# ITSC 203 - Offensive and Defensive Tool Construction

**Course Description:**
This course examines the structure, implementation and use of scripted and object oriented languages. The crafting of tools for both defensive and offensive conditions will be emphasized in the labs and assignments.

3 credits

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

**Effective Year**
2021/2022

**Prerequisite(s):**

- ITSC 202

**Course Assessment:**

| | |
|---|---|
| Tests | 41% |
| Lab Assignments | 45% |
| In Class Quizlet | 14% |
| 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

**Required Course Publication(s):**
TO BE ANNOUNCED IN CLASS.

**Course Learning Outcome(s):**

1. Use version control systems.

    Objectives:

        1.1 Review the functionality and behavior of Command Line Interface (CLI).

        1.2 Install text editor.

        1.3 Use text editor.

2. Apply basics of secure coding using scripting language.

    Objectives:

        2.1 Explain the purpose of Python.

        2.2 Use variables, expressions, and statements to write Python code.

        2.3 Explain functions.

        2.4 Explain conditionals and recursion.

        2.5 Explain functions with return value.

        2.6 Use strings to create text variables.

        2.7 Use lists and tuples to implement ordered structures.

        2.8 Use dictionaries to index data.

3. Create classes and objects.

    Objectives:

        3.1 Define a Python class.

        3.2 Create Python object.

        3.3 Use class functions.

        3.4 Define class methods.

        3.5 Explain inheritance.

4. Explain debugging theory.

    Objectives:

        4.1 Explain general-purpose CPU registers.

        4.2 Explain stack.

        4.3 Explain debug events.

        4.4 Explain soft breakpoints.

        4.5 Explain hardware breakpoints.

4.6 Explain memory access faults.

5. Build user mode debugger for Windows.

Objectives:

5.1 Write debugger code.

5.2 Attach debugged process to debugger.

5.3 Get CPU register state.

5.4 Handle debug events.

5.5 Use breakpoints.

6. Apply a debugger to exploit development.

Objectives:

6.1 Outline the basics of the Immunity debugger.

6.2 Find exploit-friendly instructions.

6.3 Describe bad-character filtering.

6.4 Defeat anti-debugging code in malware.

7. Apply process-observation techniques.

Objectives:

7.1 Use soft hooking with Python debugger.

7.2 Use hard hooking with Immunity Debugger.

7.3 Demonstrate Python to IDA Pro API.

8. Apply DLL and code injection.

Objectives:

8.1 Create a remote thread.

8.2 Inject code (e.g., malware) into DLL or shared library.

8.3 Evaluate file hiding techniques.

8.4 Write backdoor code.

8.5 Compile Python script into Windows executable.

9. Apply fuzzing techniques.

Objectives:

9.1 Explain the concept of “fuzzing”.

9.2 Distinguish bug classes.

9.3 Write code to implement a buffer overflows.

9.4 Use integer overflows.

9.5 Use format string attacks.

9.6 Create several types of fuzzers.

10. Implement basic networking utilities.

Objectives:

10.1 Create simple clients and servers.

10.2 Write Python code to replace Netcat.

10.3 Build a TCP proxy.

10.4 Create ssh server.

10.5 Apply ssh tunneling.

11. Analyze raw sockets using packet sniffers.

Objectives:

11.1 Build a UDP host discovery tool.

11.2 Contrast packet sniffing on Windows and Linux.

11.3 Decode IP and TCP layer packets.

11.4 Analyze TCP/IP traffic.

12. Hack the web servers and protocols.

Objectives:

12.1 Apply socket library urllib2.

12.2 Map open source web applications.

12.3 Evaluate brute-forcing directories and file locations.

12.4 Evaluate brute-forcing HTML form authentication.