CS 138 – System Security Spring 2026

Project Overview

Instructor: Eleanor Birrell

Project Description

CS 138 students are expected to participate in a group project to build a software system that has non-trivial security functionality.

You have considerable flexibility in choosing what system to build. But because the course project is intended to provide an opportunity for you to exercise the material covered in CS 138, projects are acceptable only if the following "gold standard" security elements are necessary for the system to fulfill its mission:

Authentication. The system must authenticate its clients and/or allow its clients to authenticate the system. Clients may include human users as well as programs executing on machines. You must implement reasonable means of authentication, which may include passwords, user registration, generation of secrets, distribution of keys, etc.

Authorization. The system must enforce some non-trivial authorization policy to control some subset of its operation. Some systems will require DAC; others will require MAC. Implement an access control mechanism that is appropriate and natural for your system's functionality and expected scale.

Audit. The system must provide infrastructure for audit or other means of establishing accountability for actions. The security of that infrastructure must itself be ensured.

Projects must also intrinsically require information security:

Confidentiality and Integrity. The system must involve information that resides in long-term storage or that is transmitted over a network. The system's mission must require that information to be kept secret and/or be protected from corruption.

The list of essential security elements above defines only a subset of the security functionality your project will implement. What is the rest of that functionality? Answering that question will be the primary task of Milestone 1. If you need some ideas, a few example projects are listed below.

Example Projects

Here are sketches for a few example systems that could involve all of the above essential elements. Each sketch has important elements missing, as befits a sketch. Nevertheless, each could be refined into an acceptable course project, and you should feel free to do so. But you are welcome and encouraged to invent your own project idea!

Secure Anonymous Communication. This system enables users to communicate with each other secretly, accurately, and anonymously. Users can specify what information other users may learn about them and their communications.

Electronic Voting System. This system enables users to privately express their preferences about some issue. The system produces a verifiably correct aggregate of all the users' preferences.

Grade Management System. This system allows student grades to be stored by course staff, which may include TAs and professors, and to be retrieved by students. Grade information is stored in a back-end file system.

Multi-player Game Service. This system might implement a game, where clients are players; or it might implement a virtual world, where clients control participants. There might or might not be a back-end server.

Password Manager. This system allows users to create and store usernames and passwords for other systems. Users could manage their passwords across different devices.

Implementation

Your system should be designed for public distribution. It should run on Ubuntu and/or OS X, and all milestones should be accompanied by a README.txt file containing installation instructions and, as necessary, installation scripts. If users cannot run your system, they can't use it. If I cannot run your system, I can't grade it.

When building a system in industry, it is generally a good idea to extend existing components rather than build your own. For example, there are many third-party systems and tools available for building web services. But using these tools in CS 138 would preclude activities the project is supposed to cover. This is because, when you use a third-party tool, you must (i) accept somebody else's choices about what is useful security functionality and (ii) accept somebody else's assurance argument. I therefore impose the following rules about using code or systems written by others:

- Standard language libraries (i.e., those part of the distribution) may be used. This should include various cryptographic routines, which you shouldn't be writing yourself anyway.
- GUI builders that are part of, or plug into, Eclipse or VS Code may be used.
- Operating systems installed on the CS department lab machines may be used. This includes the networking infrastructures and file systems native to those operating systems.
- Database management systems that function as local, library-level services may be used. This includes the Java interfaces to Berkeley DB and SQLite. However, databases that run as separate servers and are accessed over the network may not be used. You should be able to understand and justify any security assumptions adopted by the system(s) you use.
- ChatGPT may be used to generate code, test cases, documentation, or other parts of your project, but it may not be used interactively at runtime. If you use ChatGPT or similar tools in any way, you must explicitly discuss how you used it as well as why we should trust the result in your assurance documentation.
- You may use Google, StackOverflow, and other Internet resources for debugging.
- Existing web browsers, web servers, or any other web services infrastructure may *not* be used. These technologies make too many security decisions for you.
- Third-party libraries or other large chunks of code written by third parties generally not allowed.
- Of course, the above rules are incomplete. If you believe it makes sense to incorporate other third-party code into your project, you must (i) obtain written approval from me in advance, (ii) confirm that the license of that code is amenable, and (iii) clearly acknowledge the source of that code in your documents and demos.
- In some cases, exceptions to the above rules may be made. *If you would like an exception to be made for your group, you must get permission from me in advance!*

Groups

Part of the purpose of this course is to give you experience in building software, including engineering its security, as a member of a development team. Why? Because...

- Working in a group offers you the powerful tool of discussing ideas with others.
- Working in a group affords the opportunity for parallel development activities and specialized expertise.

- Working in a group helps hone skills needed to be effective in the workplace (where groups are the norm) and impresses potential employers.
- Working in a group enables you to complete a more ambitious course project.

Group size. Your group must start with 3-4 members. If through attrition your group size becomes too small, personnel may be re-assigned by "Management" (i.e., me) from another group.

Milestones

The majority of the work—and the grade—for this course will come from the course project. This project is broken down into six phases. For each milestone, you will submit a written report and a working implementation of the project (up through that milestone). For Milestones 2-4 you will also demo your project. You will present your completed project in class during the last week of classes (May 4 - May 6, 20206).

Milestone 0: Groups. Form a group of 3-4 students and submit a one-paragraph project idea. Due: February 10, 2026.
Milestone 1: Requirements. Describe the project you plan to implement this semester. You will also provide a detailed list of the security goals you plan (or hope) to implement. Due: February 17, 2026.
Milestone 2: Alpha Release. Implement some core functionality for your system. Due: March 9, 2026.
Milestone 3: Authentication. Implement the authentication-related security goals for your project. Due: April 6, 2026.
Milestone 4: Authorization. Implement the authorization security goals for your project. Due: April 27, 2026.
Milestone 5: Final Project. Implement the audit-related security goals for your project, along with all remaining functionality. Due: May 6, 2026.

Grading

Your project grade will comprise 60% of your course grade in CS 138. Grades will be assigned as follows:

- Milestone 0 (1%): You will receive credit if you submit your proposal on time. If this document seems to be taking your group in the wrong direction, I may invite you to resubmit it.
- Milestone 1 (4%): Your requirements document will be graded for completeness and quality of your documentation. If this document seems incomplete or seems to

be taking your group in the wrong direction, I may invite you to resubmit it with substantial improvements.

- Milestones 2-5 (5% each): You will be evaluated on the completeness and quality of your documentation, the completeness and security of your implementation, and the ease with which I can get your project running.
- Final Project (25%): You will be evaluated on the completeness and quality of your documentation, the completeness and security of your implementation, and the ease with which I can get your project running. This grade will also be influenced by the originality, difficulty, and non-artificiality of your project.
- Final Presentation (10%): Your final presentation will be assigned a letter grade that reflects the quality of the presentation, demo, and Q & A about the security aspects of your project.