Luis Medina

CS4351

Due 10/4/20

**Assignment 2 - Buffer Overflow**

**Problem**

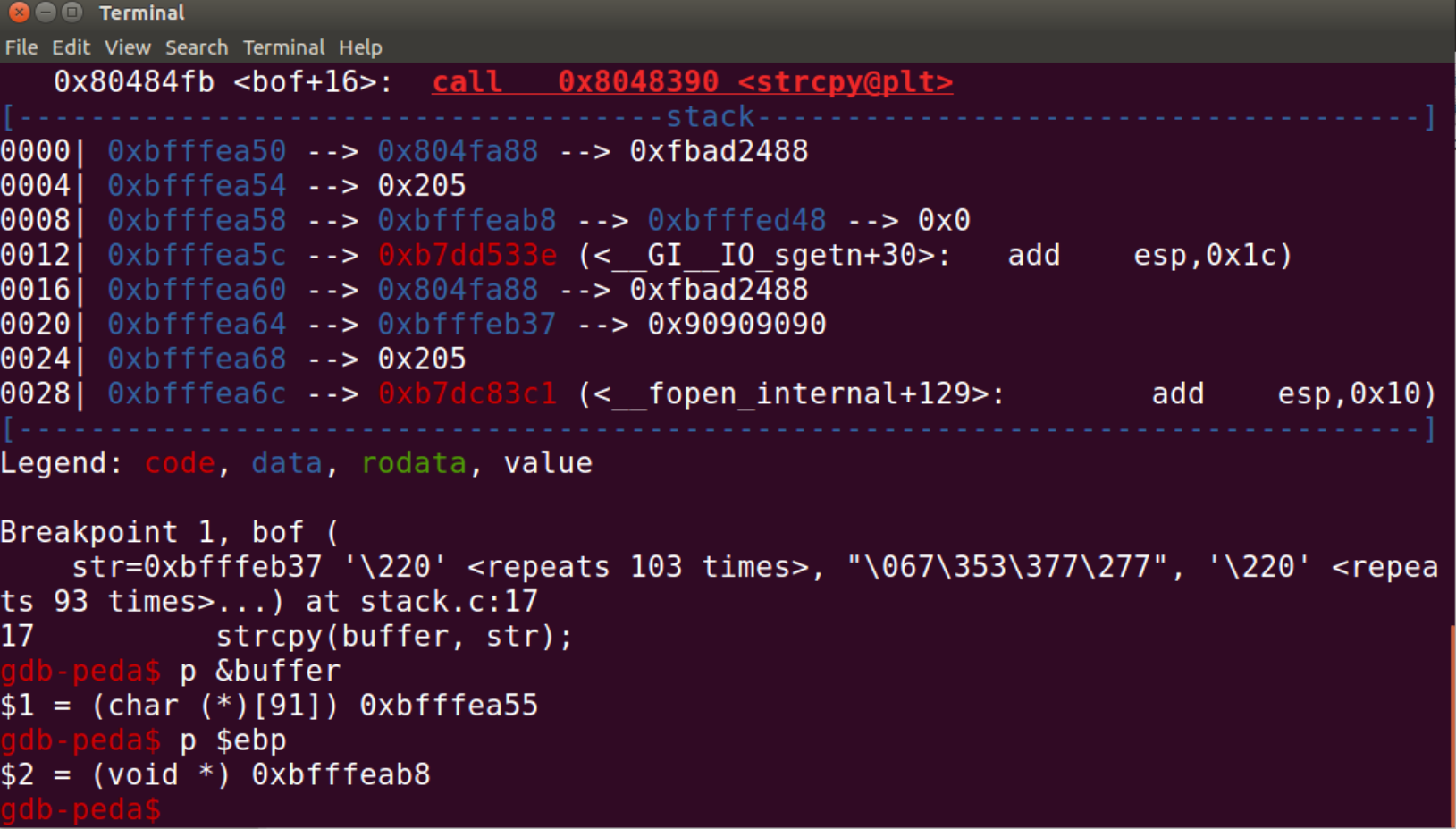
For this assignment we were to perform a buffer overflow attack on the SEED VM using the provided stack.c, exploit.py files and automation script.

