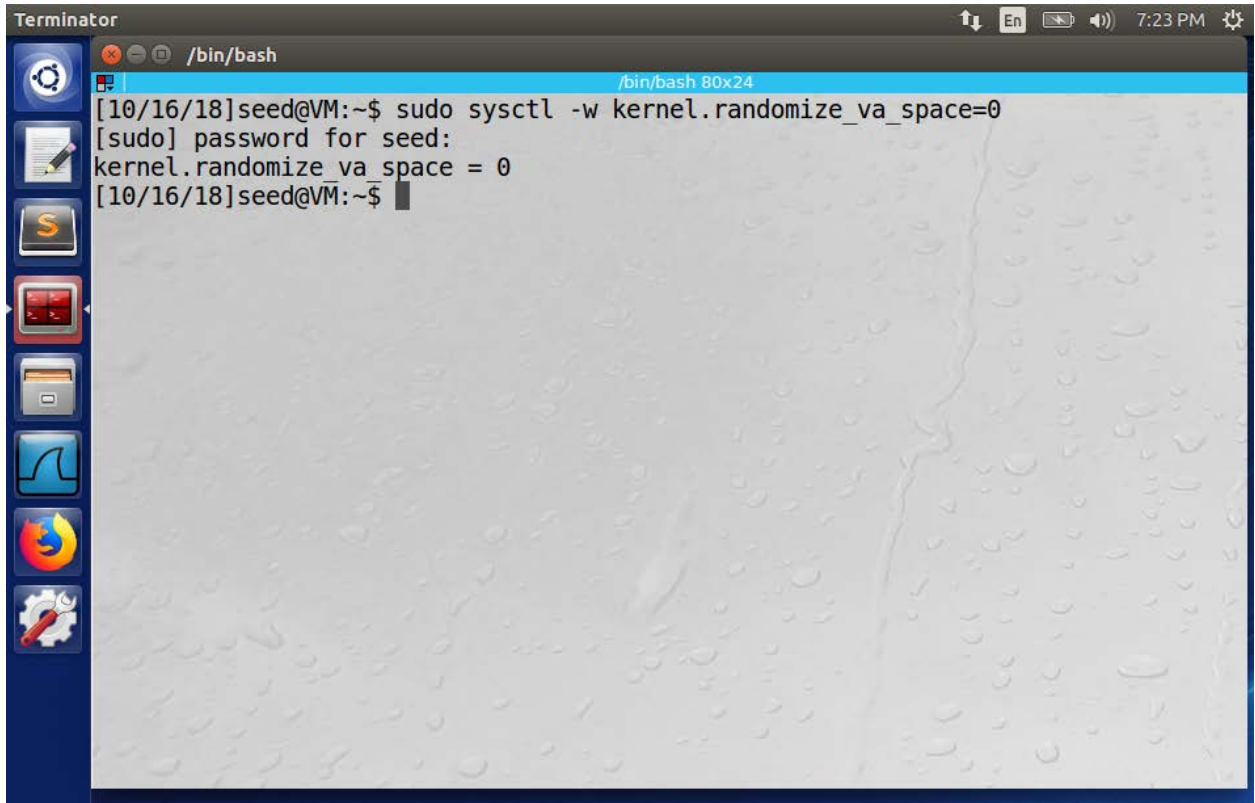


CSC 6222: Assignment 3

Q1. The password for the system is “dees”.



