**CSCE 548 Spring 2020**

**Project 2**

**Itay Goldfaden**

**03/26/2020**

**Task 1 – 2.4**

Before I could have started, I had to follow a few initial setups.

1. At first, I had to set the address space randomization to be equal to 0 in order to make the next steps easier.

* su root
* sysctl -w kernel.randomize\_va\_space=0

1. Then, I had created a shellcode(a code to launch a shell) and load it into the memory.

* gcc -z execstack -o call\_shellcode call\_shellcode.c

1. the final step was to compile the stack.c program and make it root.

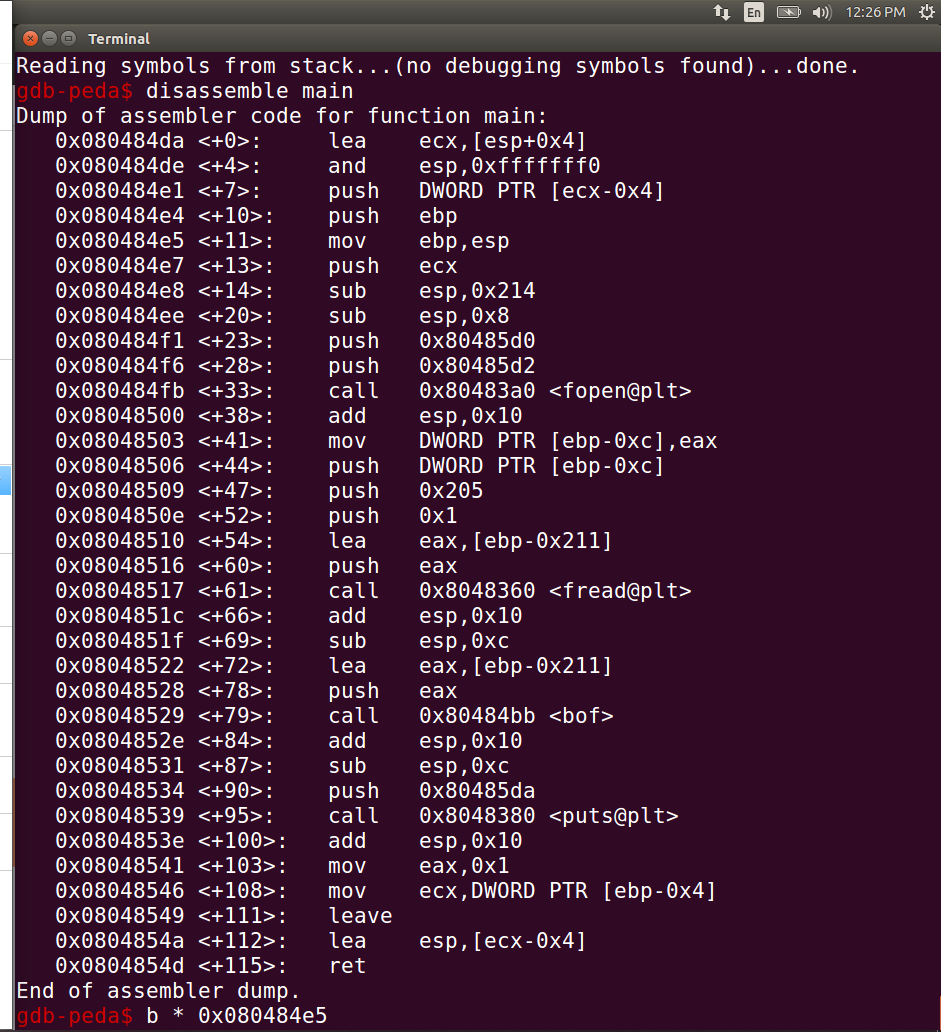
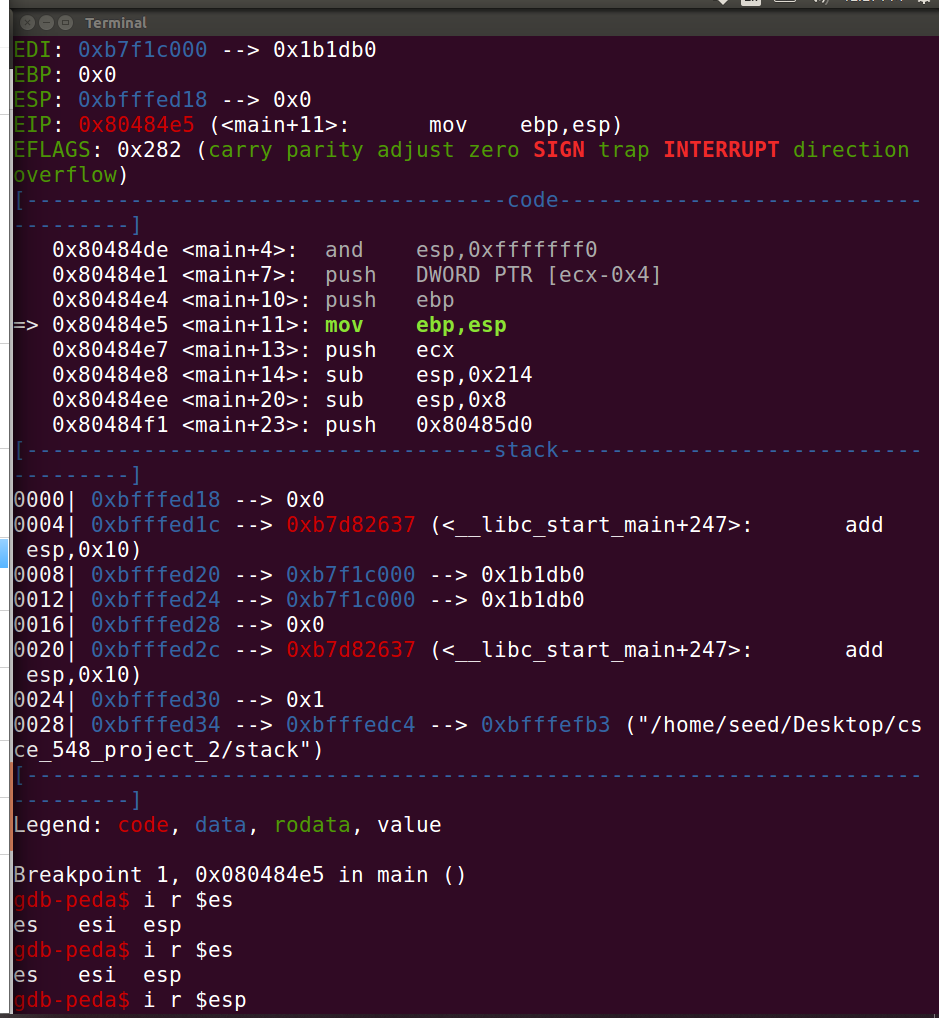
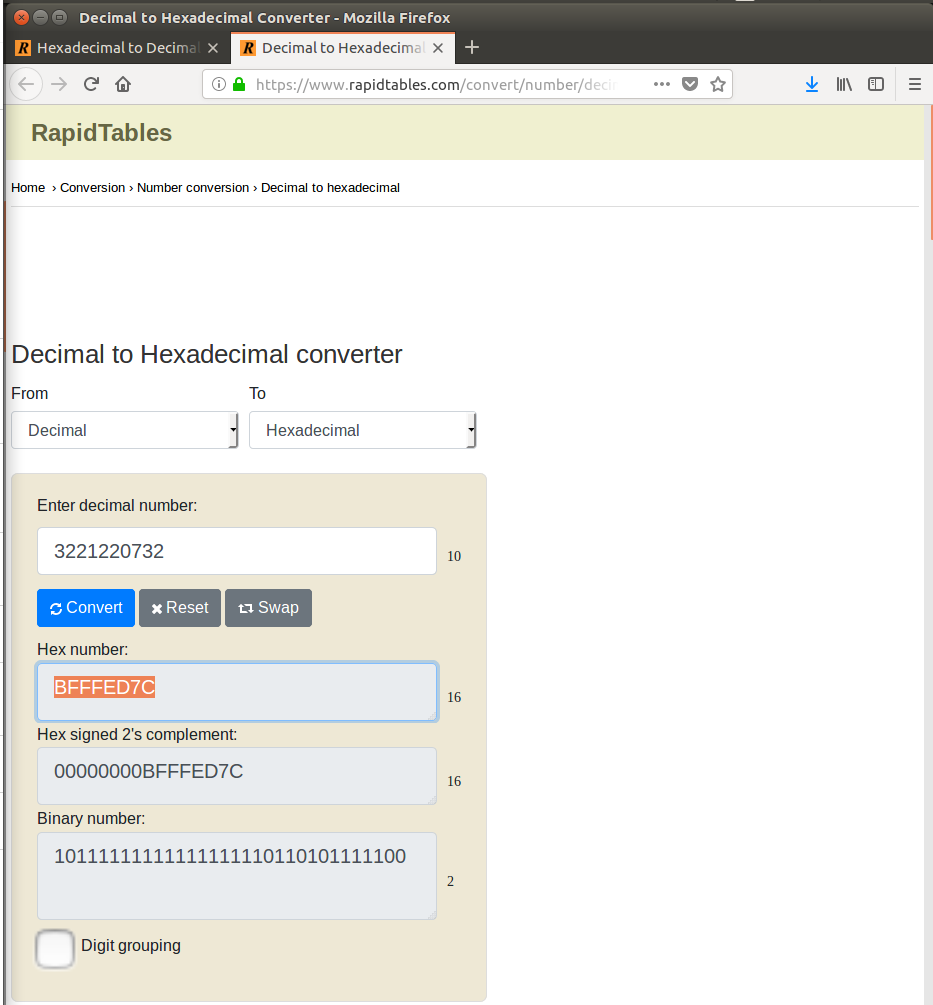
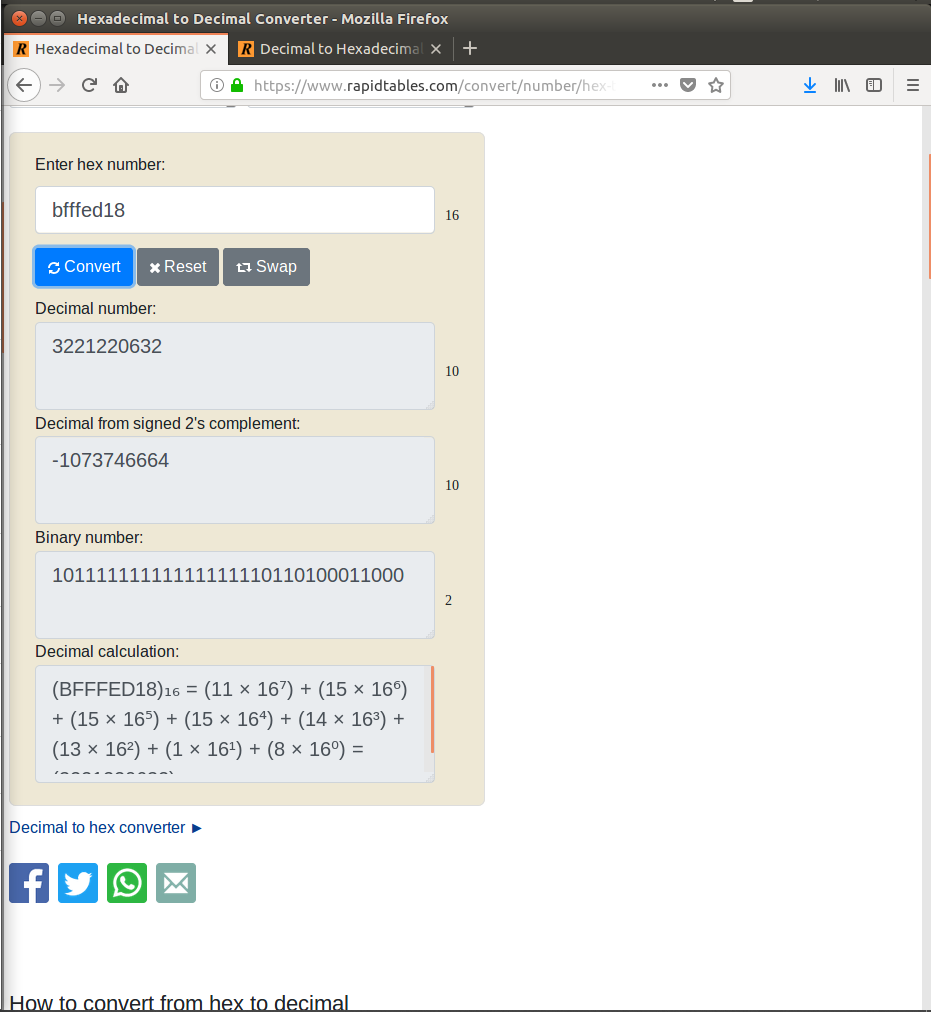
* su root Password
* gcc -o stack -z execstack -fno-stack-protector stack.c
* chmod 4755 stack
* exit

