SWE3001: System Program

Lecture 0x00: Course Overview and Introduction

Hojoon Lee

Systems Security Lab

SUNGKYUNKWAN UNIVERSITY 1398 성균관대학교

Instructor: Hojoon Lee (이호준)

- Research Areas
  - Software Security
    - Vulnerability analysis
    - Attacks and Defenses
  - Systems Security
    - Trusted Computing
    - Secure Computation for AI
- Leader of Systems Security Lab @ SKKU
  - `https://sslab.skku.edu`
  - Hiring MS/PhD students and undergraduate interns
  - Looking for CTF(Capture the Flag) Team members

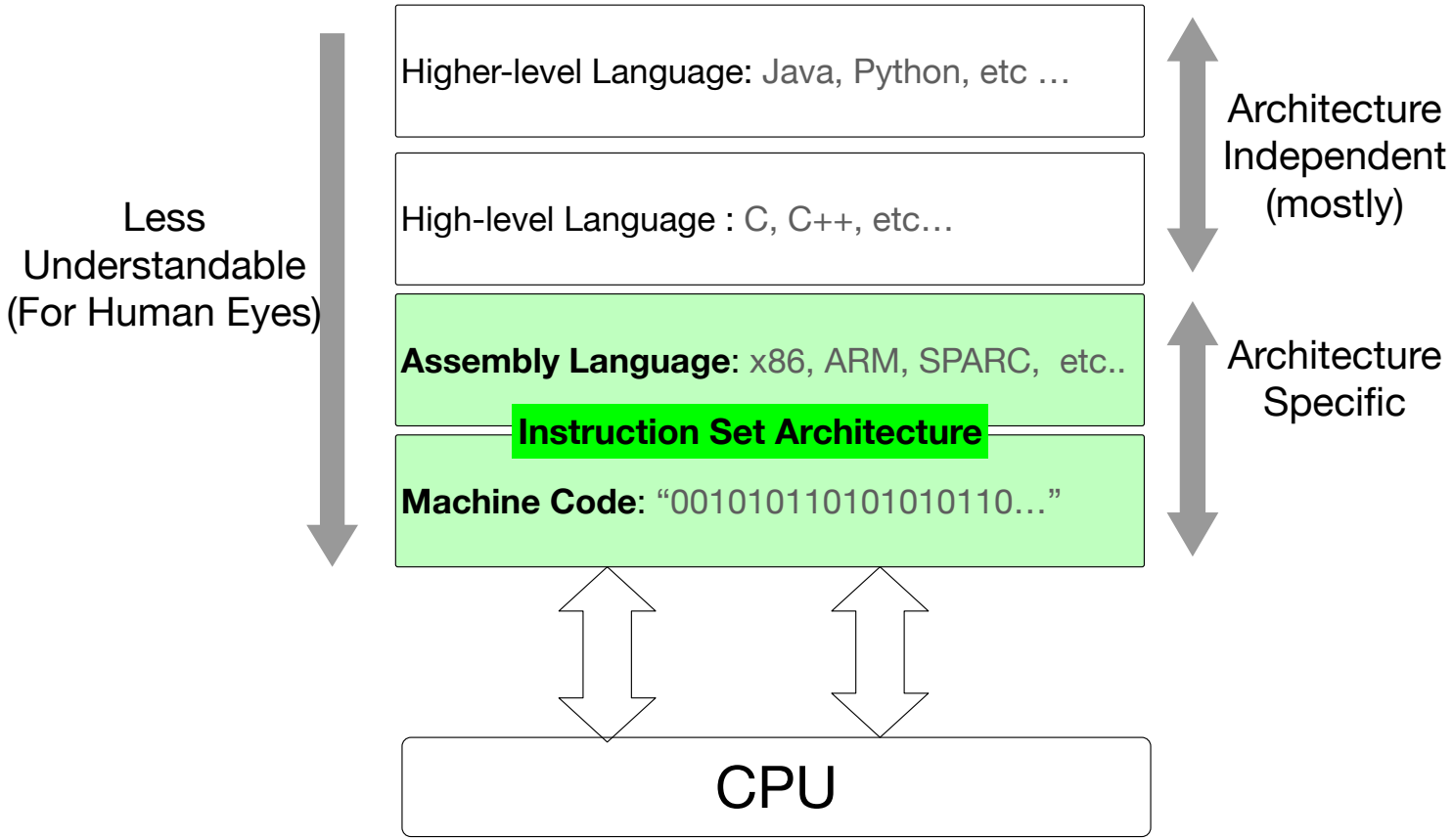
Systems Security Lab

SUNGKYUNKWAN UNIVERSITY 1398 성균관대학교

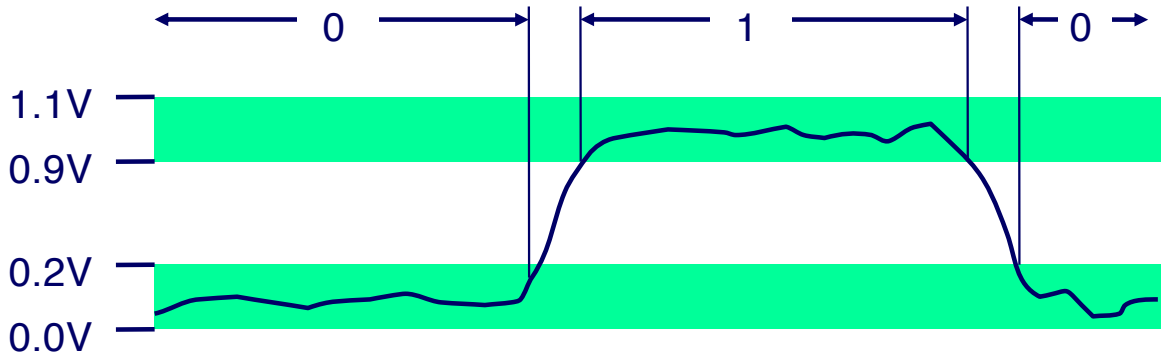
What Is This Course about?

- Introduction to "How Computer Really Works"
- One of the most foundational course in computer science

Getting Low


Higher-level Language: Java, Python, etc …
High-level Language : C, C++, etc…
Assembly Language: x86, ARM, SPARC, etc..
Instruction Set Architecture
Machine Code: "0010101101010101 10…"
CPU
Less Understandable (For Human Eyes)
Architecture Independent (mostly)
Architecture Specific

Representing Information with 0s and 1s



- Computers save and transmit information in bits
- Bits represent "voltage" or "1" vs "no voltage" or "0"

Representing Information with 0s and 1s

- You will learn to work with binary numbers
- For example
  - How do we represent strings in binary?
  - How do we represent negative numbers?
  - Hexadecimal?

X	B2U(*X*)	B2T(*X*)
