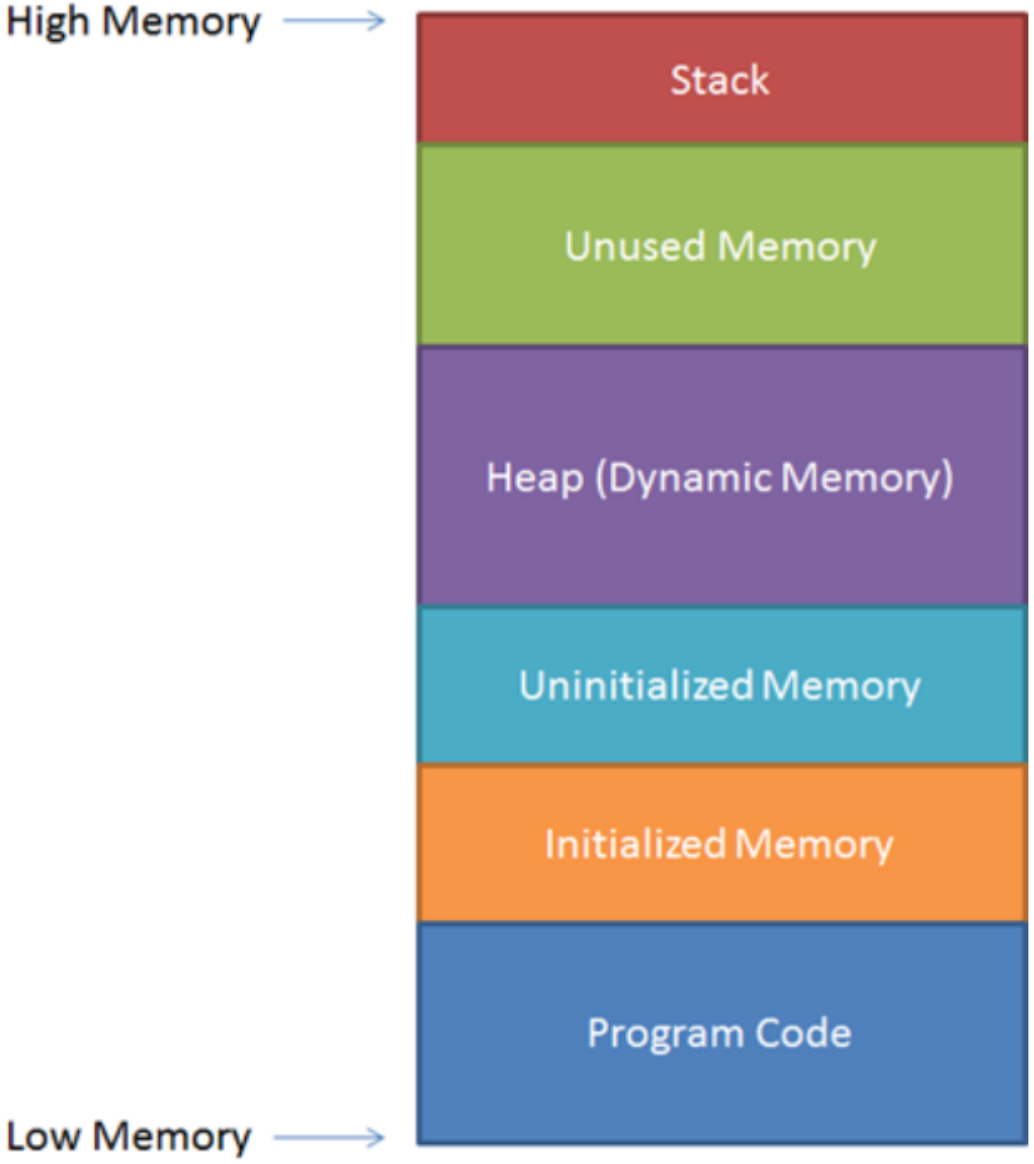
Assignment 1

Buffer Overflow & Race Conditions

Security Tools Lab 2

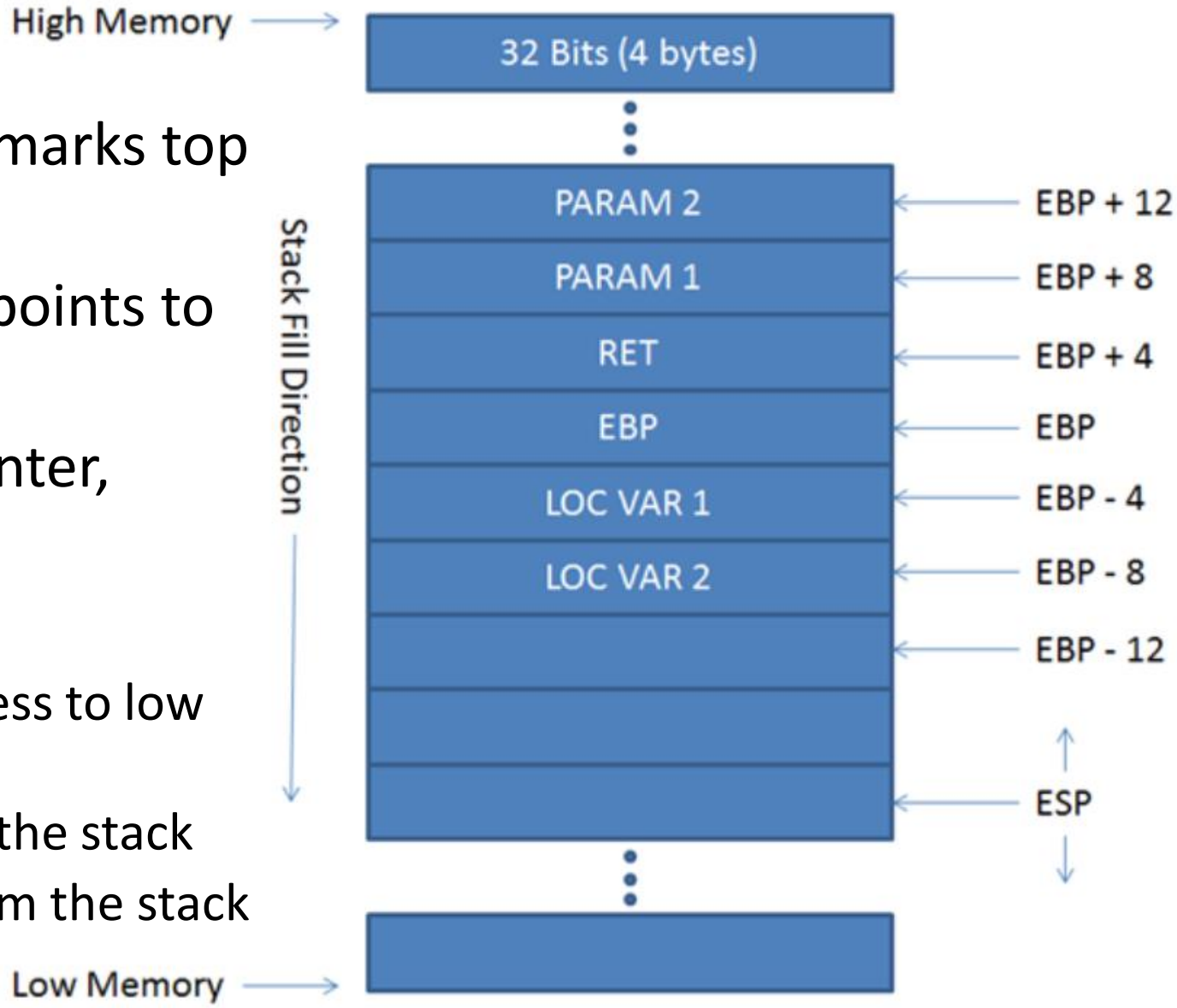
Memory Allocation

- Stack & heap are both in RAM
- Stack -> static memory allocation
  - Assigned at compile time
  - Grows from high memory address
- Heap -> dynamic memory allocation
  - Assigned at run time
  - Grows from low memory address
- Stack
  - LIFO (Last in First out)
  - Temporary storage to quickly access data used in a program
  - Registers store address that point to other positions in memory
  - ESP, EBP, EIP, etc