**Part A**

1. Disabled address space randomization (sudo sysctl -w kernel.randomize\_va\_space=0).
2. Compiled “stack.c” with stack guard turned off (gcc -DBUF\_SIZE=91 -g -o stack -z execstack -fno-stack-protector stack.c)
   * Interesting note, had to return to this step and add “-g” flag to prevent the offset from being too large.
   * Interesting note: had to set file permissions and ownership to use properly

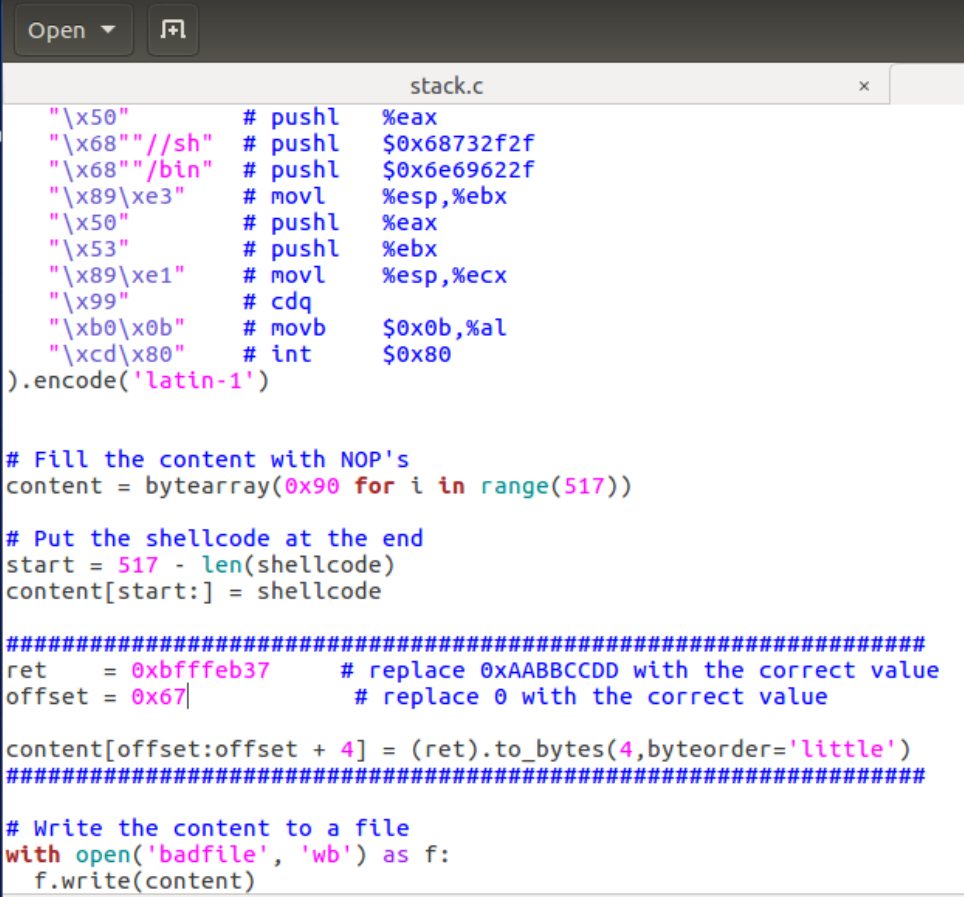
(sudo chown root stack), (sudo chmod 4755 stack)

1. Opened stack.c with gdb (image 1)
   1. Set breakpoint on “bof” function and entered “run”
   2. Printed buffer address
   3. Printed ebp address
   4. Obtained offset: 0xbfffeab8 – 0xbfffea55 = 63 + 4 = 67
   5. Took note of str location: 0xbfffeb37