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	–8
1001	9	–7
1010	10	–6
1011	11	–5
1100	12	–4
1101	13	–3
1110	14	–2
1111	15	–1

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Understanding Programs at Binary-level

```c
#include <inttypes.h>
#include <stdio.h>

int main() {

  int number1, number2, sum;

  printf("Enter two integers: ");
  scanf("%d %d", &number1, &number2);

  // calculating sum
  sum = number1 + number2;

  printf("%d + %d = %d", number1, number2, sum);
  return 0;
}
```

```
0x0000000000401136 <+0>:     sub    rsp,0x18
0x000000000040113a <+4>:     lea    rdi,[rip+0xec3]        # 0x402004
0x0000000000401141 <+11>:    mov    eax,0x0
0x0000000000401146 <+16>:    call   0x401030 <printf@plt>
0x000000000040114b <+21>:    lea    rdx,[rsp+0x8]
0x0000000000401150 <+26>:    lea    rsi,[rsp+0xc]
0x0000000000401155 <+31>:    lea    rdi,[rip+0xebd]        # 0x402019
0x000000000040115c <+38>:    mov    eax,0x0
0x0000000000401161 <+43>:    call   0x401040 <__isoc99_scanf@plt>
0x0000000000401166 <+48>:    mov    esi,DWORD PTR [rsp+0xc]
0x000000000040116a <+52>:    mov    edx,DWORD PTR [rsp+0x8]
0x000000000040116e <+56>:    lea    ecx,[rsi+rdx*1]
0x0000000000401171 <+59>:    lea    rdi,[rip+0xea7]        # 0x40201f
0x0000000000401178 <+66>:    mov    eax,0x0
0x000000000040117d <+71>:    call   0x401030 <printf@plt>
0x0000000000401182 <+76>:    mov    eax,0x0
0x0000000000401187 <+81>:    add    rsp,0x18
0x000000000040118b <+85>:    ret
```

