CSE 484 - Section 1

Topics

•Gdb
•Buffer overflow
•Format strings

Roy McElmurry

Assembly

```
int foo(char *argv[])

{

  char buf[128];

  strcpy(buf, argv[1]);

}


int main(int argc, char *argv[])

{

  foo(argv);

  return 0;

}
```

- How is this implemented?
- What is on the stack?

x86

- General purpose registers
  - eax, ebx, ecx, edx, esi, edi, …
- Stack registers
  - esp: points to the top of the stack
  - ebp: points to the bottom of the current frame
- Instruction Pointer
  - eip

x86

- Stack manipulation instructions:
  - PUSH arg: ESP - 4 -> ESP, arg -> @ESP
  - POP arg: @ESP -> arg, ESP + 4 -> ESP
- Function call instructions:
  - CALL addr: PUSH EIP, JMP addr
  - RET: POP EIP
  - LEAVE: EBP -> ESP, POP EBP

Gdb

- Useful commands
  - info registers (i r)
  - info frame (i f)
  - disassemble, list
  - catch exec
  - run
  - continue
  - break
    - b foo
    - b foo+10
    - b *0x080fcde8
  - step, stepi, next
  - Examine
    - X var
    - X $reg
    - X 0xdeadbeef
    - X/20i main
    - X/40x var
  - print

.gdbinit

- You can write custom functions when you find yourself repeating the same tasks
- Similar to a .bash_rc file