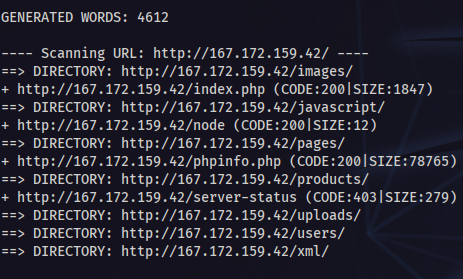
**Walkthrough of ShadowStore**

## **Vulnerabilities**:

1. LFI
2. PHP object Injection
3. SQLi
4. File Upload
5. XXE
6. RCE
7. RSE
8. Sensitive Data Exposure

Start with directory scanning and port scanning.



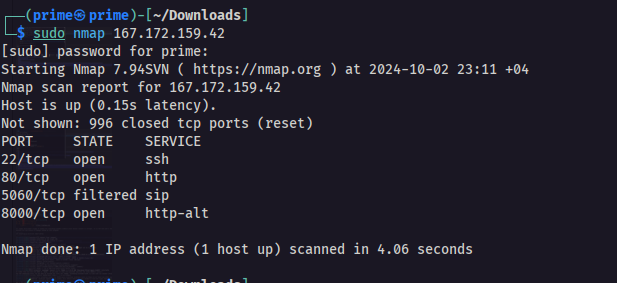
In here we have 3 important directories that each one have vulnerabilities or important for enumeration vulnerabilities. These are:

1. /phpinfo.php (Sensitive Data Exposure)
2. /xml (XXE)
3. /uploads (for File Upload and RSE)
4. /node (RCE and File Upload)

There are 2 services running separately but works together.

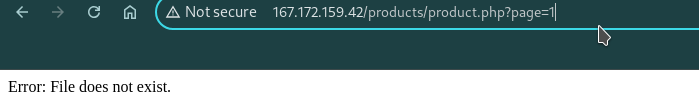
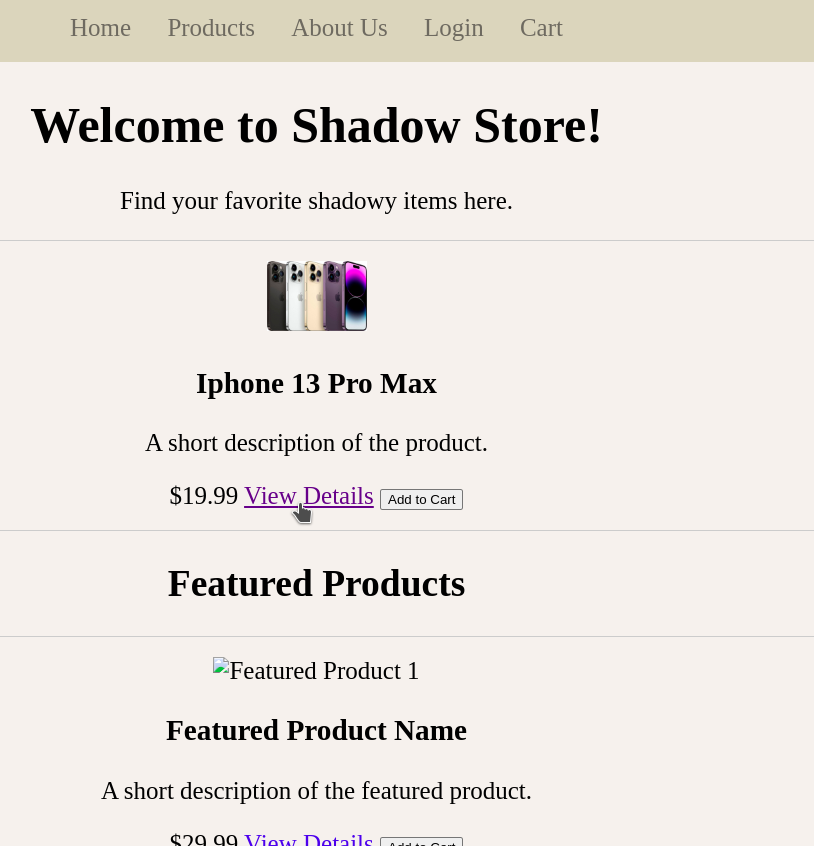
We can see the second service in port scan.

And port scan.

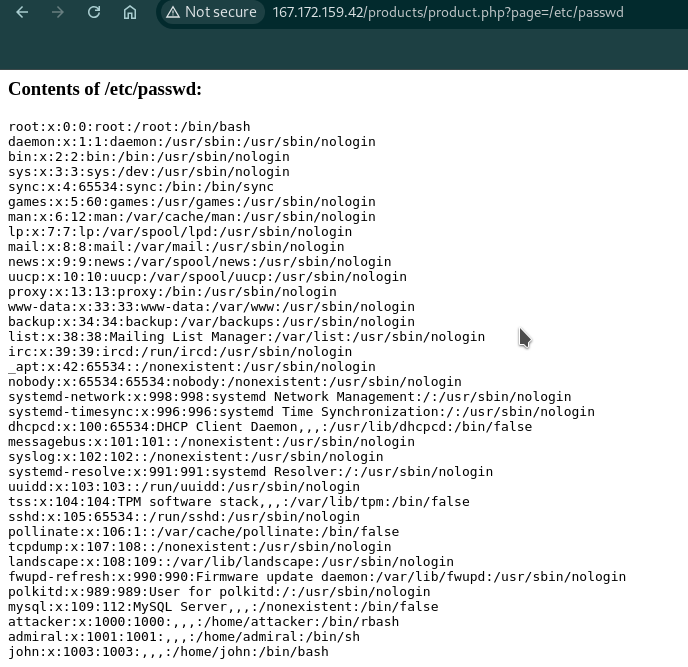


Now take a look at vulnerabilities.