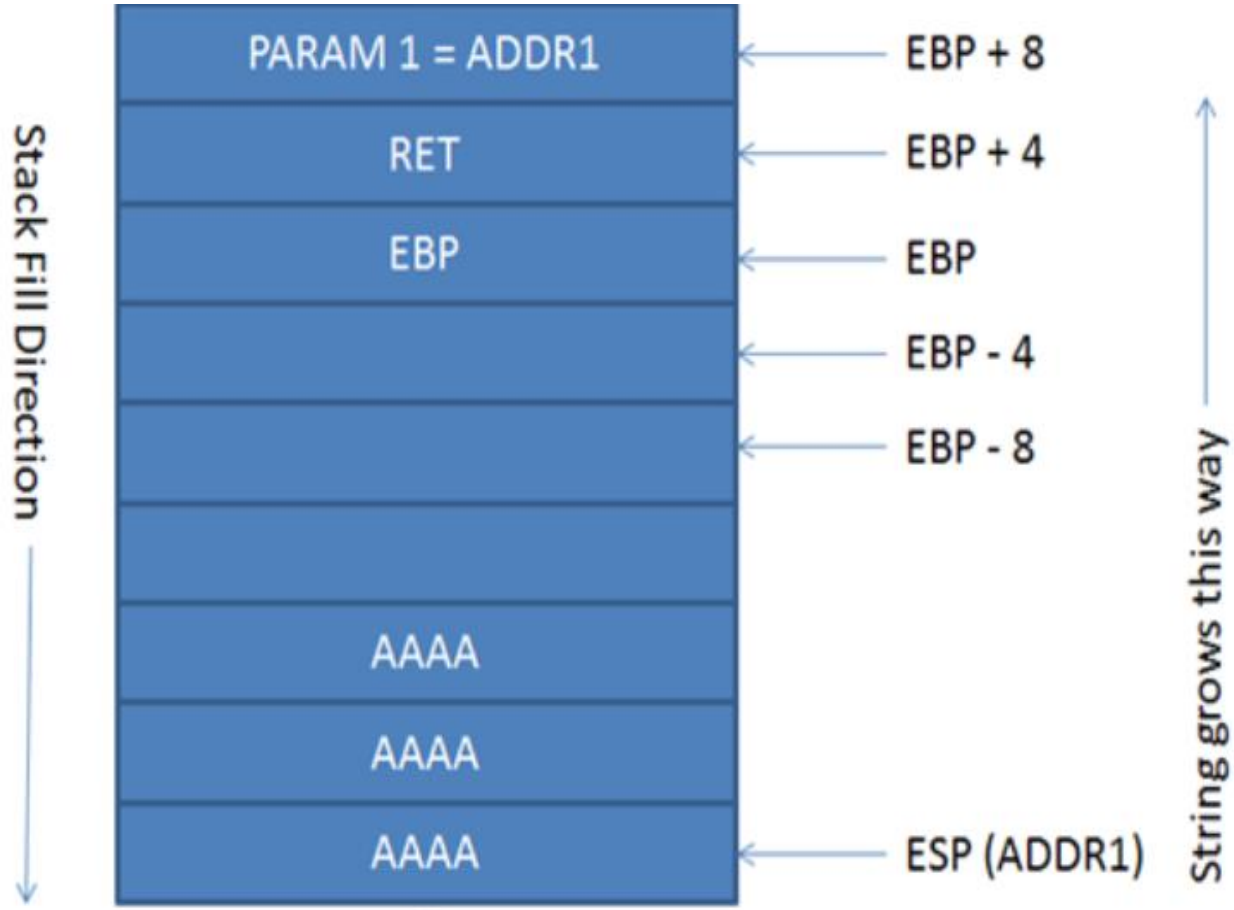
Memory Allocation

- ESP – extended stack pointer, marks top of stack
- EBP – extended base pointer, points to base of stack (the anchor)
- EIP – extended instruction pointer, points to next instruction
- Stack
  - Grows from high memory address to low memory address
  - Grows when we ‘PUSH’ data in the stack
  - Shrinks when we ‘POP’ data from the stack



Buffer Overflow

- When a function is called:
  1. Stack created, Insert EBP in stack to set anchor
  2. Parameters (argc, argv) are passed to EBP+8 …
  3. Func called, Return data pointed by RET at EBP+4
- ESP points to top of stack
- String is copied from ADDR1
- If function doesn't control length of buffer, we can overwrite whole stack
- Buffer Overflow = Program crashes

