# SE 421: Assignment #1

Due on August 27, 2018 at 12:00 PM (noon)

*Instructor: Ben Holland*

**Student Name:**

# Problem 1

A. (*1 point*) Compile and upload the `Exploit.class` file of CVE-2013-0422 (source code included with this assignment) to `virustotal.com`. How many antivirus products detect the unmodified proof of concept exploit?

B. (*3 points*) Either manually or with the assistance of a tool such as `https://github.com/Neo23x0/yarGen` develop a YARA rules file (see `https://github.com/virustotal/yara`) that could be traded with other malware researchers to detect the compiled code in `Exploit.class`. *Explain* how you devised your YARA rules or explain how the tool generated the resulting YARA rules file.

C. (*6 points*) Obfuscate the source code of the CVE-2013-0422 exploit in order to evade detection by antivirus products on the `virustotal.com`. Provide your obfuscated source, a virustotal link to the detection results of the compiled bytecode of the obfuscated source, and a summary of the techniques you applied and the effectiveness of each technique. Include a link to your obfuscated source file in your private GitHub assignment repository.

# Problem 2

A. (*2 points*) In your own words, summarize the attack described by Ken Thompson in the paper *Reflections of trusting trust* [1].

B. (*2 points*) Why is the described attack difficult to detect?

C. (*2 points*) Do you believe the attack described in the paper is plausible? What would be the challenges involved in implementing the attack? *Explain* your answer.

D. (*4 points*) How could the attack be countered through the use of two or more compiler implementations? Give an example of how you might check the open source GCC compiler (`https://gcc.gnu.org`) for a hidden backdoor.

# Problem 3

**(Extra Credit)**

In computability theory, the *recursion theorem* states that Turing machines can obtain their own description, which can then be used in further computation [2]. The ability of a Turing machine to implement self-referential algorithms means that any Turing complete language can be used to construct a program that outputs an exact copy of its source code when executed. These self-reproducing programs were later coined as quine programs by Douglas Hofstadter in his book Gödel, Escher, Bach: An Eternal Golden Braid [3]. A quine-relay extends quine computing to multiple levels or recursion. For example, a quine relay program A generates program $B$ which generates program $C$ and so on. An ouroboros (like the serpent eating its own tail) is a quine-relay that eventually produces the original starting program.

A. *(8 points)*: Your task is to write a ouroboros quine-relay program that writes to two output streams (stdout and stderr), while adhering to the following requirements.

   *Requirements:*

   - Your ouroboros program should be written in Java and contained in a single class file named `Ouroboros.java`
   - Your ouroboros program cannot receive any form of input, including reading from a file.
   - Each output program in the ouroboros should write the source code of the next program to be executed to stdout (`System.out`).
   - Each output program must include comments in the output source code.
   - Your first name and last name must be in a multi-line comment block following the author annotation (Example: `/* @author Ben Holland */`)
   - Each program in the ouroboros should output a single character to stderr (`System.err`).
   - Using the provided runner (*runner.sh*), the ouroboros should repeatedly output your first and last name followed by a newline to the file result.txt

   Include a link to your ouroboros program in your private GitHub assignment repository.

B. *(2 points)*: What is the relationship of your ouroboros program to a computer virus and what implications does the recursion theorem have on antivirus software? *Explain* your answer.

# References

[1] K. Thompson, "Reflections on trusting trust," *Communications of the ACM*, vol. 27, no. 8, pp. 761–763, 1984.

[2] M. Sipser, *Introduction to the Theory of Computation.* Thomson Course Technology Boston, 2006, vol. 2.

[3] D. R. Hofstadter, *Gödel, escher, bach.* Vintage Books New York, 1980.