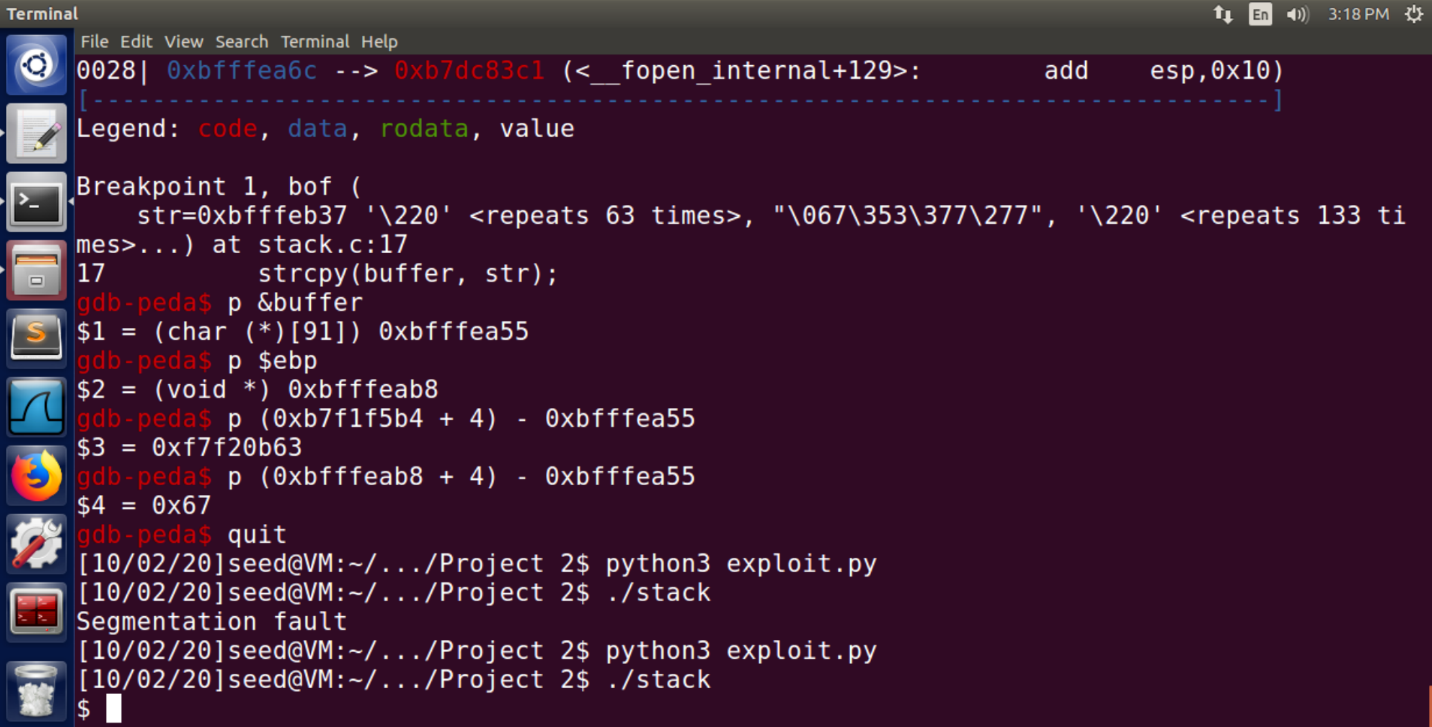
*Image 1*

1. Entered offset and str address onto corresponding values in exploit.py(image 2).



*Image 2*

1. Compiled exploit.py to generate badfile (python3 exploit.py).
   * Interesting note: had to specify python3 as it had trouble with the encoding otherwise.
2. Ran stack program and gained access to shell(image 3).



