Spring 2020 special topics course:

Computer Science 5980 (sec. 3)/8980 (sec. 2): Manual and Automated Binary Reverse Engineering

Description: The process of compiling turns source code, which is intended to be relatively easy to understand and modify, into a binary executable format that is efficient for a CPU to execute, but relatively hard to understand and modify. Nevertheless there are a number of situations where it is valuable to be able to understand code that is available only in binary form: when inter-operating with commercial software, when investigating security vulnerabilities, or when understanding malicious software (malware). This task is called "reverse engineering" of software because it is about recovering an understanding of software closer to what the original developers had. Some subtasks of binary reverse engineering are called disassembly and decompiling, corresponding to reversing the processes that assemblers and compilers do when software is compiled.

Binary reverse engineering is a more specialized skill than software development, and it has traditionally been done in a largely manual way. Tools for disassembly and decompiling are not as well developed as compilers, and the problem is more difficult because the compilation process loses information. However improved techniques for automated reverse engineering are an area of ongoing research, and some more usable open-source tools have recently started to become available. This course will introduce the area of reverse engineering of binary code from both manual and automated perspectives: we will start by looking at binary code mostly by hand, and then see which things can and can't be automated by open-source and research tools.

For examples the course will most often use 32 and 64-bit x86 binaries on Linux, but other kinds of binaries such as for Windows/x86, ARM, and Java bytecode will also come up. Students should already have a basic familiarity with C, assembly language, and low-level programming, as covered in CSci 2021.

Course Format: The course will combine hands-on skill building with a research seminar. For some class periods the instructor will lecture and give demonstrations of manual reverse engineering and the use of tools, and this will be supported by hands-on homework exercises in reverse engineering, such as creating C source code equivalent to a provided binary. Other class periods will be devoted to discussion of research papers, especially about automated techniques for binary reverse engineering. Papers will be presented by a mix of the instructor and students, and there will be written questions about comprehension and discussion in advance of the in-class coverage. There will be no exams, but students will complete a substantial final project in groups of up to three. The project should address a novel and generalizable problem, either by performing reverse engineering or by researching tools to improve reverse engineering capabilities. Near the end of the semester, groups will submit final reports, and give an in-class presentation.

The 5980 and 8980 versions of the course will meet together but have somewhat different requirements for out-of-class work. The 5980 course focuses more on reverse engineering practice, so it will have longer hands-on assignments, and the final project will require an electronic deliverable as well as a shorter report. The 8980 course focuses more on the research perspective, so 8980 students will do a larger number of in-class presentations, and thei final report will have the format of a research paper.

Instructor: Stephen McCamant (`mccamant@cs.umn.edu`, Keller Hall 4-225E)

Schedule: Mondays and Wednesdays, 9:45am-11am, 43 Rapson Hall

Prerequisites: CSci 2021 or consent of the instructor.