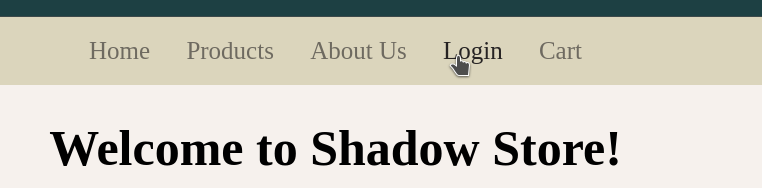
### **1. LFI**



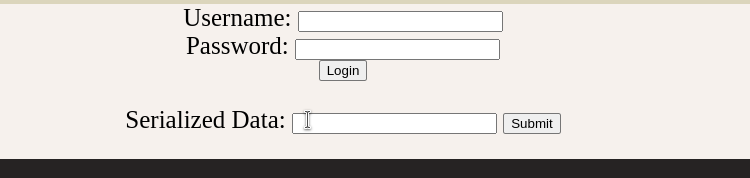
Change “1” to “/etc/passwd”



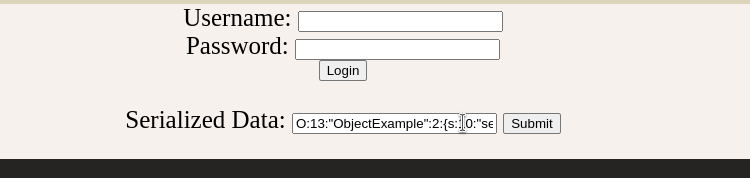
### **2. SQLi and 3. PHP Object Injection (in login.php)**

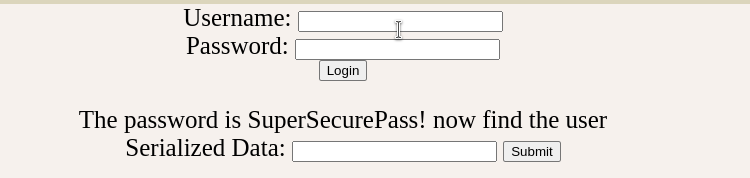


First, we need to use PHP object injection to find user password.

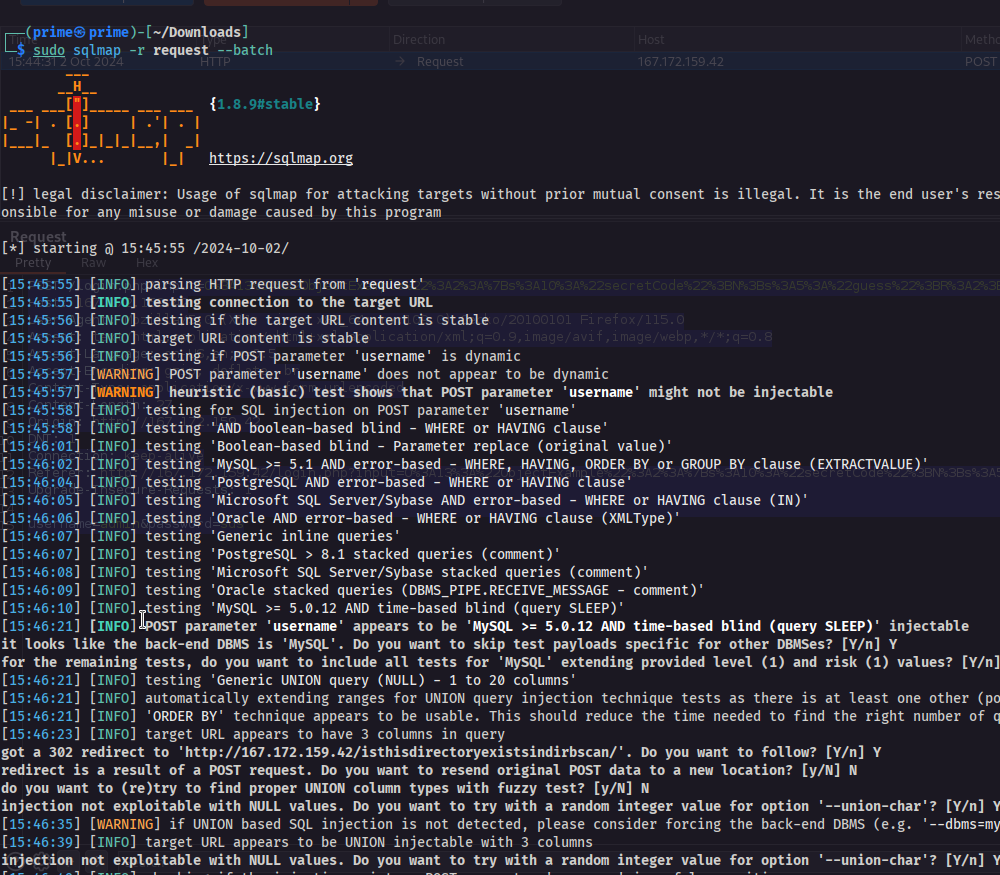
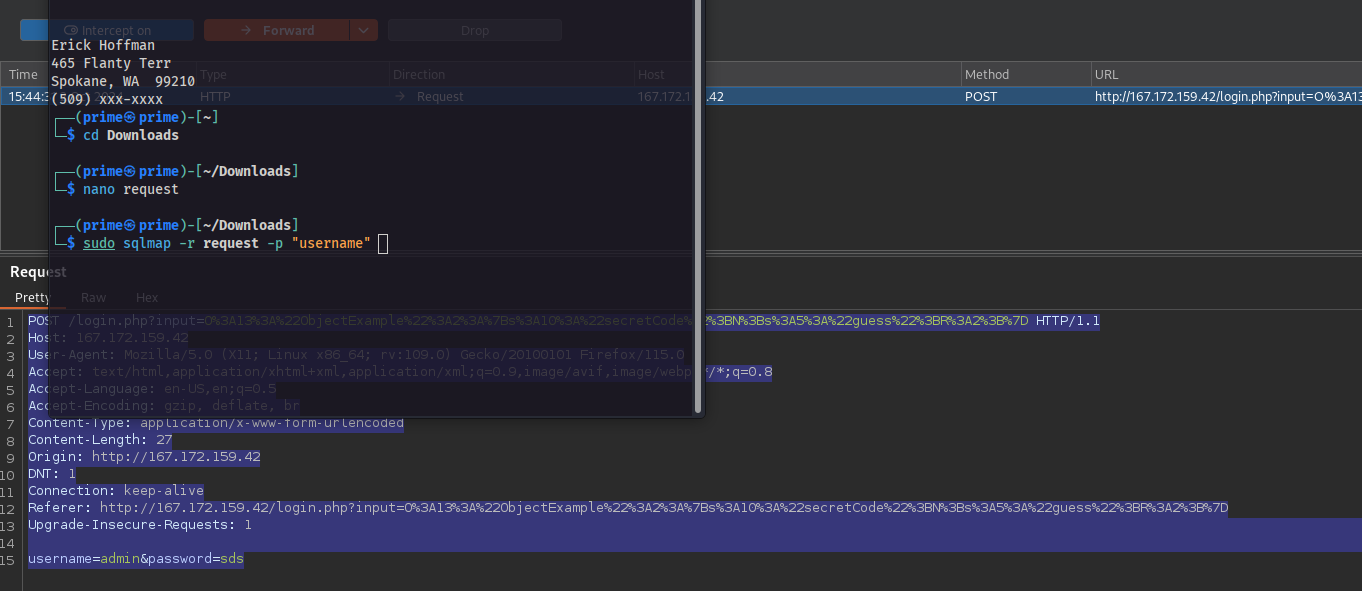
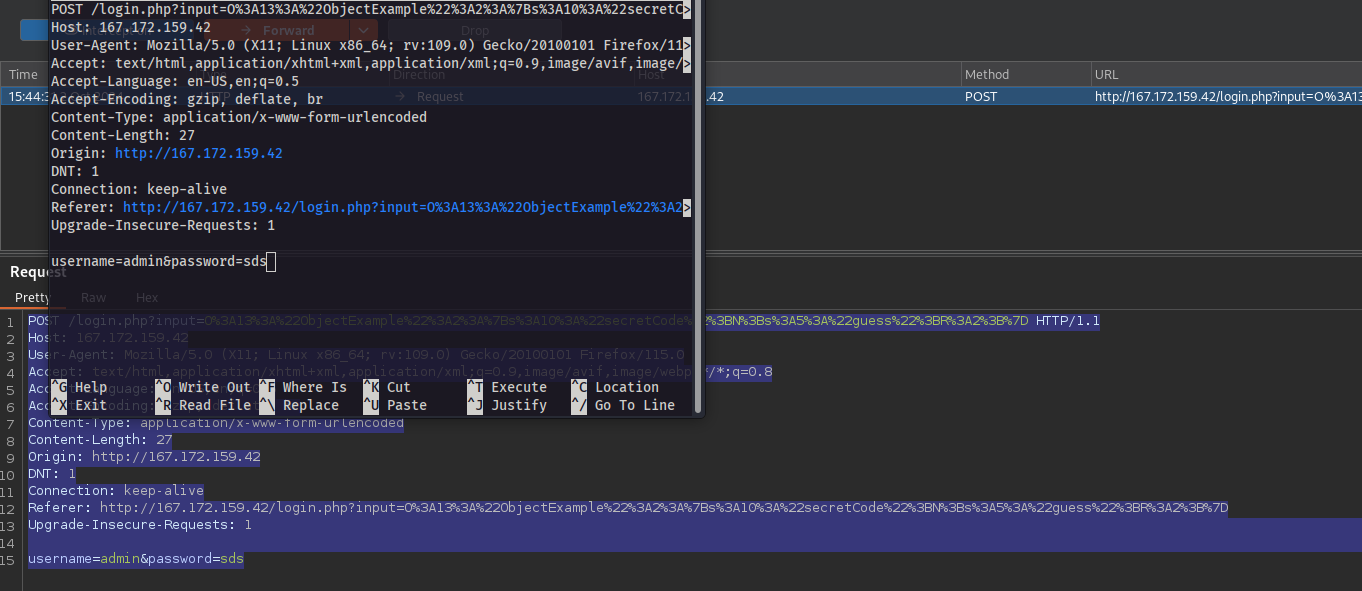
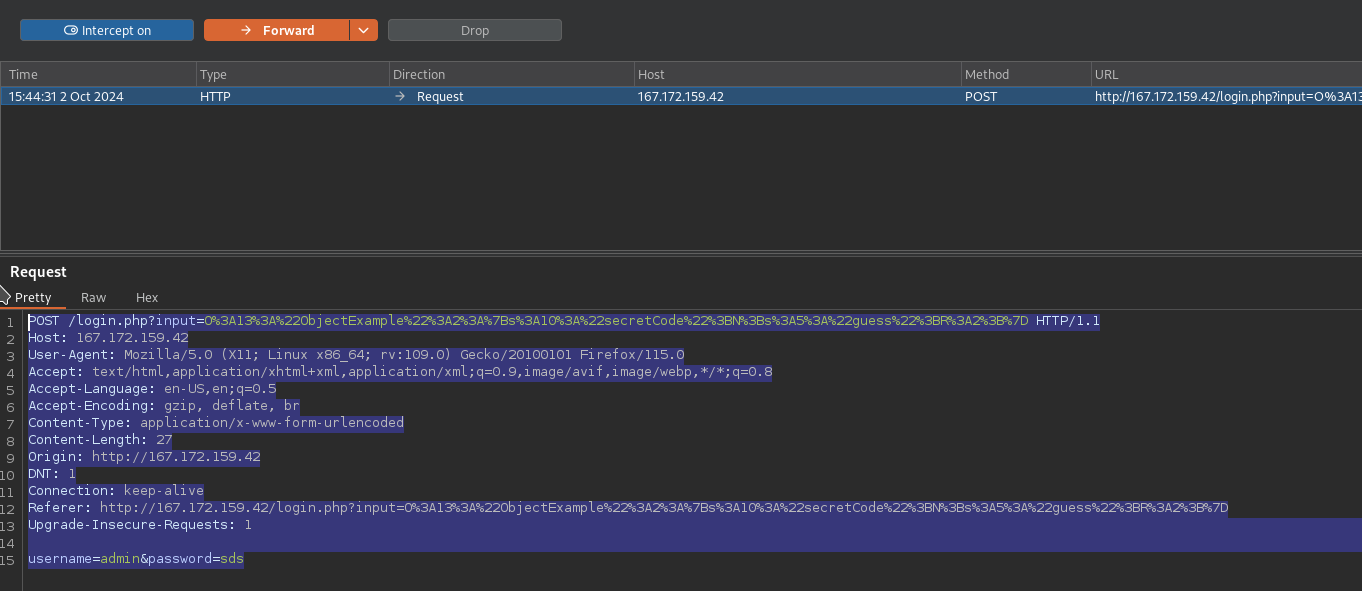


