Secure Sharing Project

12-4-2017

CS6238

Logan Carpenter

Juahui Li

**Overview:**

This document will discuss the mechanism used to implement secure file sharing in a distributed systems environment. This system is a client-server operation where the server oversees all file sharing between its users to insure confidentiality and integrity. Each client must be registered and session with the server before any requests can be made that require file access.

**Libraries:**

* Flask
* Shelve
* Hashlib
* Base64
* Openssl
* Crypto
* Binacii
* Datetime
* Request
* Json
* Uuid
* M2Crypto

**SSL Environment:**

* The directory sub-ca must be generated using the tlsauth library, this is where the certificate authority lives
* The server has the cert and keys generated and issued by the certificate authority in its directory
* Client must point to the certificate authority in order for the client to trust the server’s certificate otherwise the client will refuse to handshake with the server

**Main Server Functions:**

CheckFileIn:

1. Check if the user already logged in
2. Process the flag and set integrity and confidential based on the flag
3. Read the file received from the client
4. If it’s confidential we are going to encrypt the file content using a symmetric key encryption and then encrypt the key using the public key server has.
5. If it’s integrity we are going to sign the document
6. If the file id does not exist in the files database, that means it’s a new file and the owner is the client. So the owner will have the authorized to create this file. And then we are going to create the meta data for this file which includes:
   1. file id (hash(client\_id + filename)
   2. integrity (boolean)
   3. confidential(boolean)
   4. client\_id,
   5. file\_path(we are going to store this file inside the server’s file system)
   6. key(the key used to encrypt the file content, if no key is used key = “”)
   7. sign(the signature we used to sign the file content, if no signature require sign = “”)
   8. filename(the file id sent by the client)
7. If the file is not a new file then we are going to get the file information from the files database
8. if the client owns the file we will allow the client to rewrite the file and to rewrite the metadata
9. If the client is not the owner for this file we will check the file’s delegation list to see if the client has the right to check-in this file (!!! Check-in right needs to be implement). If the client have such rights, we will allow the client to update the files but not the meta data.
10. At the end we will return the file id create by the server

CheckFileOut:

1. Check if the user already logged in
2. If the client is the owner of the file, then we will allow the client to check out the file
3. If the client has delegation to check out the file then we will allow the client to check out the file too.
4. And then we are going to open and read the file from the file system
5. If confidential is required for the file, then we are going to decrypt the file using server’s private key and then use the decrypted key to decrypt the file content
6. If integrity is required for the file, then we are going to verify the signature store in file’s meta data
7. And then we are going to send the file content back to the client

Login:

1. Checks the clients key to see if it matches the stored key
2. It checks to see if the user is already in a session
3. If it passes it will generate and store a session key and then return that key to the user

Delegate:

1. Set authorization variable to false
2. Check to see if users session key is valid
3. Checks to see if the requesting user is the owner of the file it has requested to modify if so the authorization variable is set to true
4. It will then check if the user is in a delegate, if so it ensures its time slot has not expired and then sets the authorization variable to true
5. If the authorization variable is still false the program will deny access
6. Then if the user sets the “addPermissionTo” variable to “ALL” the system will add the rights posted by the user and time deadline to all users in the system
7. If the user set the “addPermissionTo” variable to a specific user it will check to see if that user exists and if so it will add the rights posted by the user and time deadline to the value of “addPermissionTo”.

RegisterUser:

1. This function will first check to see if the user exist in the system already
2. If not it will add the user’s login information to the database

SafeDelete:

1. The server checks to see if the requesting user is the file owner of the requested file if so it will set the authorization variable to true
2. It will then check to see if the user is a delegate with deletion permissions on the file, if so it will set the authorization variable to true
3. If the authorization variable is still false the system will deny the deletion request
4. Otherwise it will honor the request and delete the file

**Client Functions:**

CheckFileIn:

1. Client will open a file to check in to the server
2. Client will then collect and package metadata as json to send with the check-in file to the server
3. Next the client will send a post request to the server with metadata and the file it intends to chek-in
4. If login fails the client will quit

Login:

1. The client will package its ID and Key as metadata
2. Next it will send the login request with the metadata
3. The client will the store the session key returned by the server

Register:

1. The client will package its ID and Key as metadata
2. Next it will send the registration request with the metadata
3. The client will wait on a response from the server\

CheckOut:

1. The client will specify the file it intends to check out along with other meta data like session keys and other identifying information
2. The client will send the checkout request to the server
3. If the checkout fails the client will diagnose the reason why
4. Otherwise the client will save the file returned by the server

Logout:

1. The client will send its user ID as metadata to the server
2. The server will then proceed to deactivate clients session key