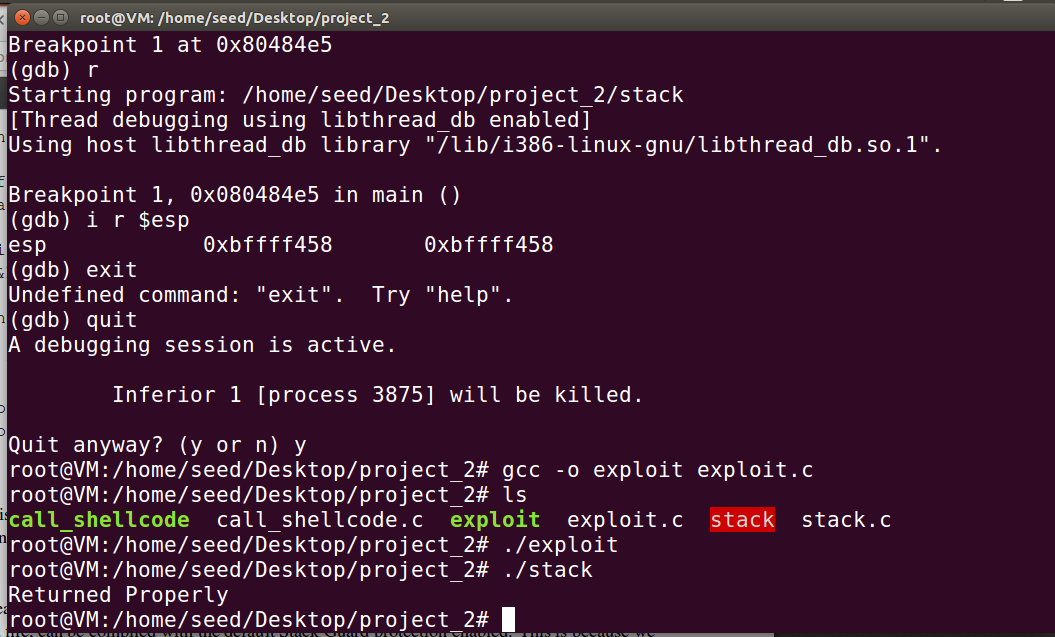
After I was done with those steps, I started the first task – exploiting the vulnerability. I had to use the exploit.c program that was provided to us by the instructor. The code constructs a bad file name “badfile”. In order to do so, I had to create the buffer-overflow part. To do so, I used the function “gdb stack” in order to find the address of the shellcode. After finding the address, I simply added 100 decimal points to the address and copy it to the buffer.