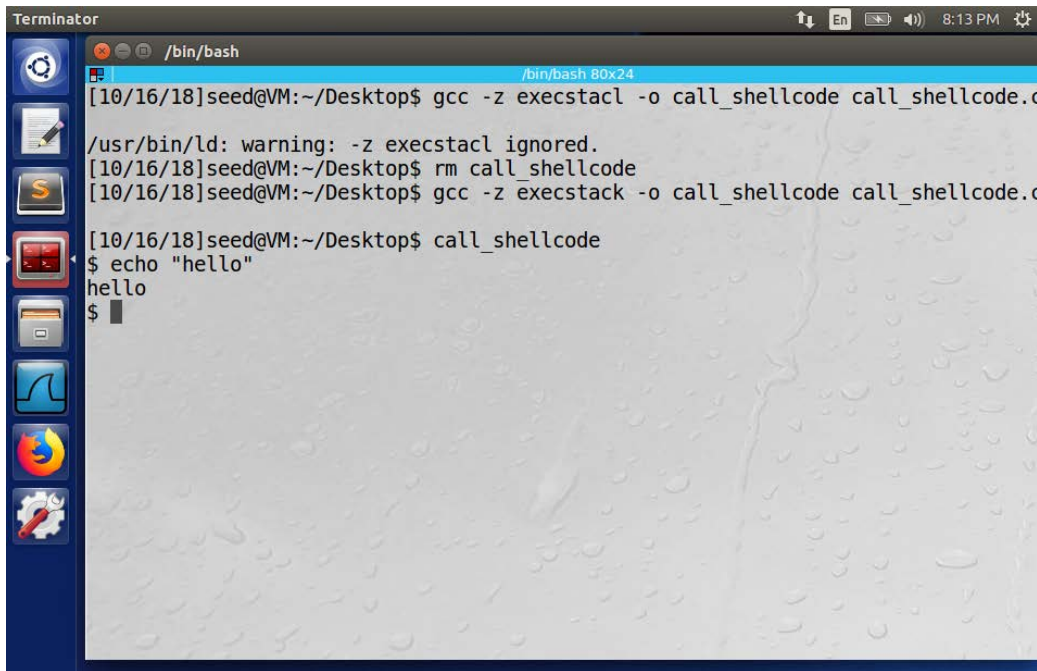
The image shows a screenshot of a Terminator terminal window. The window title is "Terminator". The terminal displays the following text:

```
/bin/bash  
[10/16/18]seed@VM:~$ sudo sysctl -w kernel.randomize_va_space=0  
[sudo] password for seed:  
kernel.randomize_va_space = 0  
[10/16/18]seed@VM:~$
```

The terminal window has a blue title bar and a dark blue sidebar on the left containing several icons. The main area of the terminal is light gray with a subtle texture. The status bar at the top right shows the time as 7:23 PM and some system icons.

Q2.