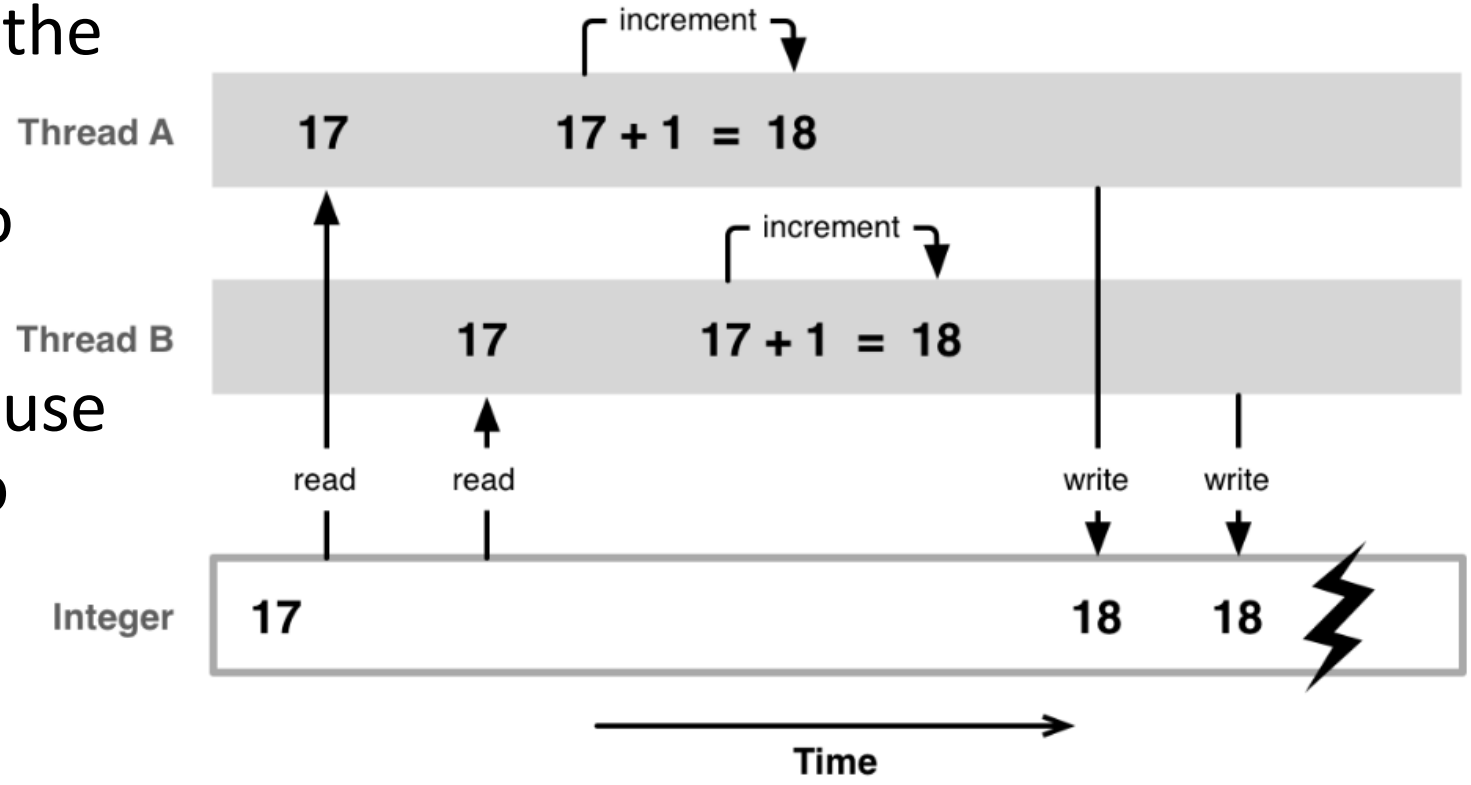


Protection

- Canaries
  - Random values placed on stack after buffer
  - Checking the value against original value to determine buffer overflow
- ASLR
  - Address space layout randomization
  - Make offsets harder to determine

Race Condition

- Occurs when two or more threads can access shared data and they try to change it at the same time
- Both threads are "racing" to access/change the data
- To prevent race conditions, use a lock on the shared data to ensure only one thread can access the data at a time



Race condition

- Dirty COW (Copy-On-Write)
- One thread tries to write to a read only memory location, creating a modified copy in the process
- A second thread uses a function called *madvise* to tell the kernel that newly allocated memory is not needed
- By executing these two threads simultaneously in a loop, the kernel eventually gets tricked into pointing to the modified copy of a file in memory that should be read only