ITSC 204 - Computer Architecture - Exploitation and Security

Course Description:
Through the detailed examination of ARM and X86 assembly language, the structure of CPUs, Trusted Platform modules, peripherals, memory subsystems and bootloaders will be examined. Exploits to each of these subsystems will be discussed. The labs will emphasize the use of subsystems using both C and assembler.

3 credits

Time Guidelines:
The standard instructional time for this course is 60 hours.

Effective Term:

Winter 2020/2021

Prerequisite(s):

- ITSC 202 - Secure Programming Essentials

Course Assessment:

Theory Quizzes	15%
Lab Quizzes	15%
Lab Reports	50%
Final Exam	20%
Total:	100%

SAIT Policies and Procedures:
For information on the SAIT Grading Scale, please visit policy AC 3.1.1 Grading Progression Procedure: http://www.sait.ca/Documents/About SAIT/Administration/Policies and Procedures/AC.3.1.1 Grading and Progression Procedure.pdf

For information on SAIT Academic Policies, please visit: www.sait.ca/about-sait/administration/policies-and-procedures/academic-student

Course Learning Outcome(s):
1. Examine the architecture of a general-purpose processor.

Objectives:

1.1 Review number systems.

1.2 Describe the building blocks of a processor.

1.3 Define Harvard and von Neumann architectures.

1.4 Discuss the advantages and disadvantages of Harvard and von Neumann architectures.

1.5 Define RISC and CISC architectures.

1.6 Discuss the advantages and disadvantages of RISC and CISC architectures.

1.7 Explain the interaction among the building blocks.

1.8 Differentiate byte, Word, Double Word, and Quad Word.

1.9 Perform processor-activity simulation (by hand).

2. Examine the x86 hardware architecture.

Objectives:

2.1 Describe general-purpose registers.

2.2 Describe segment registers.

2.3 Describe EFLAGS and IP registers.

2.4 Explain ALU functionality.

2.5 Explain cache.

2.6 Explain the memory management unit.

2.7 Describe processor control registers.

2.8 Describe data path.

2.9 Compare x64 and x86 architecture.

3. Examine the x86 software architecture.

Objectives:

3.1 Describe basic instruction format.

3.2 Describe addressing modes.

3.3 Describe data types.

3.4 Describe floating-point numbers and math.

3.5 Explain stack structure.

3.6 Explain calling conventions.

3.7 Explain interrupts and exceptions.

3.8 Explain procedure calls and frame pointer for x86 architecture.

3.9 Explain procedure calls and frame pointer for x64 architecture.

3.10 Describe instruction encoding.

4. Apply x86 programming tools.

Objectives:

4.1 Describe programming tools.

4.2 Install programming tools.

4.3 Review high-level languages.

4.4 Explain assembler, linker, dissembler and debugger.

4.5 Write, compile, execute simple assembly code.

4.6 Use debugger to single-step through simple assembly code.

5. Examine the ARM hardware architecture.

Objectives:

5.1 Describe registers.

5.2 Explain ALU functionality.

5.3 Explain memory protection unit.

5.4 Explain the memory management unit.

5.5 Describe processor control registers.

5.6 Describe data path.

5.7 Describe cache.

6. Examine System on Chip (SoC) hardware architecture.

Objectives:

6.1 Identify common peripherals.

6.2 Explain a subset of common peripherals.

6.3 Explain I/O pins and pin multiplexing.

7. Examine the ARM software architecture.

Objectives:

7.1 Describe ARM instructor format and Thumb-2.

7.2 Describe addressing modes.

7.3 Describe data types.

7.4 Describe floating-point numbers and math.

7.5 Explain stack structure.

7.6 Explain calling conventions.

7.7 Explain interrupts and exceptions.

7.8 Explain procedure calls and frame pointer.

7.9 Describe instruction encoding.

8. Apply ARM programming tools.

Objectives:

8.1 Describe programming tools.

8.2 Install programming tools.

8.3 Review high-level languages.

8.4 Explain assembler, linker, dissembler and debugger.

8.5 Write, compile, execute simple assembly code.

8.6 Use debugger to single-step through simple assembly code.

9. Analyze instruction sequences for reverse engineering.

Objectives:

9.1 Write, compile, execute assembly code for x86, x64, and ARM architectures.

9.2 Disassemble existing code samples.

9.3 Identify instruction sequences.
