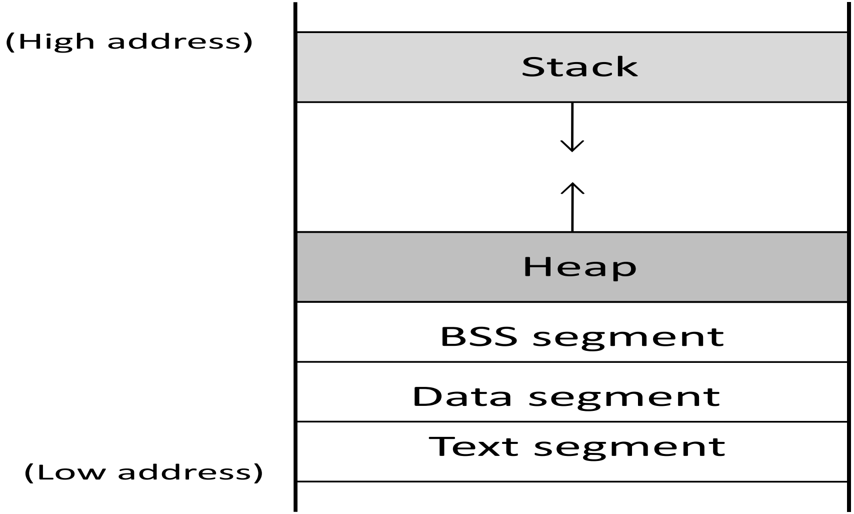
Agenda

- Buffer Overflow

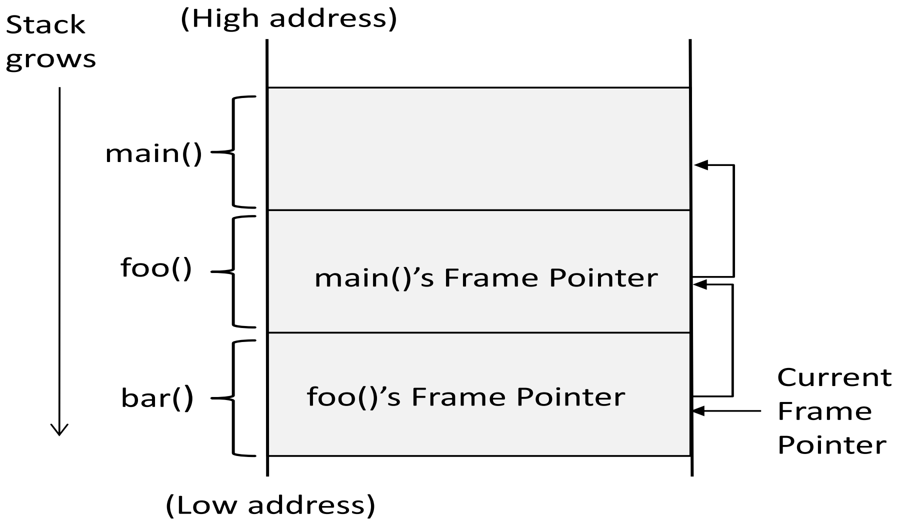
Terminology

- `Buffer Overflow`: during memory copying, more data copied to the destination buffer than the amount of allocated space. Consequence: program crash, or arbitrary code execution - the logic of the program will be different from the original one.
- `Stack`: used for storing local variables defined inside functions, as well as storing data related to function calls, such as return address, arguments, etc.
- `Stack Frame`: when a function is called, a block of memory space will be allocated on the top of the stack, and it is called stack frame.
- `Return address`: the address following the call instruction. When a function finishes and hits the return instruction, it needs to know where it returns to.