The next step was to compile the program and run the exploit and stack execution files.

**Commands:**

* gcc -o exploit exploit.c
* ./exploit
* ./stack



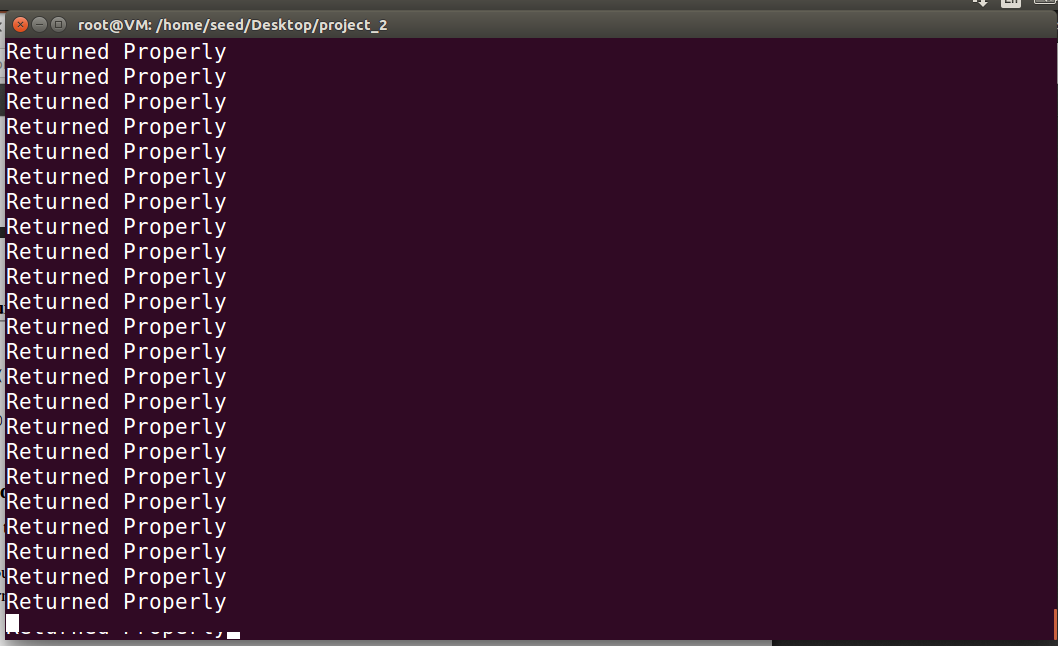


**Task 2 – 2.5**

First, I had to turn the address randomize in order to run the attack (using a loop). I was waiting for 15 minutes to loop to finish but the loop wasn't done. I believe that it takes a while, I just don’t have a fast computer with a lot of memory.

**Commands:**

* su root
* /sbin/sysctl -w kernel.randomize\_va\_space=2
* sh -c "while [ 1 ]; do ./stack; done;"

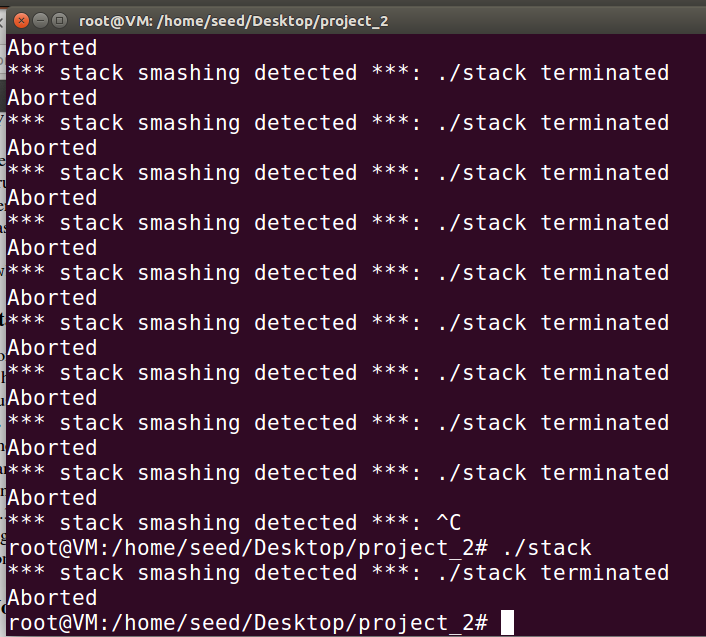


**Task 3 – 2.6**

First, I turned off the address randomize. Then, I compiled the stack.c program without using the fno-stack-protector option which is going to make the stacks executable. In the end, I got the following message (in a loop): “\*\*\* stack smashing detected \*\*\*: ./stack terminated”