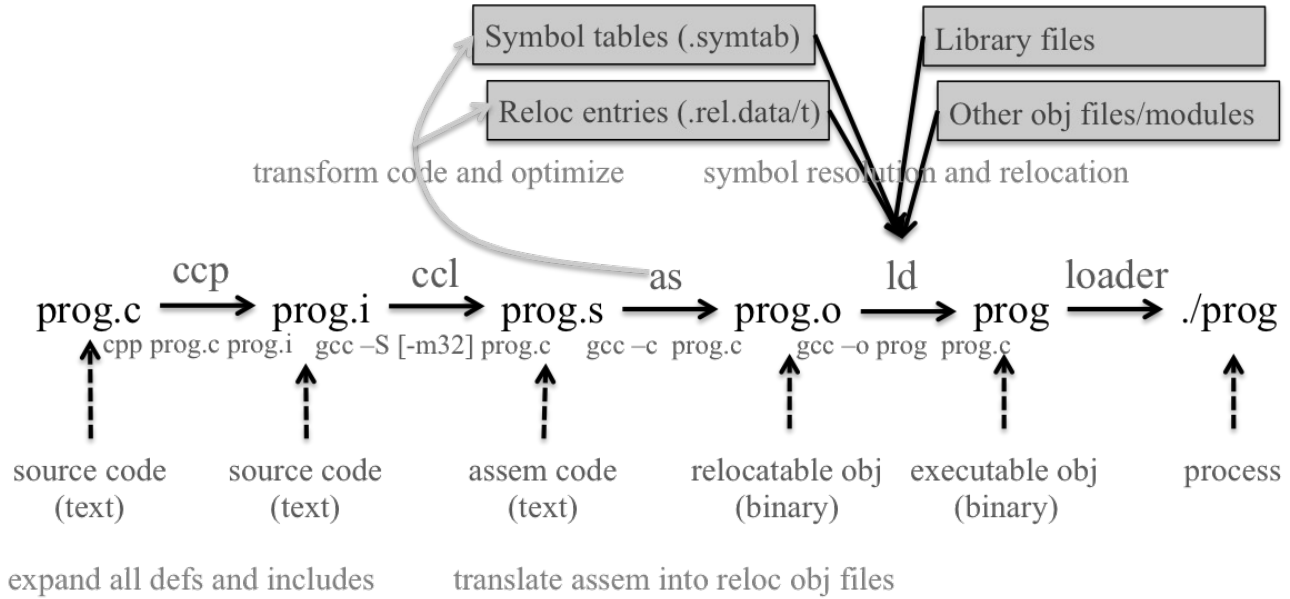
Systems Security Lab

SUNGKYUNKWAN UNIVERSITY 1398 성균관대학교

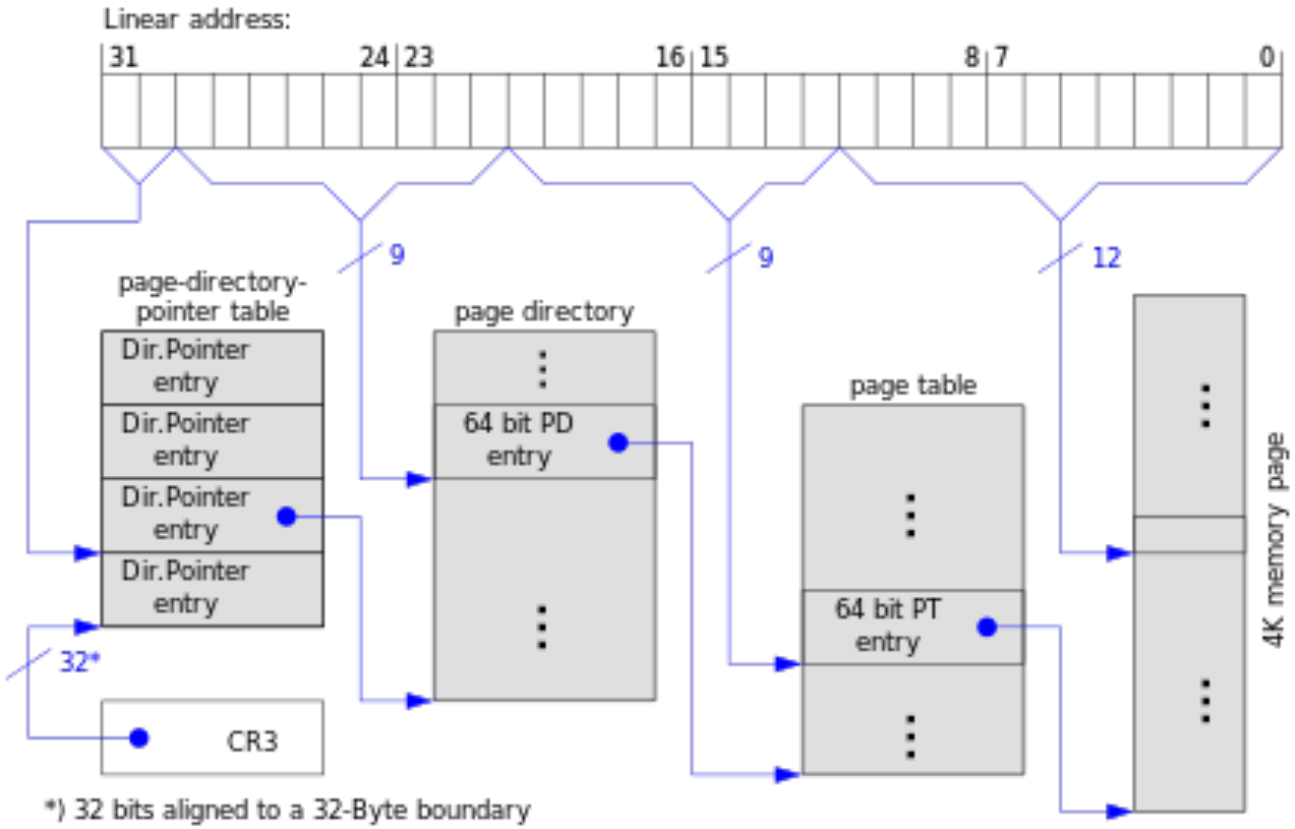
Understanding Programs at Binary-level

- Understanding binary programs requires understanding of
  - (x86) Instruction Set Architecture (ISA)
  - Common patterns of building logics with instructions
  - Application Binary Interface (ABI)
  - ETC..
- More importantly, lots of reading programs
  - Assembly *Language* is a language
  - Just like human languages, it takes time and practice to be fluent in assembly languages

Linking


Symbol tables (.symtab)
Library files
Reloc entries (.rel.data/t)
Other obj files/modules
transform code and optimize
symbol resolution and relocation
ccp
ccl
as
ld
loader
prog.c
prog.i
prog.s
prog.o
prog
./prog
cpp prog.c prog.i
gcc –S [-m32] prog.c
gcc –c prog.c
gcc –o prog prog.c
source code (text)
source code (text)