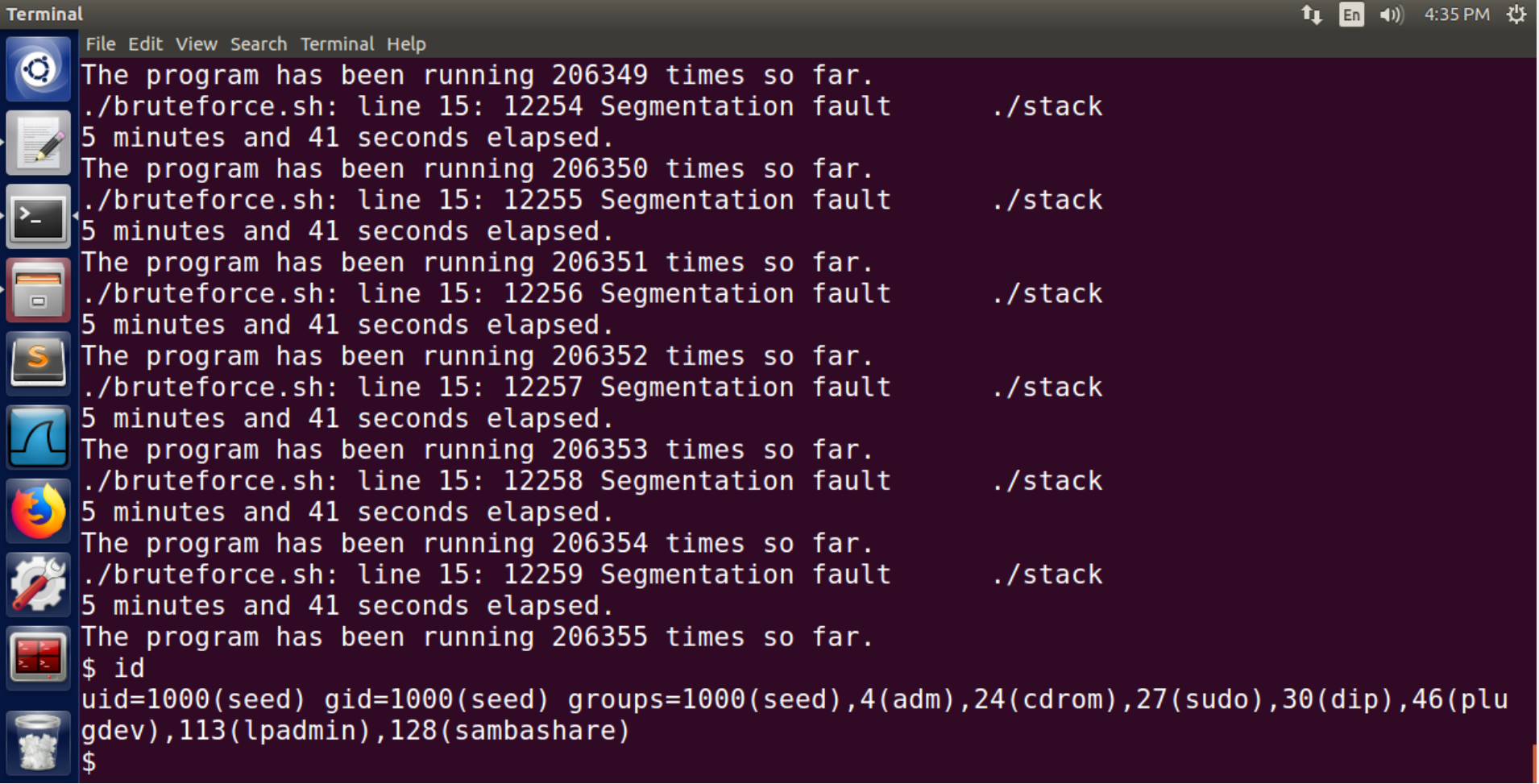
*Image 3*

**Part B**

1. Enabled address randomization (sudo /sbin/sysctl -w kernel.randomize\_va\_space=2)
2. Created a shell script “bruteforce.sh” file using the provided script code to automate running the stack program.
   * Interesting note: had to set file permissions and ownership to use properly.

(sudo chown root bruteforce.sh), (sudo chmod 4755 bruteforce.sh)

1. Ran bruteforce.sh and allowed the program to run for a few minutes.
2. Gained shell access after 5 minutes and 41 seconds after 206,355 tries(image 4)



*Image 4*