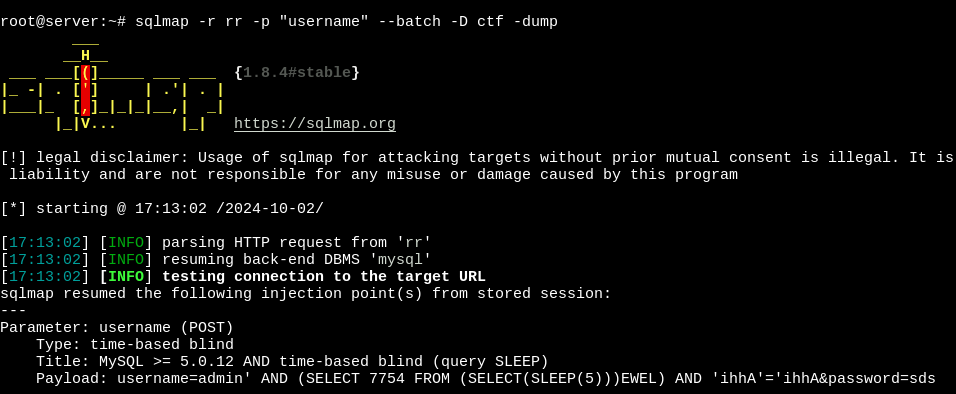
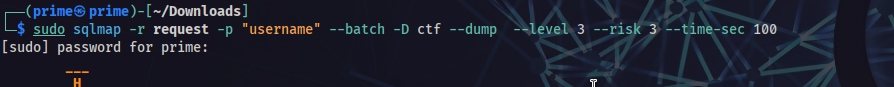
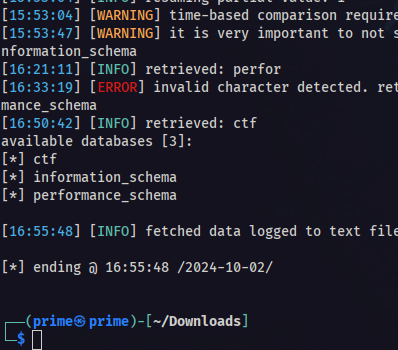
Payload : O:13:"ObjectExample":2:{s:10:"secretCode";N;s:5:"guess";R:2;}



