Homework: HW 1 posted.
- victim1 is buffer overflow
- victim2 is format string attack
- Demo to be given on Oct3. Sign up online.

Return Oriented Programming
- find gadgets in original binary code
  Gadget: consists of insn and ends with ret
  insn
  insn
  .
  .
  insn
  ret
- String together the gadgets to accomplish work.
- FACT: There are enough gadgets in the libc to do any computation

LOAD inst %eax address
pop %eax
ret

STORE address, %edx
pop %edx
mov (%edx), %eax
ret

- Doesn't require execution of any code.
- Way of doing ROP in real time
  - Use gadgets to call mmap to allocate some writable and executable memory
  - write some instructions into newly allocated space
  - Jump to them

Address space (Layout) Randomization:
- Defense against all kind of buffer overflows
  - code injection
  - return to libc
- ROP
  - etc
- Place stack at random in VM space
- adjust %sp (Stack Pointer) to match
• For code injection this works because the stack location isn’t known
• For return-to-libc, attackers know the libc address as randomization is only of stack location
  o in this passing arguments might be a bit difficult
  o but there is a work around. Say a program stores username in heap. We give 1000s of username “/bin/sh” and get our arguments to system call

  ![Stack Diagram]

• Alternative?
  o Also Randomize (this has no overhead. No effect on performance)
    ▪ location of all libraries (they have to page aligned)
    ▪ location of heap
    ▪ location of mmap
  o Downside is fragmentation of VM address space
• Major issue with ASLR (atleast in Windows)
  o Some libraries must be loaded at a fixed address
  o Attackers can do ROP with these fixed address libraries as source of reliable gadget.