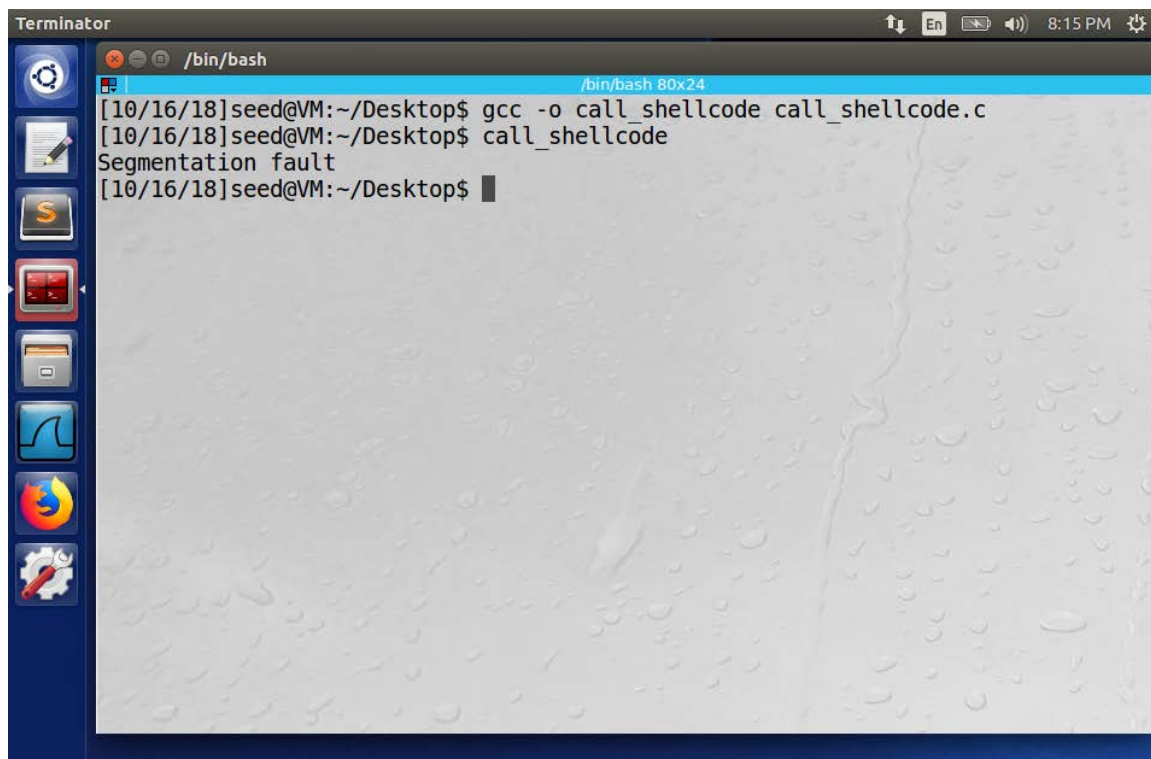
w/execstack



The screenshot shows a Terminator terminal window with a blue title bar and a dark blue sidebar containing icons for various applications. The terminal window has a title bar with the text "/bin/bash" and a status bar at the bottom showing "8:13 PM". The terminal output is as follows:

```
/bin/bash
[10/16/18]seed@VM:~/Desktop$ gcc -z execstack -o call_shellcode call_shellcode.c
/usr/bin/ld: warning: -z execstack ignored.
[10/16/18]seed@VM:~/Desktop$ rm call_shellcode
[10/16/18]seed@VM:~/Desktop$ gcc -z execstack -o call_shellcode call_shellcode.c
[10/16/18]seed@VM:~/Desktop$ call_shellcode
$ echo "hello"
hello
$
```

w/out execstack



The screenshot shows a Terminator terminal window with a blue title bar and a dark blue sidebar containing icons for various applications. The terminal window has a title bar with the text "/bin/bash" and a status bar at the bottom showing "8:15 PM". The terminal output is as follows:

```
/bin/bash
[10/16/18]seed@VM:~/Desktop$ gcc -o call_shellcode call_shellcode.c
[10/16/18]seed@VM:~/Desktop$ call_shellcode
Segmentation fault
[10/16/18]seed@VM:~/Desktop$
```

Q3.

Strcpy(a,b) is a C function that copies the string b to the string a. However, the function does not check to see whether the number of bytes in b are equivalent to the number of bytes in a. If the string b has more bytes, then memory allocated to a will “overflow” into memory already allocated for other functions.

Q4.

The badfile should be filled with a sled of NOOP instructions with the shellcode at the end of the file.

Q5.

Yes.

Q6.

- i) A buffer overflow is when data written to a buffer (a block of memory) is larger than the buffer size.
- ii) A buffer overflow attack will cause the corruption of some memory, thus resulting in different values for some variables that can be exploited.
- iii) Using a strong-typed language like Java, not using unsafe functions like strcpy(), and barring execution of the stack like gcc's -z noexecstack flag can help prevent buffer overflows.

Q7.

People will start finding more bugs or exploits in the browser as more people switch to it. If there was a significant exploit in the browser, then Internet security as a whole would diminish since a lot of people would be using it.

Q8.

The malware could harvest data from the user in a variety of ways and get access to the user's physical home address (say credit card details in a transaction).

Q9. 554K bytes

Q10. The distance from the two host are significant enough to cause a delay higher than 10msec.