assem code (text)
relocatable obj (binary)
executable obj (binary)
process
expand all defs and includes
translate assem into reloc obj files

Virtual Memory



The Tools

- Get familiar with Linux and command line
  - (You can't avoid it in this course)
  - Create a Linux virtual machine before semester start using {VMWare, VirtualBox} if you are not using Linux already
- Learn to use debuggers (GDB)
  - Debuggers allow you to observe program execution at binary level
  - You will learn how to use GDB in this course

Why Do I Need to Learn All This?

Systems Security Lab

SUNGKYUNKWAN UNIVERSITY 1398 성균관대학교

Low-level Knowledge Helps You Write Better Code

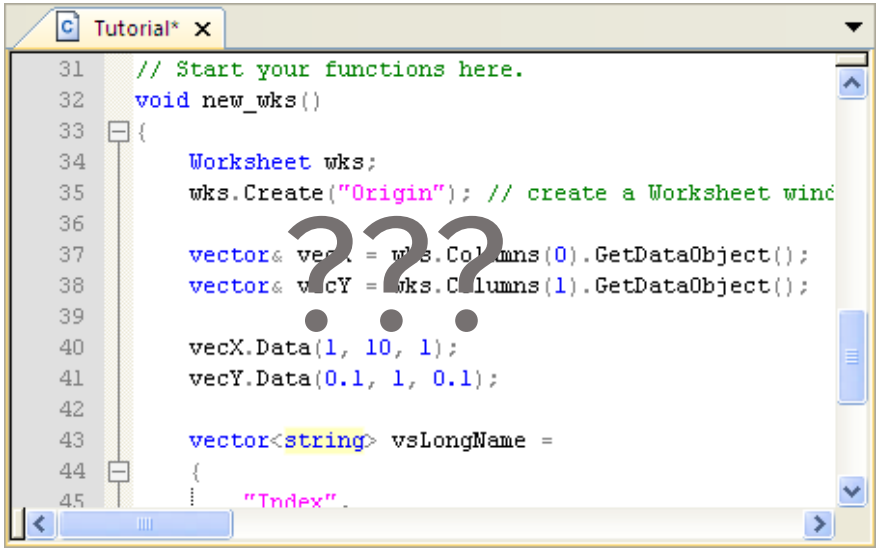
- If you know "how computer REALLY works" below algorithm level …
- It gives you the intuition into how your program will interact with hardware
- You will natrually see why Code A will run faster than Code B

