Upcoming CSG Topics/Events

● March 8th: Reverse Engineering presentation
● March 22nd: Presentation by Citi (topic tbd)
● March 25th: Cigital CTF (on campus)
  ○ Registration: https://ctf.findthe.ninja/
● March 24th - 26th: Volga CTF (online)
● March 29th: Binary Exploitation (HACK THE PLANET!)
Introduction to x86 Assembly Architecture

March 1st, 2017
Overview

- Big Picture
- Assembly Languages
- x86 Assembly
- Assembly analysis
Looking at the Bigger Picture
Looking at the Bigger Picture

- Most programmers are concerned with code written in higher-level languages
Looking at the Bigger Picture

- Most programmers are concerned with code written in higher-level languages.
- However, code has no effect until it is compiled.
  - Must be translated into machine code (binary) by compiler in order for processor to perform functionality.
  - Machine code not human-readable.
Assembly Language

Assembly (ASM) language - lowest-level language, readable by humans

- Intermediary step between higher-level code (like C) and machine code
- Almost 1-1 correspondence to between ASM instructions and processor architecture instructions
- Multiple architectures → Different ASM instruction sets

For the next few weeks, we will use 32-bit Intel x86
X86 Assembly Architecture

Developed by Intel for the 8086 and 8088 Intel CPUs (16-bit)

Still widely in use today

- Used for XBOX, Core i3/i5/i7, Apple’s MacBook line, etc.
- Continual refinement and community contributions keep x86 as leading architecture

Little-endian byte ordering
x86 Registers

- EAX - Accumulator register
- EBX - Base register
- ECX - Counter register
- EDX - Data register
- EBP - Stack base pointer (sometimes called frame pointer)
- ESP - Stack pointer (top of stack)
- ESI - Source index
- EDI - Destination index
- EIP - Instruction pointer
x86 Instructions

Format

\[
\text{inst <source>, <destination>}
\]

Ex) add eax, 0x8

Instructions are generally mnemonics or resemble the operation they perform

- Add → add, Subtract → sub
- Jump Less than or Equal to → jle

WAY too many instructions to learn them all (about 9000 by some estimates), so we will learn the basic/common commands
x86 Instructions

Mathematical instructions

- add, sub, mul, div, or, xor, and, inc, dec

Comparison/Assignment instructions

- cmp → compare, mov → assign value or pointer to a register
- lea → load effective address

Calling/Conditional instructions

- call, ret, leave
- jmp, jg, jge, jle, jl, jz, jnz

nop
The x86 stack
The x86 stack
The x86 stack
The x86 stack

Stack structure used in processor to quickly access data

- “Grows” downward
- Can contain multiple stack frames
  - Usually occurring as result of multiple functions within code
  - Managed using ESP and EBP registers
The x86 stack
An error has occurred. To continue:

Press Enter to return to Windows, or

Press CTRL+ALT+DEL to restart your computer. If you do this, you will lose any unsaved information in all open applications.

Error: 0E : 016F : BFF9B3D4

Press any key to continue _
How to view x86 assembly

Usually achieved with debugger/disassembler

- Useful for looking through assembly instructions to find exact behavior of the code

GNU Debugger (GDB)
IDA Pro
Binary Ninja

We will be using gdb today
GDB Commands

- **X - Examine**
  - `x/u addr` → print value @ addr as base-10 decimal/int
  - `x/s addr` → print value @ addr as a string
  - `x/x addr` → print value @ addr in hexadecimal
  - `x/i register` → print instruction stored in register

  Can include amount to be printed (`x/2x` → 4 bytes in hexadecimal) or size of unit (`x/2xw` → prints 2 words, or 2*4-bytes, in hexadecimal)

- **info [register | registers | breakpoints]** - view contents of a register, all registers, or the list of breakpoints
GDB Commands continued

- list - if file was compiled with -g flag, will display source code
- break [func_name | line_number | *addr] - stops execution at indicated location
- run
- continue - continue execution until next breakpoint
- step - execute next instruction or step into called function
- next - next instruction without stepping into called function
- kill - end execution of program
- quit - exit gdb
Demo Time
Resources

- Wikibooks: https://en.wikibooks.org/wiki/X86_Disassembly/The_Stack
- Unixwiz: http://unixwiz.net/techtips/win32-callconv-asm.html
- GDB and GCC man pages
- IDA Pro: https://www.hex-rays.com/products/idapro/index.shtml
- Binary Ninja: https://binary.ninja/