CS 576 Secure Systems
CS/SES
Fall 2015

PROJECT DESCRIPTION

The goal of the project is to build a system that goes beyond toy applications that you may have developed for other courses.. Teams of students will build, evaluate, and document a complex software system.

This year the teams will build a password manager. The project is based on project given on the corresponding course at Cornell University.[1] The final product should be usable and secure enough that the students would consider to use in their daily routine to store sensitive passwords, like their bank credentials.

FUNCTIONAL REQUIREMENTS

Every password manager needs to implement at least the following features:

- Users can create and store usernames and passwords for multiple services.
- The credentials stored in the password manager can be shared from multiple machines (e.g., multiple laptops/desktops, smartphones, etc.).

The second feature indicates that you need to implement the system with a client and server, that is, a client for the user to interact with the system and a server that facilitates the sharing of credentials among different devices. The client should run locally, while the server should run remotely. The client should provide basic functionality even if the server is not reachable, while the server can be implemented on the cloud or on the Linux lab. A graphical interface is not required, however, it may facilitate demoing the system.

Note that beyond these basic requirements the project is intentionally left vague to encourage creativity. Create the best password manager you can without sacrificing security. Think of what kind of features you would like to use in such a product. In all cases remember that your primary goal is to develop a working and usable system and, second, to improve it with novelties and features.

SECURITY REQUIREMENTS

Your system has to satisfy certain security requirements. Based on the design you adopt and the features that you decide to implement you may have varying security concerns that you need to address. However, the following three (high-level) requirements must be satisfied by all projects:

[1] http://www.cs.cornell.edu/courses/cs5430/2015sp/project.php

- **Authentication.** The system must authenticate its clients using reasonable means of authentication, which may include passwords, user registration, generation of secrets, distribution of keys, biometrics, etc.
- **Confidentiality and Integrity.** The system includes information that resides in long-term storage locally and remotely and that is transmitted over a network. You must ensure that the information is kept secret and is protected from corruption.

Note that it is crucial that the security of the system does not rely on the client software or the obscurity of the protocols and server operations. Assume that an adversary can reverse engineer your software and protocols.

IMPLEMENTATION

You must use C/C++ or Java to implement your system. Note that Java by design prevents buffer overflows and other vulnerabilities, but it does require a JVM to run. Consider the advantages and disadvantages of each programming language before choosing one. Note that we expect the projects compile and run on the Linux lab. While you can use your laptops to demonstrate your work, if a project does not build on the linux lab, we will consider it is not building/working. Zero credit is awarded for systems that do not build.

While in real life, it is generally a good idea to reuse existing components, for the goals of this course you will need to develop most of the components of your system. You are welcome to use tools that check your software for bugs, etc., but you should develop all the code in the project with some exceptions. These include:

- Standard libraries (e.g., standard Java packages and STL for C++)
- Cryptographic libraries
- GUI builders
- Data structure implementations

You are not permitted to use web service frameworks (including browsers and web servers). If you build a component that runs on a mobile platform (e.g., Android or iOS), the standard libraries and frameworks on that platform may be used. Database management systems may be used, but are not necessary! Do not make your system more complex than it needs to be. In some cases, it may make sense to incorporate other third-party code into your project. To do this, (i) you must have the instructor's prior approval, (ii) the license of that code must be amenable, and (iii) your project documents and presentations must clearly acknowledge the source of that code.

DEMO

You should be able to demonstrate your system throughout in each of the phases of the project. Take this project as an opportunity to seriously impress your peers and potentially your future employers.

MILESTONES

PROPOSAL

The proposal should include:

- A working name for your system.
- The members of your team.
- Description of the proposed system.

The whole proposal should be a one page on a 10 or 11pt font. It should include:

- A very short statement of the vision/ idea for your system.
- A bulleted list of the key features of the system.
- A bulleted list of the essential security elements and a one sentence description of how your system will incorporate each element.
- Provide a narrative description of the system you intend to build. Go into enough detail in the body that, if your proposal were given to another team, and that team were never allowed to talk to you, they could still build more or less the system that you have in mind.

DESIGN

This is going to be the design document for your system. Sketch your requirements, plan of implementation (i.e., what goes into the alpha, beta, and final releases of your system), and your evaluation plan. This documents should be a maximum of three pages using a 10 or 11pt font.

Dedicate some space to describe the system, as you did in the proposal, highlighting modifications to the original proposal. You should also describe the functional and security requirements of your system. Functional requirements should include a list of the features you plan to implement, how important they are for the purpose of the system, and an example action where the feature is used. To designate the importance of a feature use the following scale:

- **(A)** Must have.
- **(B)** Should have, if at all possible.
- **(C)** Would be nice to have, but not necessary.