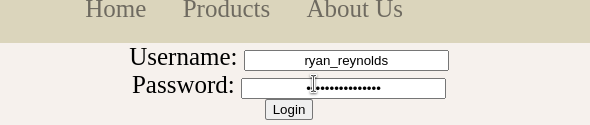
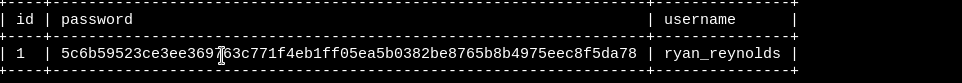


Now we need to find user that stored in SQL database. Because of the POST method, we must use BurpSuite. Copy the request, save in a file (in this walkthrough, “request” or “rr” file) and using this file make a sqlmap scan



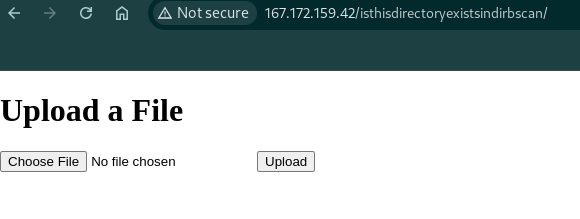


After enumeration, username is “ryan\_reynolds”

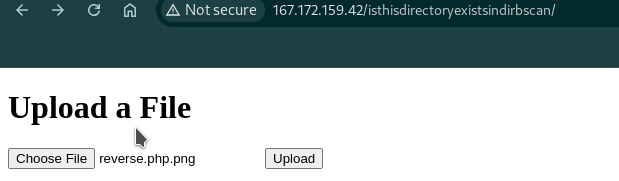
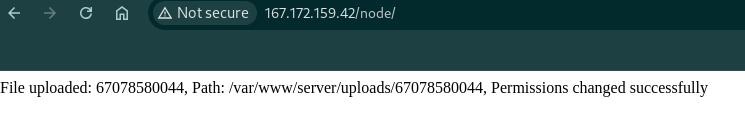


If credentials are correct, the site redirects us to a directory that we have no chance to find with scan. In there we will face File upload vulnerability and RCE.

### **4. File Upload**

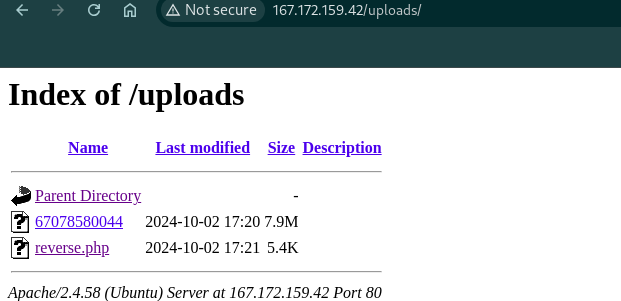


File name is <name>.ext dd



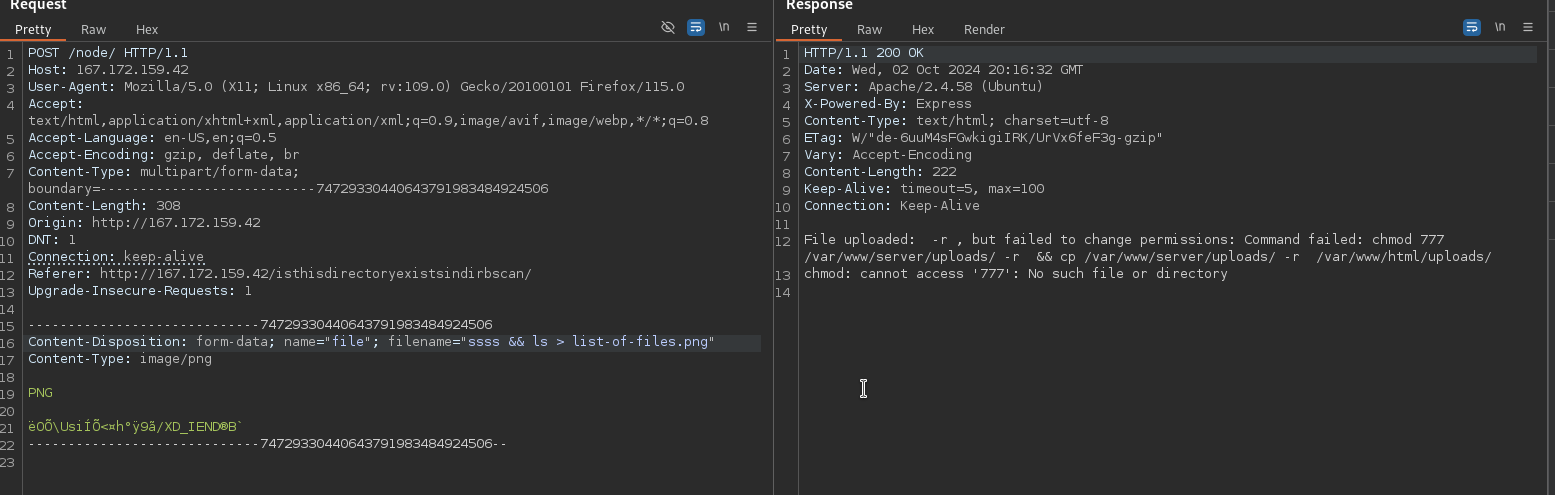
Here, the server only checks the extension one time and removes the extension of the file for storing and copying. We can’t see any copy message in the browser but on the server side, it copies and changes file permissions for us. First, file permissions are changing and second file server copies file to /uploads folder. You don’t know first but second can easily be found for you. There is a reverse shell code for php in reverse.php file (we uploaded it as reverse.php.png and server removed the extension). I will say why I named it like this.

Note: Because of DigitalOcean, I prefer to raise a server for chasing reverse shell.



### **5. RCE**

The file upload page which you can find after login, also has RCE vulnerability (Remote Code Execution). I wrote that it copies file to another directory. For this operation, the server uses terminal commands and doesn't sanitize filename. So that means we can run system commands with filename. For default user apache2 is *www-data* user. In this system, this user cannot run network commands but ve can run system commands. That means we can see what is on the server. Let's check this.