More Effective Debugging

```
00000000                push    ebp
00000001                mov     ebp, esp
00000003                movzx   ecx, [ebp+arg_0]
00000007                pop     ebp
00000008                movzx   dx, cl
0000000C                lea     eax, [edx+edx]
0000000F                add     eax, edx
00000011                shl     eax, 2
00000014                add     eax, edx
00000016                shr     eax, 8
00000019                sub     cl, al
0000001B                shr     cl, 1
0000001D                add     al, cl
0000001F                shr     al, 5
00000022                movzx   eax, al
00000025                retn
```

The Bug

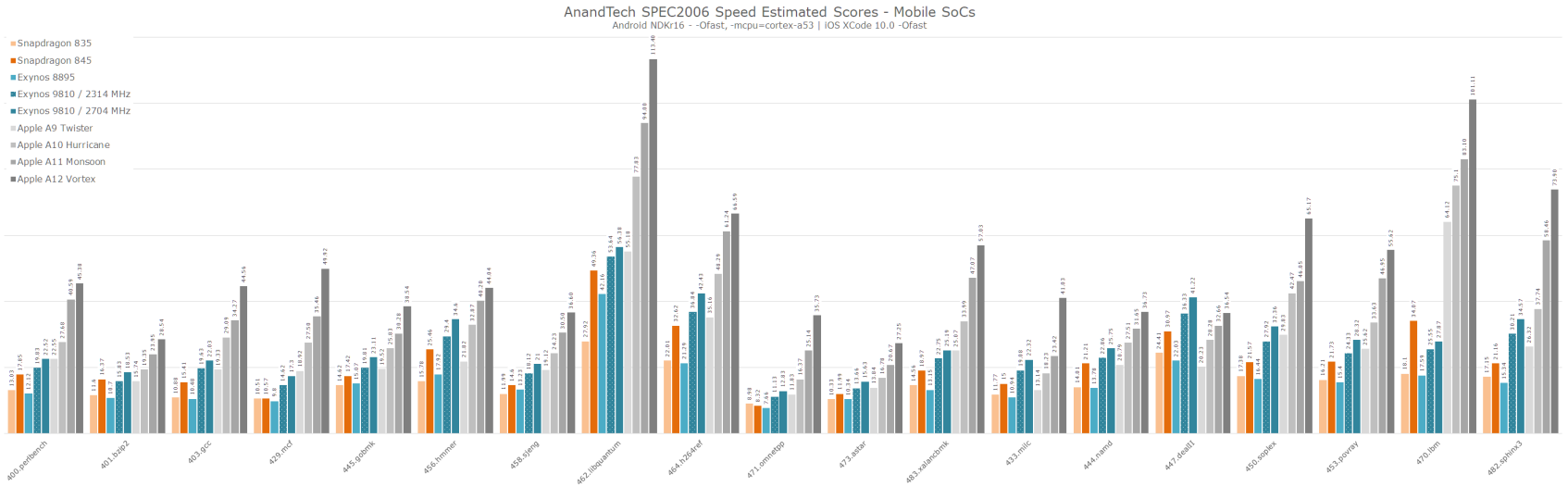
Assembly Language



C Language

- Some bugs are better understood when you look at them at an assembly language level

