ITSC 303 - Malware Analysis

Course Description:
This course provides students practical, hands on experiences with the analysis and neutralization of malware. Topics include: disassemblers, cloaking/obfuscation, malware design, packing techniques, infection vectors and polymorphism.

3 Credits

Time Guidelines:
The standard instructional time for this course is 90 hours.

Prerequisite(s):

- ITSC 204
- ITSC 205

Course Assessment:

Quizzes	40%
Lab Assignments	30%
Lab Final	30%
Total	100%

Other Course Information:
Learner Engagement:

In order to be successful, the learner is expected to be engaged in learning activities for a total of 9 to 12 learning hours per course per week, which includes both in-class and out-of-class time.

ICT Policies:

The School of Information and Communications Technologies (ICT) expects students to act professionally during their studies. These expectations are described in the school's Student Guidelines document page. Students should review the guideline regularly, as the content may change.

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):
Sikorski, M., & Honig, A. (2012). *Practical Malware Analysis: A Hands-on Guide To Dissecting Malicious Software* (1st ed.). No Starch Press. ISBN: 1593272901.

Course Learning Outcome(s):
1. Categorize the different malware types.

Objectives:

1.1 Provide description of a given code sample.

1.2 Describe the different types of malware.

1.3 Analyze scenario to identify the type of malware shown.

2. Analyze common malware static attributes in the portable executable format.

Objectives:

2.1 Outline key components of the portable executable format.

2.2 Use tools to extract static attribute information.

2.3 Identify abnormalities in the portable executable format.

3. Analyze static disassembly.

Objectives:

3.1 Analyze binaries using IDA Pro.

3.2 Apply disassembly techniques.

3.3 Interpret a binary’s capabilities based on extracted information.

4. Analyze decompiled source code.

Objectives:

4.1 Describe how decompilation is accomplished.

4.2 Use tools to decompile binaries.

4.3 Analyze source code for malicious activity.

5. Use tools to detect malicious files.

Objectives:

5.1 Explain how detection engines work.

5.2 Describe the tools available to detect malicious files.

5.3 Apply signatures against malicious files.

6. Analyze dynamic sample activity for malice.

Objectives

6.1 Describe basic dynamic analysis concepts.

6.2 Demonstrate debugging principles.

6.3 Analyze executing code for malicious activity.

7. Unpack malware samples.

Objectives:

7.1 Analyze code for possible unpacking routines.

7.2 Apply debugging to assist in malware unpacking.

7.3 Construct import tables for static analysis.

8. Assess malware samples to provide a description of how each achieves persistence, propagation or elevation of privilege.

Objectives:

8.1 Distinguish the various persistence techniques.

8.2 Evaluate an elevation of privilege.

8.3 Summarize the different propagation techniques and their associated motivations.

9. Incorporate the appropriate detection evasion techniques malware uses to avoid detection.

Objectives:

9.1 Create a sample that achieves process injection and execution environment detection.

9.2 Analyze non-native executable format samples.

9.3 Apply polymorphisms to previously detected sample to demonstrate detection evasion.

10. Evaluate malware command and control mechanisms and features.

Objectives:

10.1 Describe the purposes of controlling malware.

10.2 Analyze the techniques and common infrastructure that are used to control malware.

10.3 Distinguish encryption methods for a given sample.

10.4 Appraise when decryption is possible.

10.5 Write decryption tools for basic encryption schemes.

11. Implement malware detections for situations where static signature detections are inadequate.

Objectives:

11.1 Analyze sample using memory forensics tools.

11.2 Create in-memory detection of a sample.

11.3 Create indicator of compromise detections of a sample.

12. Evaluate malware samples that use advanced infection techniques and non-traditional targets.

Objectives:

12.1 Identify samples that target a variety of operating system targets.

12.2 Analyze kernel mode malware.

12.3 Develop a detection method for a boot loader malware sample.