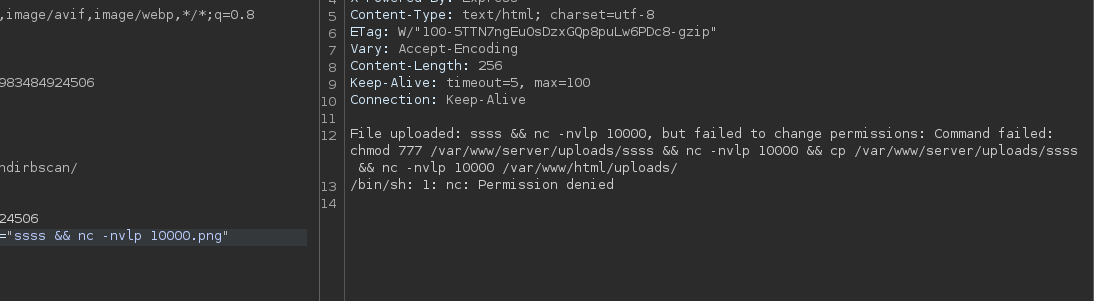


Here, using the Burp Suite, I chased a file upload request and changed the file name to malicious code. Using the LFI, I will get the content of the file “list-of-files”.



and boom! We can run system commands that the server allows us.

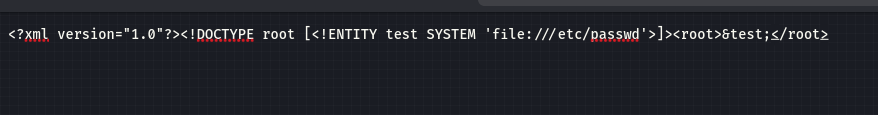
If I want to run system commands like “nc”, I get this error



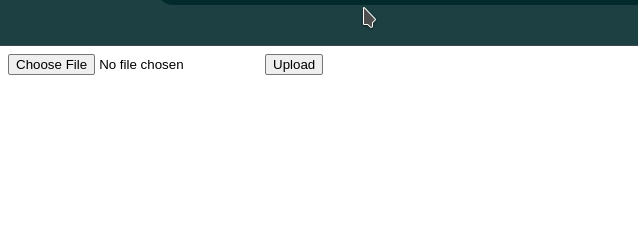
System will not allow me to listen to any port or send files via ports.

### **6. XXE**

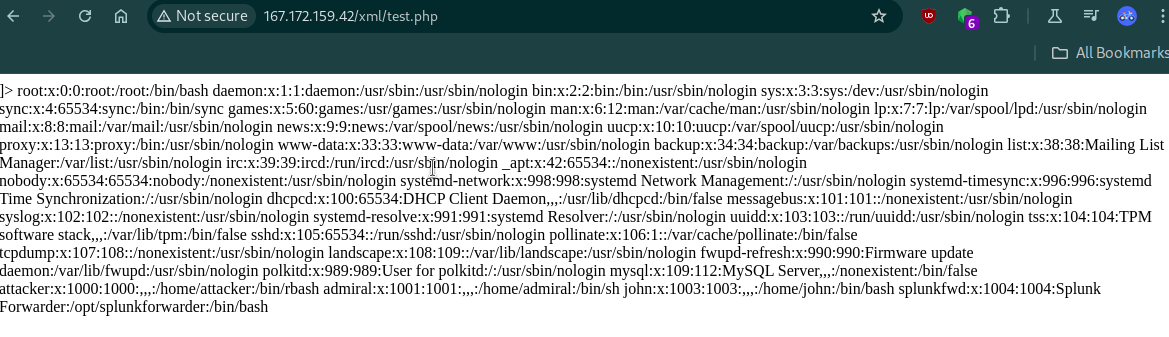
One of the vulnerabilities is XXE (XML External Entity). Create file with .xml extension and add this payload in it



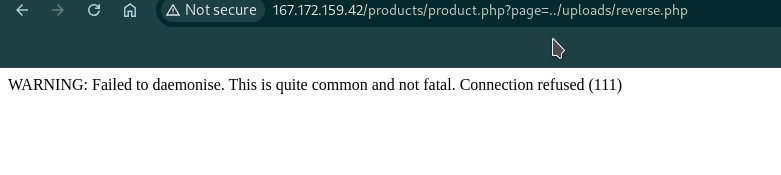
Then upload file in /xml path. When you click the upload button…



Server will parse xml code and shows us the result.

**7. RSE (Reverse Shell Execution)**

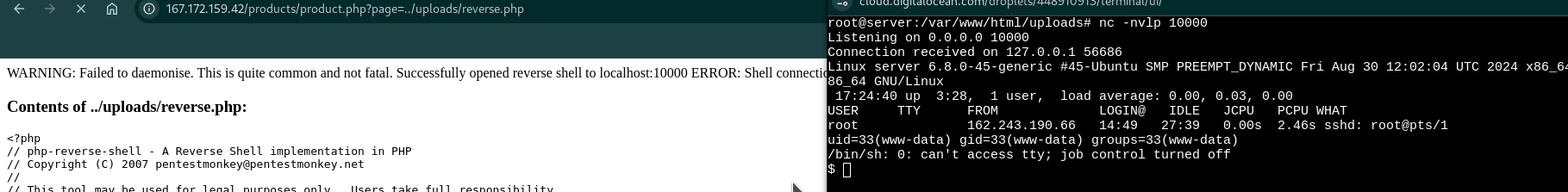
You remember that we uploaded reverse.php.png to the server. We can run this code with LFI. Why? Because product.php takes the file location from the page query and if the file is a php file, it runs and imports the file to the current page. This functionality helps us to run the reverse shell which we uploaded to the server. /uploads directory is one directory below the /products/product.php. That means, our path it way to look like this : “../uploads/reverse.php”



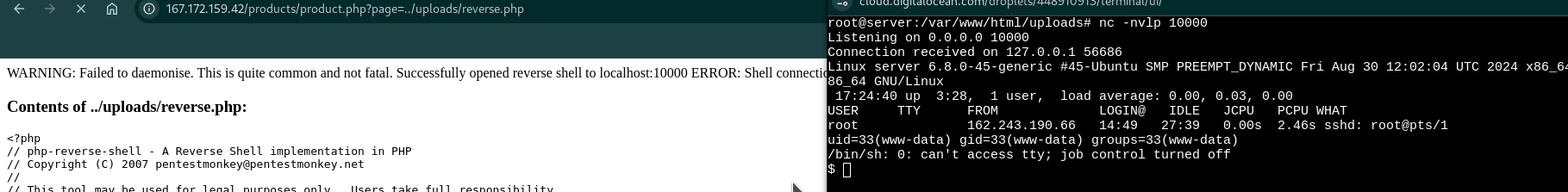
Before entering the url, we need to create a listener for the reverse shell. With help of nc command : nc -nvlp <PORT you described in reverse.php>