The design document should also include a **threat model**. This will identify the threats of concern to your system. What kinds of attackers will your system defend against? What are their motivations, resources, and capabilities? Don't just list vague, generic threats. Make them specific to your system and its functionality. In addition, if there are any threats not handled, identify them along with components that they are assumed to be trusted. For instance, you may assume that the hardware is not malicious.

After going over the design document, we will provide feedback that the teams should attempt to incorporate in their design and future versions of the document.

ALPHA SYSTEM

You are free to decide the order in which you will implement the features of your system. However, we expect incremental improvements between the Alpha, Beta, and Final versions of your system. Each version of your system **must** be accompanied with an updated version of your design document highlighting any changes you had to do to the original design and description of the implementation of the features it includes. While the design document can change during this process, you should not remove features or sacrifice security light-heartedly. We do expect everything described in the design document to be delivered. You will have to justify any deviation from your plan, which may come at a cost to your grade.

For your own good, you are **required** to keep your project in a source control system. There are many free possibilities including google code, github, sourceforce, bitbucket, etc., that provide code hosting,

bug tracking, wikis, etc. Under no circumstances should you resort to emailing your entire source code between team members.

The alpha system serves as a chance to flag projects that are not in the right track or are falling behind. While a demo is not required at this phase, a demo will indicate that the team is in the right path. Submit a snapshot of your code through Canvas including the updated design document. The alpha version of the system will be reviewed in the team's presence the week following the deadline. All team members must be present for this examination and they are required to answer questions from the instructor and TA. The review will include source code and a demonstration, if available, of implemented functionality will be also done.

BETA SYSTEM

The beta system should be a significantly improved version of the alpha system. Submit a snapshot of your code through Canvas including the updated design document. The beta version of the system will be reviewed similarly to the alpha version, but it needs to include a demo of the system from the team, and may include live testing from the instructor and TA, during which it will be tested using various inputs, etc.

FINAL SYSTEM

The final system should implement all the features promised in the design document. Submit a snapshot of your code through Canvas including the final version of the design document. The final version will be reviewed similarly to the beta version.

CHALLENGE FACTOR

Extra challenge is rewarded and, as such, it is part of the grade. Most project will receive a challenge factor of 0%, however, by making your project original, providing unique functionality, and going the extra mile to keep the user's passwords secure will increase the challenge factor. Less trust assumptions and stronger adversaries in the thread model are one way to increase your challenge factor.

Keep in mind that projects with a non-zero challenge factor are going to be much more difficult to build. For example, they may involve learning and implementing new cryptographic protocols from the research literature. If you want a good grade in the course, you're better off concentrating on building a very well-secured system that makes stronger trust assumptions.

Another way to increase your challenge factor is to build a system that is new and exciting. *New* means that your system should have some aspect that is novel, rather than just replicating some existing system. In the ideal case, exciting means you might become rich and famous.

PRESENTATION AND DEMO

During the last lecture teams will do a presentation of their work and demo their system to their classmates.

PEER REVIEW

Along with submitting the final version of your project, you need to complete a peer review form for each of your group **including yourself**. Your comments will be treated as confidential and will not be shared with other members of your group.

The form will you to assign each team member a letter grade (A through F) in each of the following areas:

- Participation: this member consistently and punctually showed up to meetings, attended milestones review meeting, work sessions, and contributed within the group
- Focus: the teammate was met deadline and took initiative
- Quantity of Work: the amount of work contributed by the teammate
- Quality of Work: the quality of the teammate's work
- Attitude: the teammate contributed to a positive and motivated atmosphere, it was a pleasure to work with them
- Overall Grade: based on all the above items

GRADING

We will evaluate your design document using the following criteria:

- Clarity, organization, and use of English
- The purpose of the system is clearly stated
- The functional requirements are sensible for the purpose of the system, complete, and clearly described
- The threat model is clear and sensible
- Security goals are clearly related to confidentiality, integrity, or availability.

The design and implementation of your system, including your code, will also be evaluated. We will look for code that is secure, readable, simple, well documented, and includes testing. The originality and difficulty of your project will also influence the evaluation.

RUBRIC

Your grade will be based on:

Proposal	(5%)
Design	(20%)
Alpha System	(10%)
Beta System	(10%)
Final system	(35%)
Peer review	(5%)
Presentation	(5%)
Challenge factor	(10%)