**Commands:**

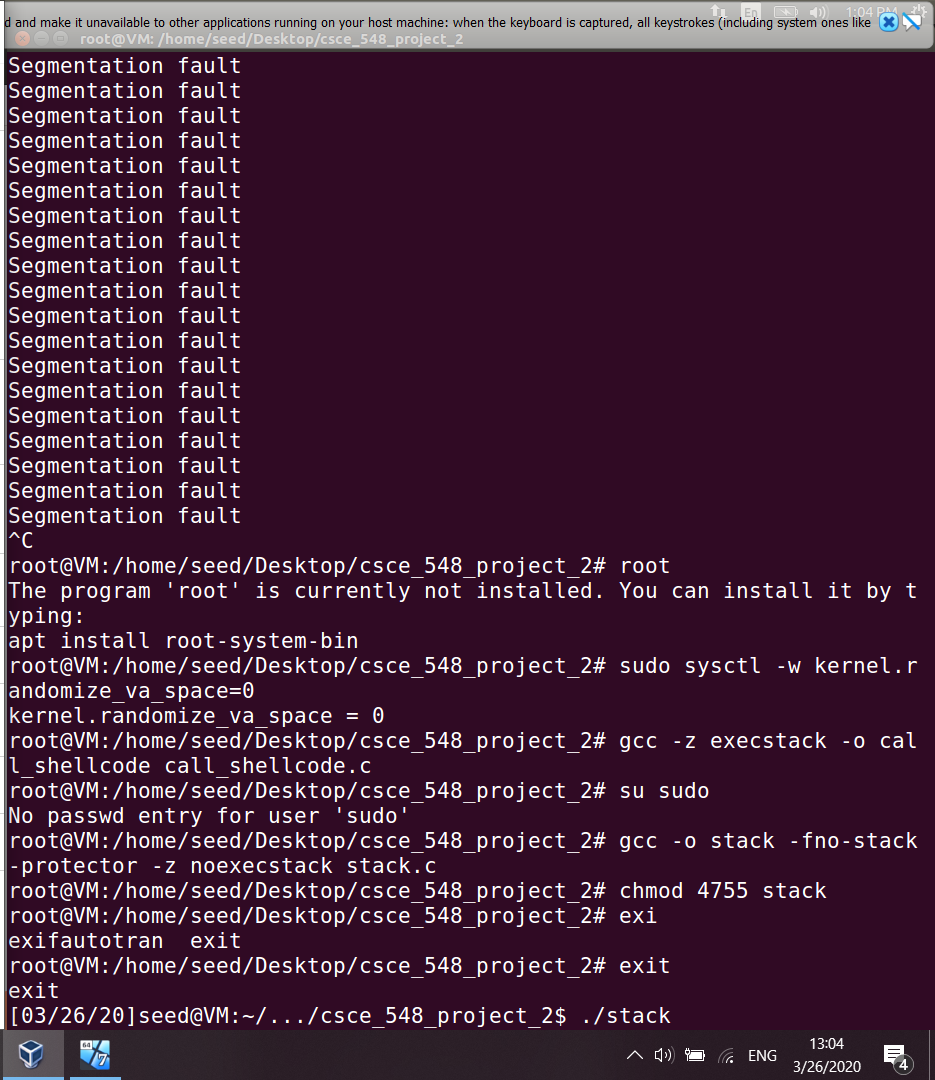
* Su root
* sysctl -w kernel.randomize\_va\_space=0
* gcc -o stack stack.c
* chmod 4755 stack
* exit
* ./stack



**Task 4 – 2.7**

Once again, I had to turn off the address randomization first. Now, unlike task 3, we needed to compile the program using noexecstack option and repeat the first task. I wasn’t able to get a shell but I noexecstack can’t protect from buffer-overflow.

**Commands:**

* Su root
* sysctl -w kernel.randomize\_va\_space=0
* gcc -z execstack -o call\_shellcode call\_shellcode.c
* gcc -o stack -fno-stack-protector -z noexecstack stack.c
* chmod 4755 stack
* exit
* ./stack