**Bounced** Overview of System Architecture

# Nomenclature

* F: Name of a file
* Client A: Requestor of the file, F
* Client B: Holder of the file, F
* Client C: One of Client A’s friends
* H1: Initial hash of the file, H1
* H2: Secondary hash of the file H1 using asymmetric encryption if we decide to use that instead of HTTPS

# Overview

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Sender | Receiver | Type | URL | Body |  |
| 1 | Client | Server | POST | /register/ | Mac,Nick |  |
| 2 | Client | Server | POST | /sync/ | JSON with folder details |  |
| 3 | Client | Server | GET | /pending/ | - |  |
| 4 | Client | Server | GET | /search/kw | - |  |
| 5 | Client | Server | POST | /download/ | MAC, FileHash |  |
| 7 | Client | Client | TCP | - | - |  |
| 8 | Client | Server | POST | /update/ | transferID [newhash] |  |
| 9 | Client | Server | POST | /notify/ | newHash,transferID |  |

# Scenario 1 – Client makes registration request

**Server – HTTP POST /register/**

**Body: MAC, nick**

1. This is the first request that a client makes to the server on application start.
2. The server will confirm that nick and MAC are unique.

## Security Threats

1. MitM attack on request and response.

## Solutions

1. HTTPS

# Scenario 2 – Syncing latest index with server

**Server – HTTP POST /sync/**

**Body: JSON containing folder details**

1. Client will begin compute of file hashes in selected folders on application start, as soon as this completes, the client will make this request. In case no files have changed (ie: hash of old index matches hash of new index) the client will simply send an unchanged flag as well.
2. Server will respond with confirmation

## Server Actions

1. Server will now begin comparisons to determine whether other clients that need files from the client which has just appeared online. If yes, the appropriate transfers will be added to their pending requests.

# Scenario 3 – Client polls server to determine if there are pending transfers to be initiated

**Server – HTTP GET /pending/**

1. At regular intervals of time, the client will make this request
2. The server will return a list of transfer orders that the client must execute (ie: pending transfers) if any.
3. If pending transfers are for a friend, the server will generate and return a key along with each such transfer. Scenario 6 will then occur.

## Security Threats

1. MitM attack on request and response.

## Solutions

1. HTTPS

# Scenario 4 – Client queries server for files, given search keywords

**Server - HTTP GET /search/keyword1+keyword2**

1. Client A sends search request to the server.
2. Server returns search results, including online and offline files

## Security Threats

1. MitM attack on search query and results

## Solutions

1. HTTPS

## Client actions

1. User will be presented a list of files, from which he may select an online or offline file

# Scenario 5 – User wants/has to download a file, requests it

**Server - HTTP POST /request/**

**Body: Hash, MAC of file holder**

1. Client makes HTTP request to the server with the requested file hash, MAC of user having the file, and other client-identifying information.

## Server Actions

1. Server will store all of this information in the clients pending transfers table, and will check this table whenever the client comes online.

## Security Threats

1. MitM attack on request

## Solutions

1. HTTPS

# Scenario 6 –Client requests the symmetric key to encrypt file with

**Server – HTTP POST /encrypt/**

**Body: MAC of receiver, Hash or just the transferID**

1. Client will inform the user that it is about to transfer the file to some other user.

## Server Actions

1. Server will generate a key and respond to the client with the key and add the key to the pending of the user who has requested the file.

## Security Threats

1. MitM attack on encrypt

## Solutions

1. HTTPS

# Scenario 7 – User requests the online file

**Peer – TCP**

1. The Client A initiates a **TCP**? connection with one of the online clients (Client B
2. The file is transferred over a new connection.
3. If it is receiving the file for itself, it compares the hash of the received file with the hash provided (H1).

## Security Threats

1. MitM attack on file being transferred.

## Solutions

1. HTTPS PUT only
2. HTTPS (for key exchange) followed by AES
3. RSA encryption for entire transfer

# Scenario 8 –Client informs server that it has received the file requested

**Server – HTTP POST /update/**

**Body: transferID**

1. Client will inform the user that it has downloaded the file it had requested.

## Server Actions

1. Server will remove all pending entries for the transferID received or set them to delete.

## Security Threats

1. MitM attack on encrypt

## Solutions

1. HTTPS

# Scenario 9 – Client informs server that it has received file for friend

**Server – HTTP POST /notify/**

**Body: newHash(H2), transfer ID (?)**

1. Client will simply inform the client that it has successfully downloaded a file for a friend.

## Server Actions

1. Server will remove the corresponding entry from the client’s pending table and add an entry in the pending user who requested the file to receive this file.

## Security Threats

1. MitM attack on encrypt

## Solutions

1. HTTPS