Optimization



- Low-level knowledge allows you to better optimize

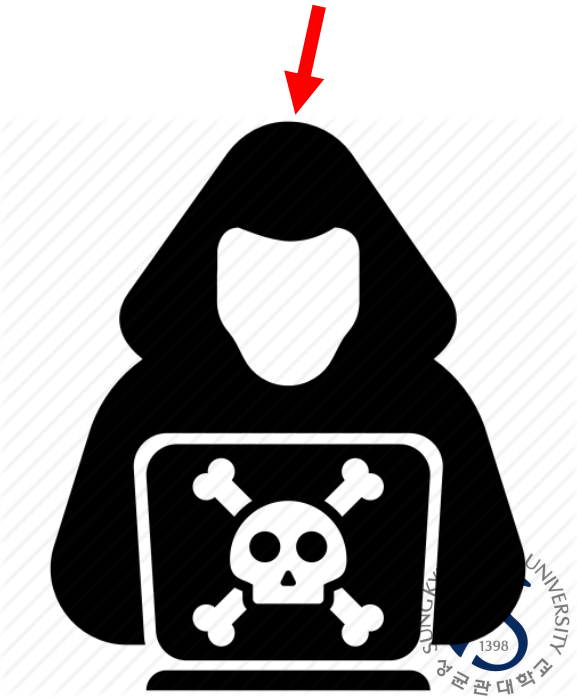
Systems Security Lab

SUNGKYUNKWAN UNIVERSITY 1398 성균관대학교

Malware Analysis = Debugging Skill Lvl. 100

- Hackers usually don't give you the source code for their malwares and exploits
- Malwares are often *obfuscated*
- *Reverse engineering* is an essential skill in software and systems security

Not a very nice guy ☺



Systems Security Lab

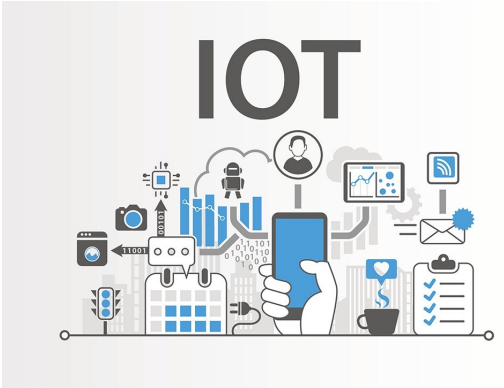
To be an (Ethical) Hacker

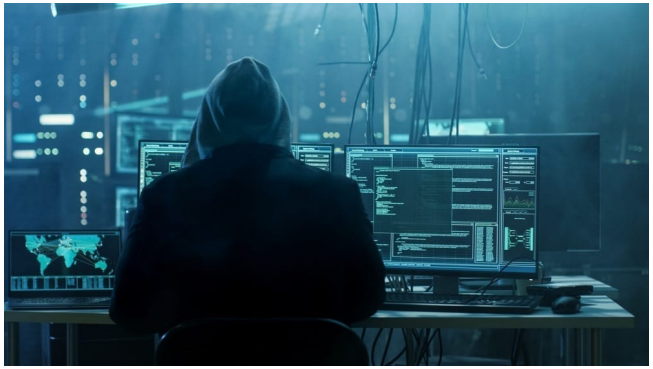
- You cannot be great at defense if you are not the attacker yourself
- Discovering software vulnerabilities and writing an exploit (attack code) takes very low-level knowledge and experience

Laying an Foundation

- What you learn in this course will help you a lot in the upcoming courses
- Compilers
- Computer Architecture
- Operating Systems
- Computer Security

