Team Members: Brandon Peh (bcp39), Louise Lee (zl245), Ruixin Ng (rn279), Wang Zilong (zw243). Emails: (netid)@cornell.edu

System Backlog

Functional Requirements	**Completed?**
As an owner, I can upload new files to the system.	Frontend - yes
As an owner, I can create new folders in the system.	Frontend - yes
As an owner, I can upload new folders in the system.	No
As an editor, I can upload new files to folders that are shared with me.	Frontend - yes
As an editor, I can overwrite files that are shared with me.	Frontend - no
As an editor, I can share folders and files that I have uploaded with other users, giving them either read/write access.	Frontend - yes
As an editor, I can remove the read/write privileges of other users.	Frontend - no
As an editor, I can rename folders and files that are shared with me.	Frontend - yes
As an editor, I can rollback my files to a previous version up to the last 5 days.	No
As a viewer, I can download files that I have access to view (either uploaded by me or shared with me by other users).	Frontend - yes
As a viewer, I can view the event log of a folder or file that is shared with me.	Frontend - no
As a user, I can sign up for a user account.	Frontend - yes
As a user/admin, I can change my password.	Frontend - yes
As a user/admin, I can require 2-factor authentication via email in order to log into my account.	No

As an admin, I can monitor the the logs of user account activity as well as file activity.	No
As an admin, I can restrict the total size of files of all users.	No
As an admin, I can add or remove users from my system.	No

Threat Model

Criminals

- Motivation: blackmail, financial gains from sale of sensitive information
- Capabilities: probably not extensive financial and computational powers, social engineering, technical skills to intercept network packages, network access to server, no physical access to servers or computers of the user

Hackers

- Motivation: technical challenge, desire to know about files they do not have access to
- Capabilities: technical skills, network access to server, no physical access to servers or computers of the user

Unhappy viewers

- Motivation: curiosity, desire to gain access to files they do not have the privileges to, desire to cause trouble by editing files they do not have access to edit
- Capabilities: not much impetus, view access to the files, network access to server, no physical access to servers or computers of editors, social engineering (could be friends of users with edit rights)

We assume that admins are trusted and will not perform any insider malfeasance.
We will use the Dolev-Yao model for attackers of our network.

System Purpose

Aim: To provide a secure file sharing system for users.

Users who are logged into the system can create folders and upload files to a private repository. They should be guaranteed that their files are not viewed or modified by others without authorization. Users can share files with other users and choose if the other users are only allowed to download the file, or if they are also given privileges to overwrite and change the permissions of the file. All actions on files and folders are logged in order to ensure accountability.

Users with viewing permissions can view the log associated with a file or folder. Changing the permissions on a folder sets all subfiles and subfolders to have the same permissions along

with any other user's permissions set earlier for particular subfiles and subfolders. A concrete example of this: User A has been granted edit rights to file F that is in folder G. When permissions are added to folder G, file F should reflect these new permissions but also retain user A's edit rights (these are *additional*).

System administrators are allowed to a log of all user account actions (creation of accounts, change of passwords and account information). They can also delete users and limit the total file size of users.

All users have to be logged in before performing any actions. We will use a 2FA system (the second factor being email) to log in.

Security Goals

1. The system shall prevent the modification of files by unauthorized principals. [I]
2. The system shall prevent the deletion of files by unauthorized principals. [A]
3. The system shall prevent unauthorized principals from viewing the name and content of the files stored on the server. [C]
4. The system shall detect changes made to a folder by a user (creation, deletion) [I]
5. The system shall detect changes made to a file by a user (uploading, deletion, overwriting, renaming) [I]
6. The system shall prevent unauthorized principals from removing the user permissions of a file/folder. [A]
7. The system shall prevent privilege escalation by unauthorized principals. [I]
8. The system shall detect the change of privileges made to all files and folders. [I]
9. The system shall prevent excessive usage of the server's storage capacity. [A]
10. The system shall prevent unauthorized viewing of the previous versions of files. [C]
11. The system shall prevent unauthorized modifications of the previous versions of files. [I]
12. The system shall prevent unauthorized access to user accounts. [C]
13. The system shall prevent unauthorized viewing of password files. [C]
14. The system shall prevent unauthorized modification of password files. [I]
15. The system shall detect changes made to users' password. [I]
16. The system shall prevent unauthorized deletion of password files. [I]
17. The system shall prevent unauthorized viewing of the event logs of changes made to files. [C]
18. The system shall prevent unauthorized modifications of the event logs of changes made to files. [I]
19. The system shall prevent the unauthorized viewing of the event logs of users. [C]
20. The system shall prevent the unauthorized modification of the event logs of users. [C]

Essential Security Elements

Authentication:

Authentication allows us to identify who is allowed to access our system. It is the first barrier in preventing the threats listed above from accessing our system and compromising the confidentiality and integrity of our file sharing system. Only legitimate users of the system are

allowed to access the system and we would like to keep any threats from accessing the system in the first place. Authentication is therefore essential in providing a first line of defense against any intruders.

Furthermore, authentication also allows us to bind principals of our system to the actions that they perform. Each user of the system is associated with a user ID, and whenever an action is performed by a user, this can be recorded in an event log. Any malicious actions by insiders or a hacked account can be traced with the event log. Thus, authentication can hold users accountable for their actions.

Authorization:
Authorization is another critical aspect of our system because there are various types of users in our system. We want to differentiate between the different kinds of actions that different users can perform, so a mechanism is needed to determine what actions can be performed. A user can either intentionally or unintentionally cause harm to the system if given too many permissions.In our system. Authorization is crucial because we do not want users to have access to assets that they should not view or modify as we must uphold the principle of least privilege. Appropriate authorization must also be given to legitimate users to ensure the availability of files to these users.

Audit:
As mentioned in authentication, we require audit mechanisms such as event logs that can record and review actions. This allows us to hold users accountable for the actions that they perform in the system. Malicious users who abuse the permissions given to them will have their account suspended. This audit mechanism is important in deterring potential attackers as well as ensuring that users of the system act responsibly. Furthermore, apart from deterrence, monitoring the event logs can allow the admin to spot any suspicious actions that have been performed. This allows the admin to take the appropriate measures such as performing a system rollback in order to revert the system back to a safe state. These reasons demonstrate the importance of the audit mechanism in our system.

Confidentiality:
Confidentiality is key in our system as well. Since our system implements file sharing, users who upload personal files would naturally want to keep their files secret and secure. This makes confidentiality important because we would want to protect these files (assets) from any unauthorized disclosure. A key feature of our system is that a user is able to share files to a certain group of users. However, at the same time, the user who shares the files might not want other unauthorized viewers to see the file. For example, a team that has their code stored using this system will not want other non-team members to see their code. Given that some users want to share sensitive information with a restricted group of people, confidentiality is therefore an important security element that we have to implement for our system.

Integrity:

Integrity is also integral to our system. Users who do not have the right to edit files should not be able to make changes to files - owners should be assured that only editors have the right to make changes to files. Aside from privilege restrictions, our system also detects all changes and records it in an event log. This will provide some deterrence through accountability.