And enter the link

Browser

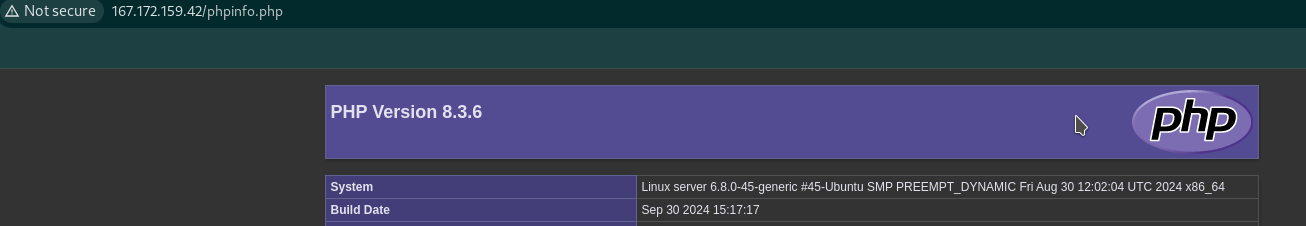


Our listener



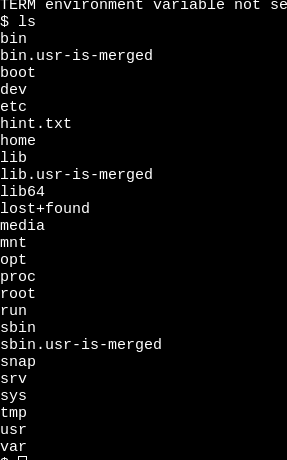
We are www-data user. We will continue after showing last web vulnerability

**8. Sensitive Data Exposure**

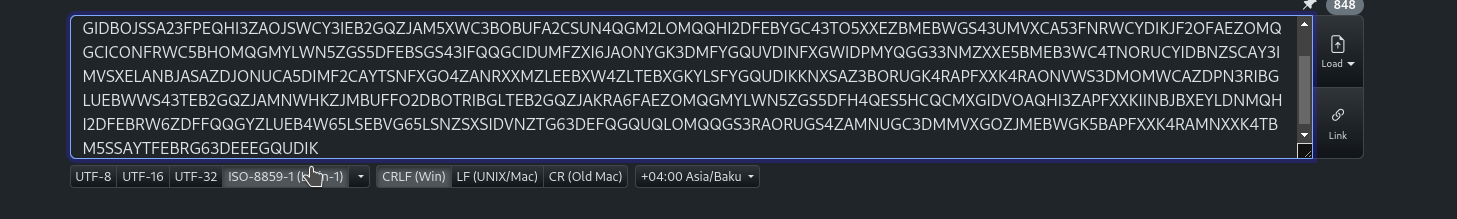
****

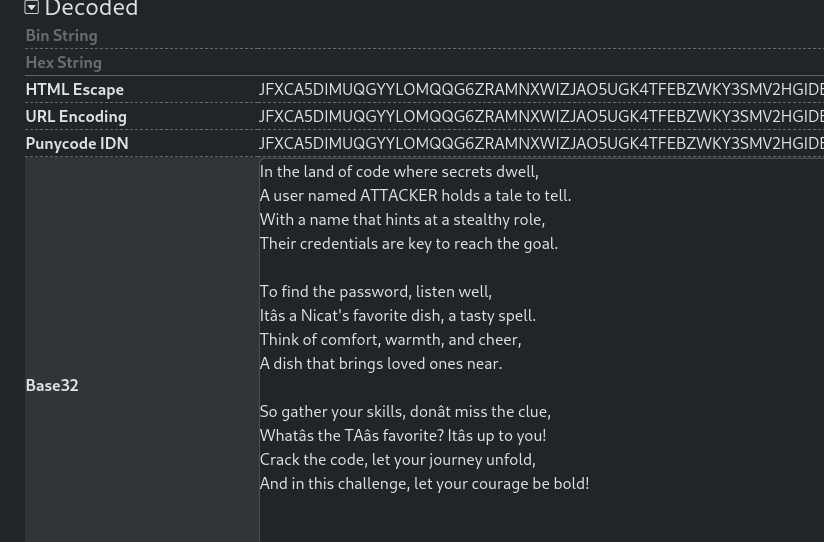
You see that phpinfo.php is open and accessible for everyone

Now, back to our reverse shell. Do you see anything different?



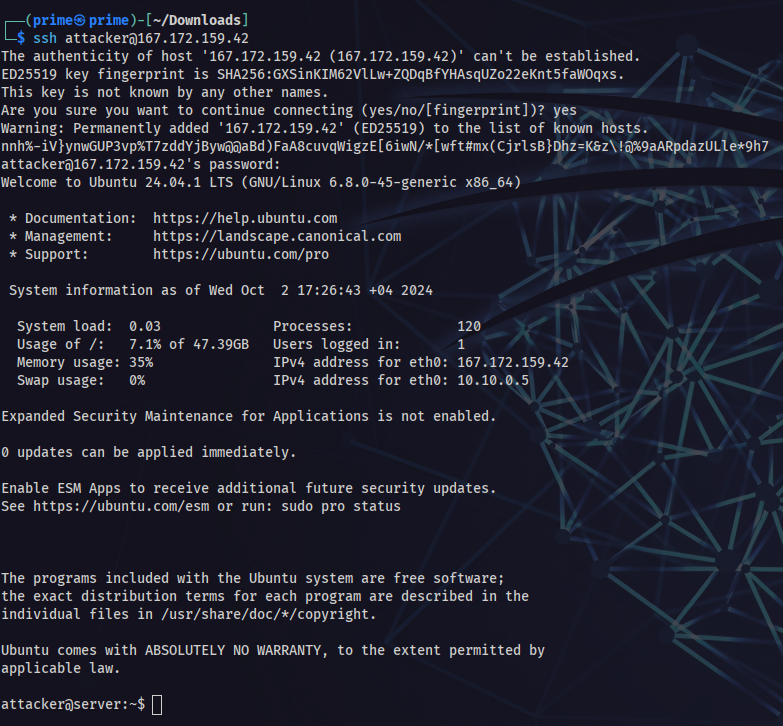
If you don’t find, let me show you the hint.txt file. In this file, you can see encoded string. For finding decoded text, use [this link](https://dencode.com/#:~:text=Encoding%20and%20Decoding%20site.%20e.g.%20HTML%20Escape%20/%20URL%20Encoding) and don’t change preferences..





A poem and hint is in it. Summary of poem is : username “attacker” password:”toyugdoneri”

make a connection with ssh and get /bin/rbash shell.

