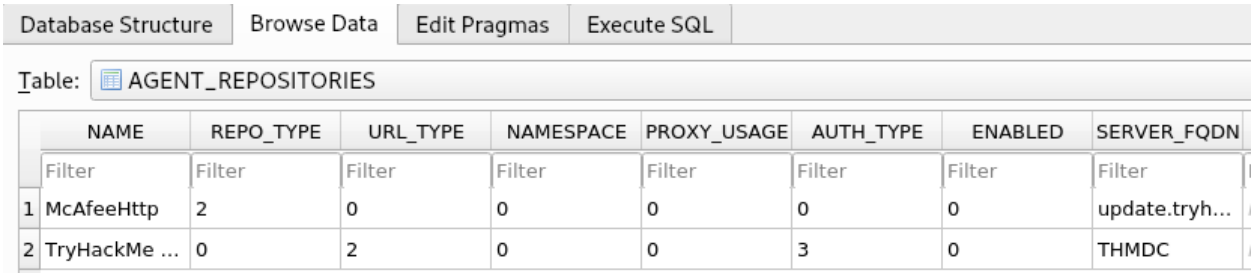
We can use SCP to copy the ma.db to our AttackBox.

```
scp thm@THMJMP1.za.tryhackme.com:C:/ProgramData/McAfee/Agent/DB/ma.db
```

we will use a tool called sqlitebrowser to read .db files.

```
sqlitebrowser ma.db
```

we will select the Browse Data option and focus on the AGENT_REPOSITORIES table.



	NAME	REPO_TYPE	URL_TYPE	NAMESPACE	PROXY_USAGE	AUTH_TYPE	ENABLED	SERVER_FQDN
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	McAfeeHttp	2	0	0	0	0	0	update.tryh...
2	TryHackMe ...	0	2	0	0	3	0	THMDC

We are particularly interested in the second entry focusing on the DOMAIN, AUTH_USER, and AUTH_PASSWD field entries. Make a note of the values stored in these entries. However, the AUTH_PASSWD field is encrypted. Luckily, McAfee encrypts this field with a known key.

If you gonna use python2 script you will use it like command below,

```
python2 mcafee_sitelist_pwd_decrypt.py <AUTH PASSWD VALUE>
```