Computer project
SSD & WS

R. Absil

Academic year 2022 - 2023

This document describes the features of the computer project you have to implement for the course. You will find here details about the requirements of your applications, along with the constraints to respect and the submission procedure.

Basically, the idea is to build up a secured online agenda. The project can be implemented by groups of maximum two students, and has to be submitted on 8 January 2023 at 11:59pm.

Contents

1 Introduction

The goal of this project is to implement a small system allowing users to *securely* handle agendas, under a client / server architecture. A lot of freedom is left to your discretion regarding the security policy and actual data storage. Considering the main aspect of this project is security, appropriate techniques have to be used, whether they have been covered in class or not.

Hence, although you are free to choose protocols and languages that you find appropriate, you are responsible for these choices. That is, should you favour some technique over another, and if it figures the choice you made is not relevant regarding security, you will be penalised.

2 System Characteristics

The architecture to use for your system is "client/server". Consequently, there are only two types of actors: a server, and a set of clients driven by users.

From a general point of view, the server allows:

- new users to register to the system;
- users to log in the system;
- authenticated users to set-up events;
- authenticated users to see / delete / edit events;
- authenticated users to "invite" other users to their events.

Information described in the next part of the document specifying these features is deliberately high-level: it is your job to detect the key points to secure in your project, and how to do it. For that purpose, you are allowed to deviate from what is written here in order to strengthen security.

Server

The server is *not* a trusted entity: you do not know its set of public keys. Consequently, you have to provide a mechanism to "securely" transfer it and checking its ownership.

On the other hand, it is your job to decide what security to implement regarding the transmission of data to the server, as well as regarding the storage of data on the server.

Clients and users

The client allows users registered in the server to authenticate themselves in order to use the features described above. Consequently, a user has the following attributes:

- some authentication material (passwords, cryptographic keys, and/or whatever you find fitting);
- a set of information mandatory for its secure communication with the server, such as keys. You are free to handle the generation of that information when a new client registers in any way you find satisfactory,
- a list of *contacts* (other users) that he might invite to his events.

Note that users must be able to change their credentials securely, and that it is expected that users can log in from different clients (that is, different devices).

Event

By event, we mean a small piece of data with the following characteristics:

- a name;
- the user who created it;
- the list of participants (other users) to that event;
- the beginning of the event, and its end (both time points, such as dates);
- a location (a small text);
- a description (a small text).

Note that all the above attributes are considered sensitive information. Furthermore, note that in no way you may consider that the server is uncompromised regarding data storage. That is, if the server is compromised[1], the confidentiality of the files stored to the server is never put in jeopardy.

Agenda

Each user has an agenda, that is, simply a list of all the events he created, or that he has been invited to. The agenda of a user is considered sensitive information.

3 Features

In each of the following protocols, data exchange must be secured at best (according to the relevance of the protection provided). The same remark can be applied to the storage of data resulting from an exchange. Obviously, if events are stored ciphered, when a user wants to see them, the system has to decipher the considered events.

The type and level of security to use are left to your discretion. Consequently, it is recommended to implement more measures than those dedicated to integrity, confidentiality and authenticity, such as denial of service, dictionary attacks and injections.

Again, note that you are allowed to deviate from the protocols described here, as long as these changes are motivated by security. However, you are responsible for these choices: should you change a protocol by another less secured, you will be penalised.

[1] For instance, if an administrator maliciously updates the server so that he recovers every password sent to it.

User registration, authentification and revocation

When a new user wants to register to the server, after the authenticity of the server has been verified, credentials have to be generated for the user. The form of these credentials (passwords, keys, etc.) is left to your discretion.

After this step, the user can log in the system, by giving its credentials.

Note that users can log in from different devices, and change their credentials.

Adding / deleting a contact

A user can request to add some other user (a *contact*) to his list of contacts. If the target contact exists, the server notifies him. The contact then decides to deny or accept the request, in which case he notifies the server. The server then notifies both users that they are now in each other's contact list.

Creating, editing and deleting an event from the server

A user can set up an event, which will be added to his own agenda. This user is considered the *creator* of that specific event.

In the same way, a user can edit an event he created, or delete it permanently. Note that in no way can a user edit or delete events that he didn't create.

Invitation to an event

A user can *invite* a user from his list of contact to an event he created. By inviting, we don't mean "transmitting" the event: event invitation is simply used as a very basic one-to-many permission check.

Checking an agenda

A user can see his agenda, that is,

1. see the list of all events he created,
2. see the list of all events he didn't created but has been invited to (by a contact in his list of contacts).

4 Submission

Projects have to be implemented in pairs (groups of 2 students), and submitted with the help of a GitLab[2] repository[3]. For that purpose, send an email to your teacher on 18 December 2022 at 11:59pm at the latest with the ssh URL[4] to your repository[5], and the name and matricule of your group members[6].

You have to submit your work on 8 January 2023 at 11:59pm at the latest. The minimal requirements for submitted projects are as follows:

- projects have to be submitted on time[7],
- projects have to provide a `README` file
  - mentioning the name and matricule of your group members,
  - explaining how to build your project (we recommend here to either provide a makefile, or a shell script to install missing dependencies, compile the project and run relevant scripts),
  - explaining how to use your project (for example, "to launch the server, type the following command in a shell").

Projects failing to meet these requirements will not be graded (that is, they will get 0/20). In particular, projects that do not compile according to your *exact* instructions will not be graded. Furthermore, note that we shall in *no way* build or run your projects in an IDE.

As usual, you must provide a modular code, easy to maintain, etc. Moreover,

- any submitted code must be duly documented and provide needed configuration files in order to produce the developer documentation.
- *only* security features are graded for this project,
- any submission must include a report under PDF format detailing your choices regarding security. You are advised to follow the guidelines presented by H. Mélot [1] for your redaction, and to answer to all questions listed in the check-list attached to this document.

References

[1] H. Mélot. Élments de rédaction scientifique en informatique. http://informatique.umons.ac.be/algo/redacSci.pdf - Consulté le November 28, 2022.

[2] That is, not a github one.
[3] Create the repository yourself, add your teacher (rabsil) as maintainer.
[4] A git ssh URL looks like `git@theserver:username/projectname.git` .
[5] The automatic email notification is *not* enough.
[6] I love footnotes.
[7] I retrieve your projects *on time* with the command `git pull origin main`. In no way shall I grade or even look at another branch, or pull anything past the deadline.