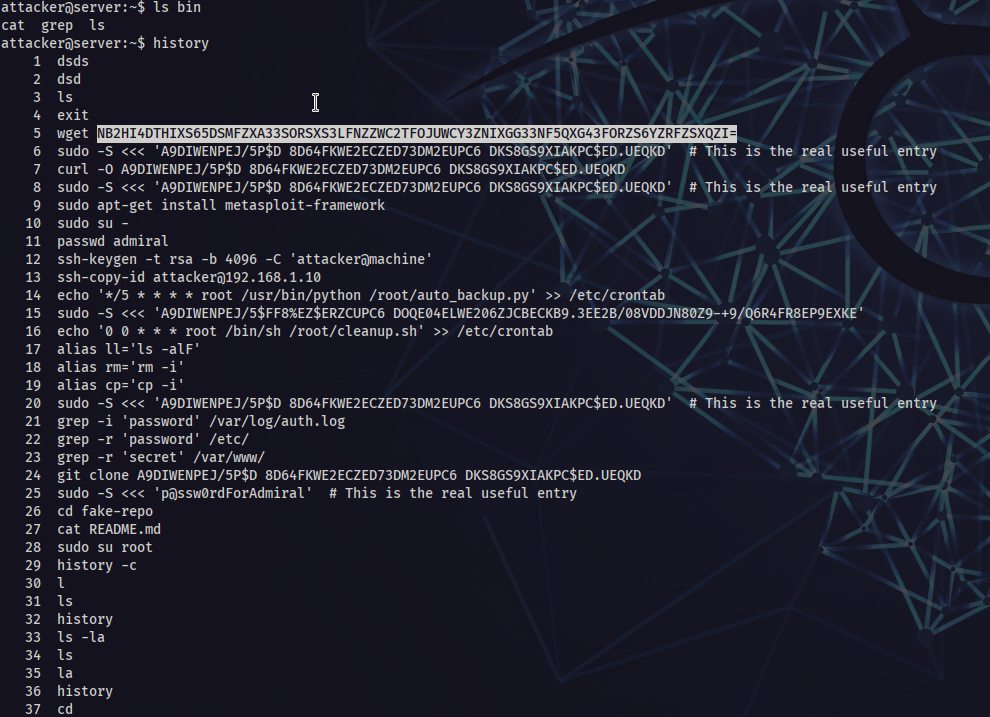


Only cat, grep and ls commands are allowed

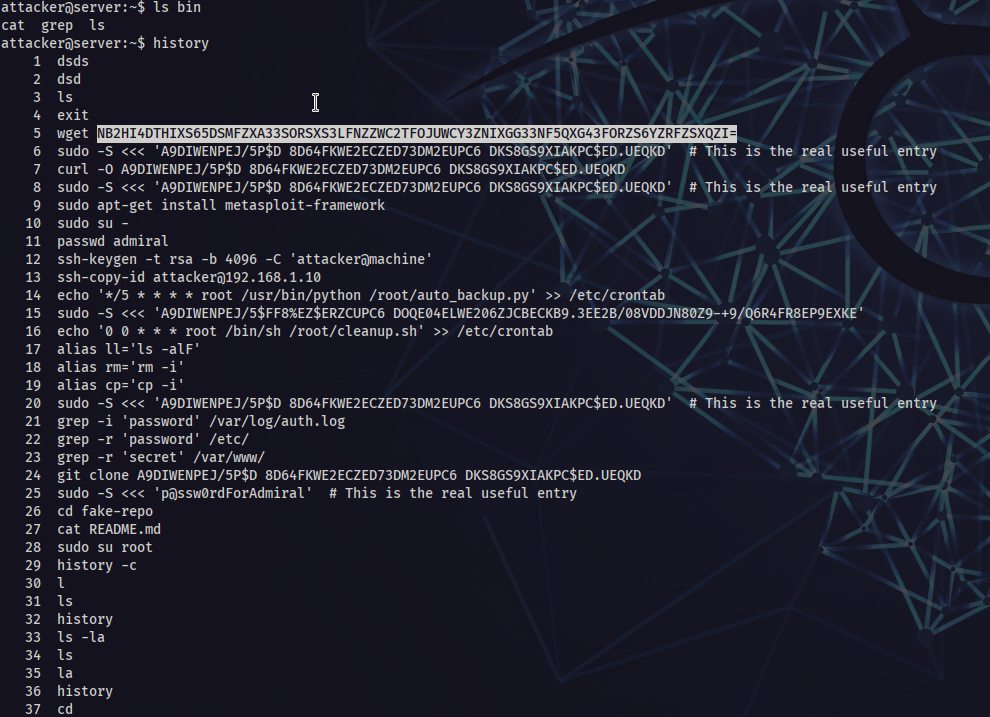


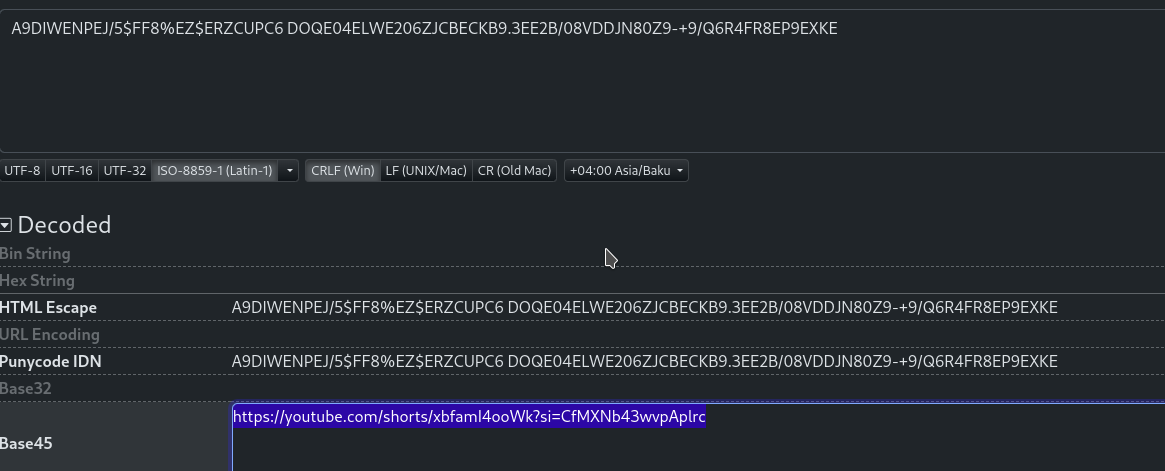
But what does the user do before restrictions?

Take look history of user.



not all of them give hints. only the 15th one is a real hint.





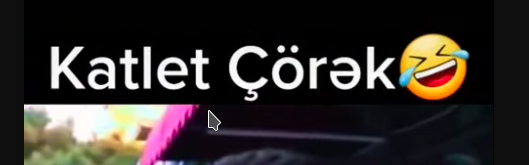
and this is the next user password. The password of admiral user. Guardian of the root door.

