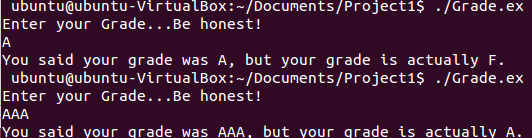
Use gdb to debug program while running. Selected known bad input of scanf (<https://www.owasp.org/index.php/Reviewing_Code_for_Buffer_Overruns_and_Overflows>)

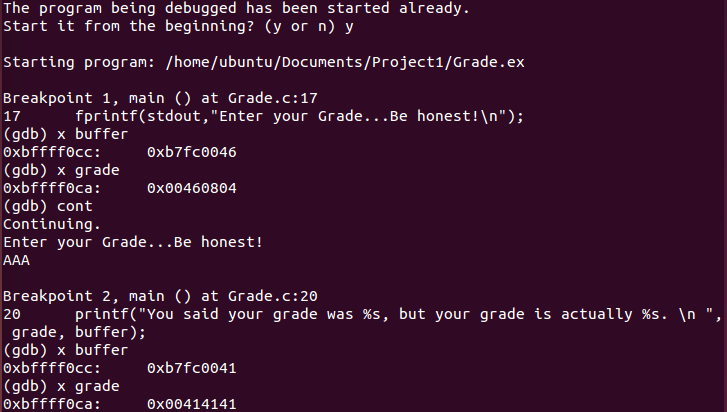
1. The program is a simple program that lets a student enter their grade. The program defaults the grade to F for every student. (Pretend like it pulls from a database.)

The student will enter the grade they have and it will print out the student's grade and the real grade for a comparision integrity check. The scanf function is vulnerable to buffer overflows when reading in as a string. The function is within main, so this overwrites the stack.

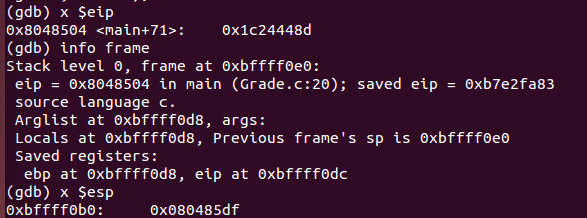


2. To view this, set breakpoints in gdb at the lines after the variables have been declared/filled.

2. The student can overwrite the location of the F variable to whatever grade they want when they enter in more than two characters as input, due to the placement of the variables within the stack. The grade F variable is initialized before the variable for the student input in the program. Which places it below the input variable on the stack.

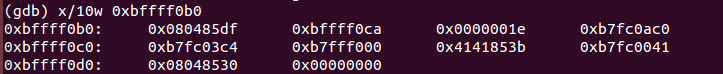
In the above screenshot the variable for “grade' that is input by the user is located at 0xbffff0ca. As the input is read in from scanf, it overflows into 0xbffff0cb and then 0xffff0cc. Buffer was initialized before grade, so it is towards the bottom of the stack.

45. Checking where $eip is in the breakpoint shows we are inside of function Main, therefore the stack of main. Using info stack, I was able to view pertinent information about the stack frame. I used $esp for theaddress at the top of the stack.

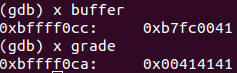
4.75 Then I checked the offset of the two so I could examine the entire stack.



4.9 I then examined the entire stack of 40 bytes, by printing 10 words of 4 bytes each.



The letter a is represented by hexadecimal 41, and you can see the three input A's.



4. with the rearranged variables, the buffer doesn't get overwritten due to the way the memory is laid out because grade grows downward with input, and buffer exists above it on the stack.

