Daniel Quiroga

Network Security

Quiz 2

1.

A: .data

B: stack

C: stack

D: heap

E: stack

F: stack – since it is being passed in as an argument

G: stack as a stack frame

2.

Line 1 is establishing the stack frame, and this is used to keep track of what function you are in (the calling function) and make it available to go back to the original place once finished with the function.

Line 2 is moving the base pointer to the stack pointer. This is establishing the stack frame so that the function can perform all the operations that it will need to return properly locally and then with the return address.

Line 3 is saving the value of whatever was stored in ebx. This may need to store the value of a previous stack frame that may be useful for the current function call that we are in.

Line 4 is used to allocate stack space for the function call, and this is where all our variables and operations will be held in the stack. This is important to maintain all information related to the stack in the stack frame and organize the operations.

Line 5 is adding a local variable in the function to the stack frame created. This is needed to keep track of all variables inside the function and then use it when needed.

3.

Line 1 does nothing other than move the pointers to align with where we may need them to be. This is needed to prepare to exit our function call in this example.

Line 2 is moving a constant value to a variable that may used somewhere else in the program but does not have to be in the current stack. We need this is ordered to maintain some global constants or move information from one function to another. This is used to keep track of the return value in 32 bit calling convention.

Line 3 moves the stack pointer to the base pointer essentially freeing up the stack frame for the function. This is essential to close out the function and return to the program.

Line 4 is used to restore the called function pointer to the base pointer. This is needed to return to the previous stack frame that the program was in before entering this function.

Line 5 is returning to the position in the code where this was called. This is needed to continue with our instructions and operations in this program.

4.

In 32-bit mode the arguments are all stored on the stack and read right to left while in 64-bit mode the first 6 arguments are stored in 6 registers and the rest are stored on the stack read right to left.

5.

The following byte sequence would create a buffer overflow since it would occupy the 4 byte buffer, local integer and then get one byte of the ebp which we would be able to redirect the return address to someplace that we may have more access too and hence can do some damage.

3b 65 69 a4 00 ff ff ff 3d 5d 4e 2a e3