The One Guardian of SS. Root ship.

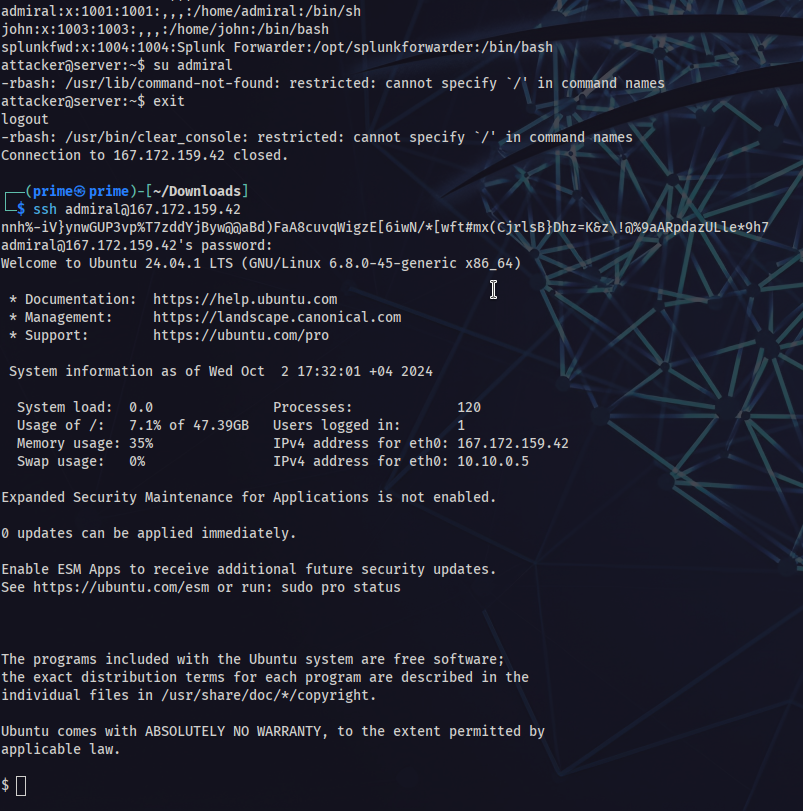
AND….

IT’S…..

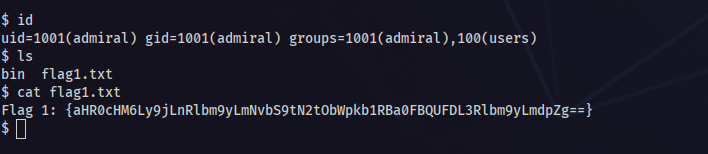
PASSWORD iss…..



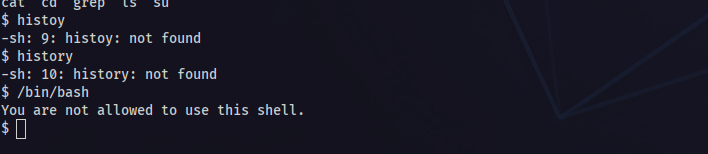
Really? Nevermind. It is a joke )



And the First Flag.

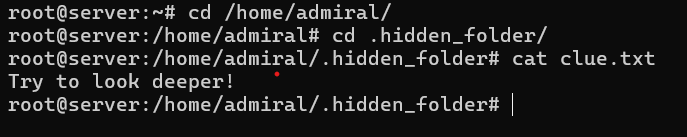


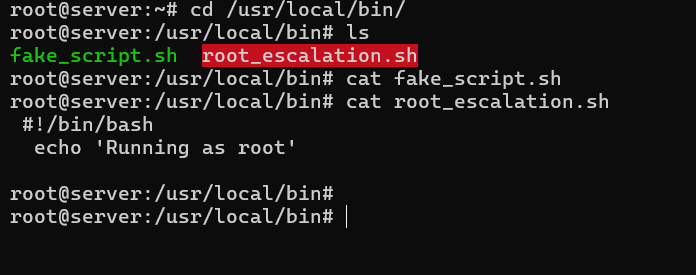
If you decrypt base64. You will get a link and you will see [this gif](https://c.tenor.com/m7kNmjdoTAkAAAAC/tenor.gif)



and some restrictions.

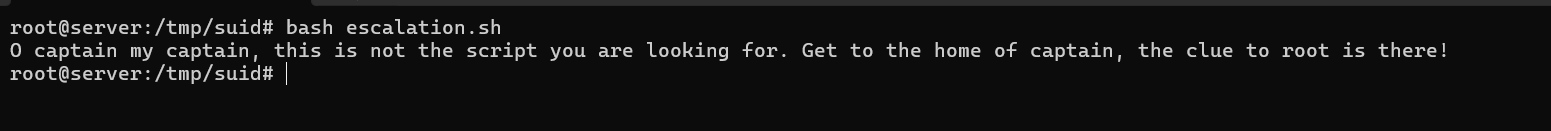
There are also small honeypots for attackers:



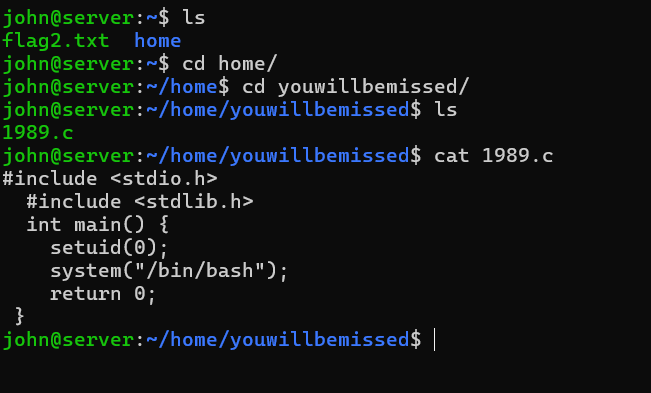


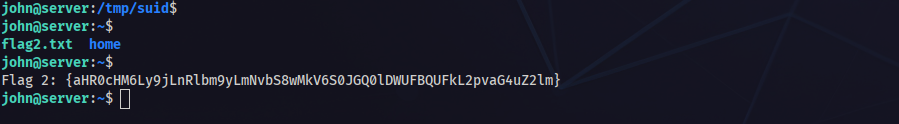
The attacker will have to run the script below:  


The admiral is actually has been provided permission to run this script without password and get to captain group. The reason for that is, when you get to captain group you can get to john user home folder, and there you can see the SUID script for getting to root. There will be two way to get there, either attacker will run the SUID and get to root, or make a small OSINT and get to john user and get to root.



The case is that “O captain, my captain” is famous sentence from a movie Dead Poets Society, and John is its primary character. So this sentence is actually a hind. And the password of john user is “deadpoetssociety”.





In either way you can get to root user and get the flag.   


FIN! :)