Researchers and Industry Are Always in Need of Low-level Masters



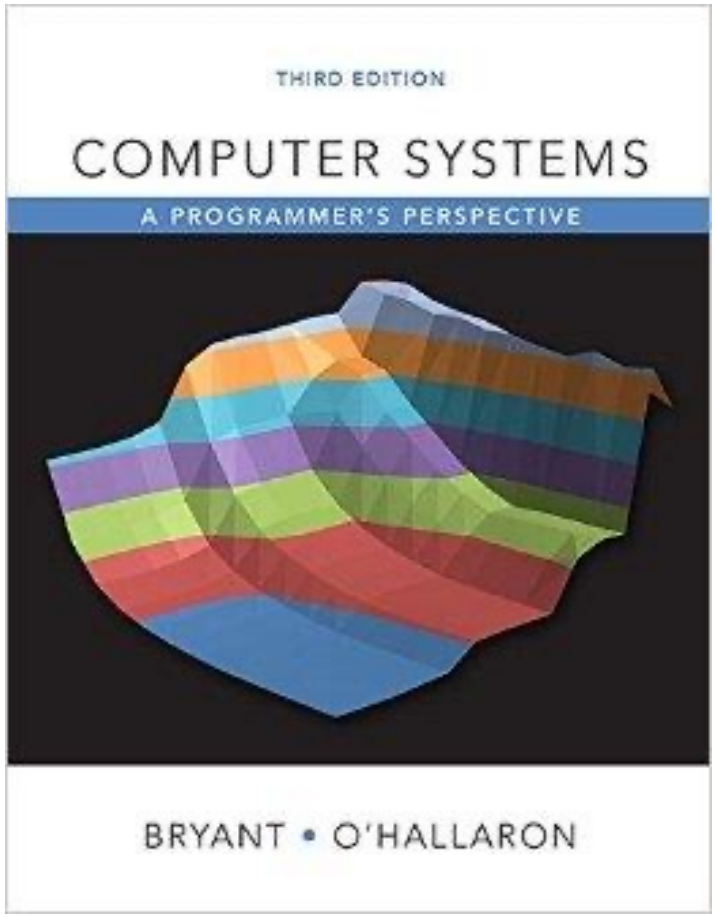


Most Importantly

- You will be a "Computer Scientist"

Syllabus Walkthrough

The Textbook



- Lecture materials will closely match the chapters in the book
- You NEED to READ assigned chapters

Systems Security Lab

SUNGKYUNKWAN UNIVERSITY 1398 성균관대학교

Lecture Format

- Tuesdays: Lecture (Recorded or Live)
- Thursdays: Interactive learning session
  - QnAs
  - Problem solving
  - Lab tutorial
- We will open a QnA page (TBA) so you can post your questions through Tuesday~Thursday

Syllabus Walkthrough: Grading

Grading

- HWs 10%
- Labs 30%
- Midterm Exam 30%
- Final Exam 30%

Syllabus Walkthrough: Attendance

- Yes. Attendance is required
- 1 Absence: -1% (Hope you don't end up with a 89.4)
- 2 Absences: -5%
- 3 Absences -10%
- 3+ Absences F

Syllabus Walkthrough: Academic Integrity

Any form of academic dishonesty is strictly prohibited in this course.

- Cheating on Exams
- Copying your friend's code
- etc …

Consequence: A grade of "0" will be given to the {Exam/Assignment} grade

Course Schedule (Tentative)

Week	Topic	Textbook Reading	Labs	HWs
Week01	How Computer Works / Intro to Linux	CH1		
Week02	Bits, Bytes, and Integers 1	CH2.1,		HW1 Out
Week03	Bits, Bytes, and Integers 2	CH2.2 ~ CH2.4	Lab1 Out	HW1 Due
Week04	Machine-Level Representation of Programs	CH3.1 ~ CH3.3	Lab2 Due	
Week05	Machine-Level Representation of Programs	CH3.4 ~ CH3.6	Lab2 Out	HW2 Out
Week06	Machine-Level Representation of Programs	CH3.7 ~ CH3.9	Lab2 Due	HW2 Due
Week07	Midterm			
Week08	Machine-Level Representation of Programs	CH3.10		
Week09	Software Attacks and Defenses		Lab3 Out	HW3 Out
Week10	Linking	CH7.1 ~ CH7.3		HW3 Due
Week12	Virtual Memory	CH9.1 ~ CH9.6	Lab3 Due	
Week13	System-Level I/O	CH10.1 ~ CH10.10		HW4 Due
Week14	Review			
Week15	Final			

Class poll

- Professor spoke too fast during this lecture
  - YES: [ ]
  - NO: [ ]

Class poll

- I know how to program in C
  - YES: [ ]
  - NO: [ ]

Class poll

- My confidence level in C is ..
  - Very confident: [ ]
  - Fairly confident: [ ]
  - Not so confident: [ ]
  - Hello world??: [ ]

Class poll

- My C development environment was
  - 구름 (Gureum): [ ]
  - VS Code: [ ]
  - Terminal editor + gcc: [ ]
  - ETC: [ ]

Class poll

- I have used Linux before
  - YES: [ ]
  - NO: [ ]

Class poll

- (If no Linux experience) I know what virtualization is and how to make a virtual machine
  - YES: [ ]
  - NO: [ ]

Class poll

- I do not have a x86-based computer (I only have a M1/M2 macbook)
  - YES: [ ]
  - NO: [ ]