Program Memory Layout



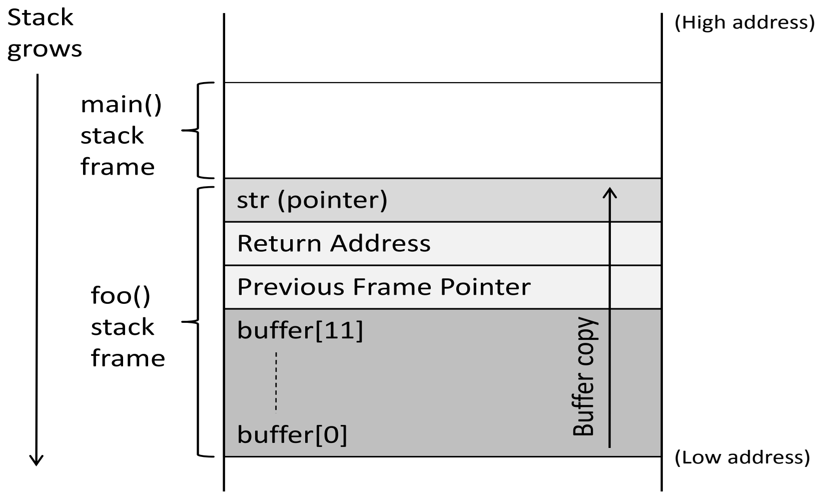
Stack Layout for Function Call Chain



Buffer Overflow Example 1

http://cs.boisestate.edu/~jxiao/cs333/code/overflow.c

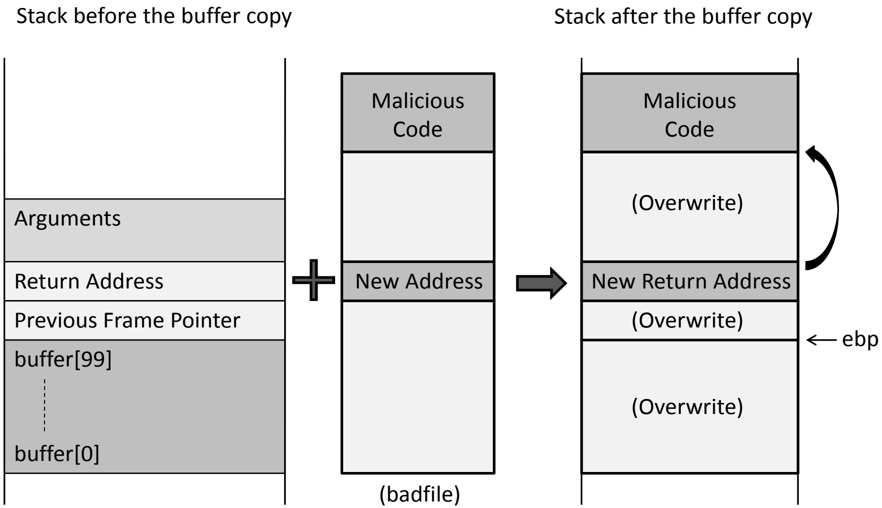
Buffer Overflow Example 1



Buffer Overflow Example 2

http://cs.boisestate.edu/~jxiao/cs333/code/stack.c

Buffer Overflow Example 2



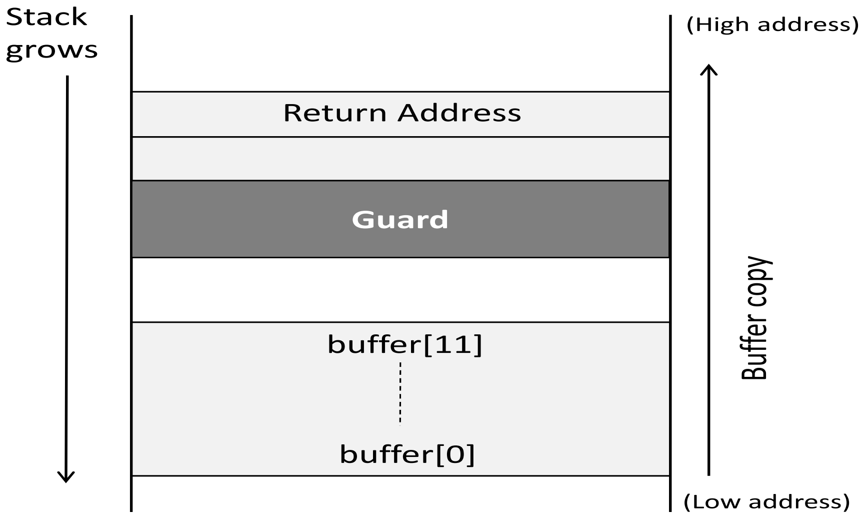
Commonality among Heartbleed, Shellshock, and Buffer Overflow

- Programs blindly trust user input(s).

Defense Against Buffer Overflow

- Address Space Layout Randomization (ASLR): `http://cs.boisestate.edu/~jxiao/cs333/code/aslr.c`
- Write XOR Execute (in Windows, this is called Data Execution Prevention or DEP).
- StackGuard: `http://cs.boisestate.edu/~jxiao/cs333/code/stackguard.c`

StackGuard


Stack
grows
(High address)
Return Address
Guard
buffer[11]
buffer[0]
Buffer copy
(Low address)

References

A large portion of the material is adapted from:

- Computer Security - A Hands-on Approach by Wenliang Du