Also…. Homework0: Surprise.c

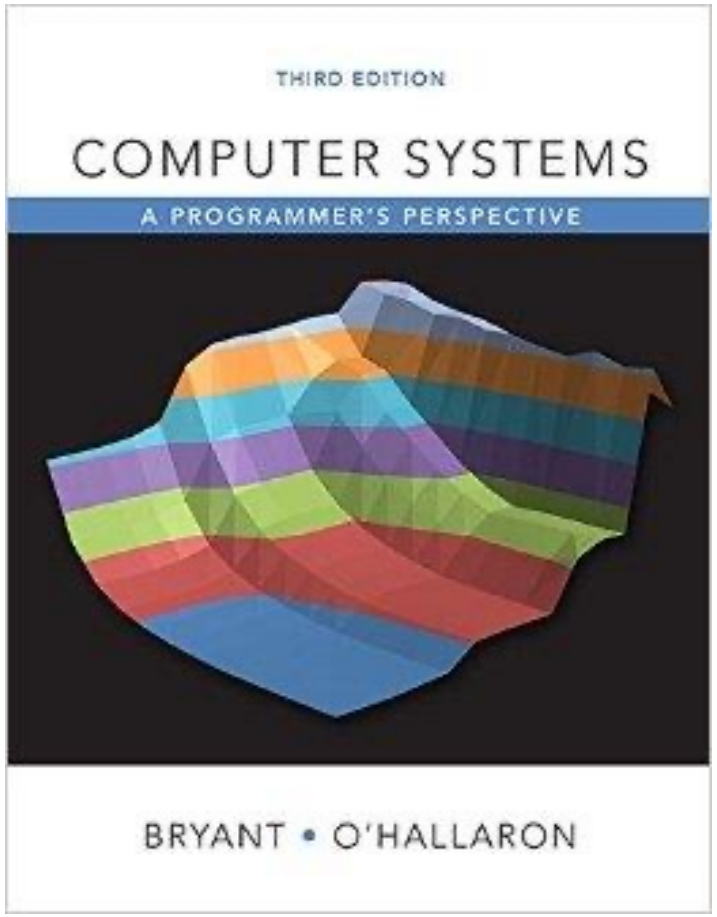
```c
#include <stdio.h>

int main(){
  // OK, nothing is wrong with this
  printf("-1 and  -2, which is bigger? >> %s\n", \
         (-1 > -2) ? "-1" : "-2");
  // OK, nothing is wrong with this
  printf("-1 and  2, which is bigger? >> %s\n", \
         (-1 > 2) ? "-1" : "2");
  // What???
  printf("-1 and  0, which is bigger? >> %s\n", \
         (-1 > 0U) ? "-1" : "0U");

  printf("-1 and  99999, which is bigger? >> %s\n", \
         (-1 > 99999U) ? "-1" : "99999U");
  return 0;
}
```

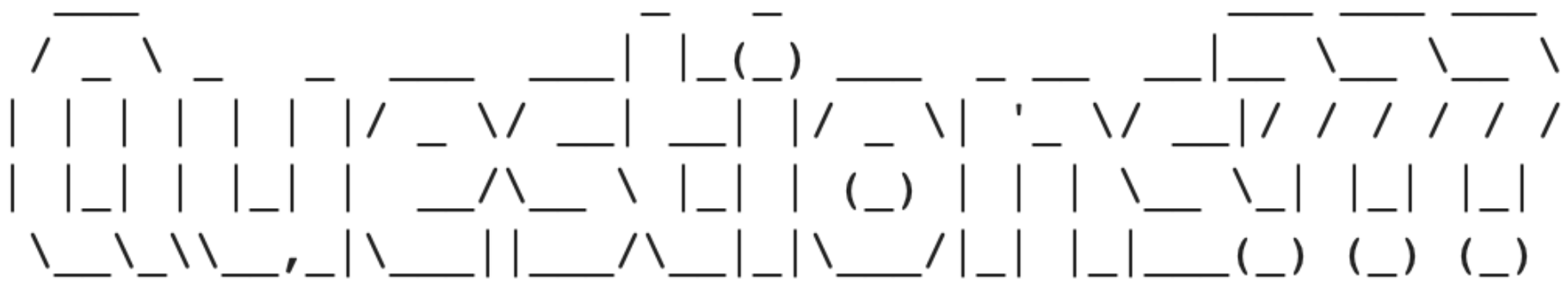
- Visit the discussion thread named "HW0 Discussion" on icampus
- You will find the above source code in the post
- Run it and see what happens (it will be weird). If you know why it behaves such way, post your answer on the thread

One more thing.... : Reading for the Week



- Chapter 1
- Chapter 2 sections [2.1, 2.2]

Any Questions?



- If you have any questions about the course
- Please feel free to drop me an email
- hojoon.lee@skku.edu
- Please title the email such that it begins with “[SWE3001]” to have a high priority in my mailbox.