Project Name: 2FACloud

Tag: Zero knowledge two factor authentication web app.

Idea:

With all the password leaks that are happening in the past few months, it shows there is a greater need for 2 factor authentication. Unfortunately current 2FA solutions work as long as your phone works, and if you lose your phone you will have to use backup keys to access your account.

To resolve this issue we are creating a zero knowledge web application that syncs the 2fa private keys in an encrypted manner with the server.

The login process is as follows:

The user logs in with their username and password, their password is hashed and sent to the server. The server hashes their hash and checks it with the database.

Once the user logs in, the server sends the encrypted keys to the client and the client decrypts them with their password. The server has no information about what the keys actually are.

Once you’re logged in, you’re going to be greeted with a web page that has all your 2FA keys and a timer that creates new TOTP tokens every 30 seconds.

The website will have QR code reading and export/import functions for a better UX.

Functional requirements:

* Web server: Nginx with proxy-pass module.
* Web software: MySQL, NodeJS, Redis (possibility).
* Client software: Javascript, AJAX.
* Other tech: Encryption and hashing libraries.

Scope of development:

These are sorted by their priority:

* Database Structure
* Register page and code
* Login page and code
* Adding 2FA private keys from images and secrets
* Viewing 2FA keys that are already on the server for that user
* Removing 2FA keys
* Changing password
* Changing 2FA key used on registration
* Forgetting password
* Import/Export
* OAuth for integration with other services such as mobile clients.

Information flow:

Registration:

User will register with their username, email and password. Their password will be hashed with a very simple hash function (MD5) and sent to the server, the server hashes the hash once more and salts the hash and stores it in the database. Then the client has to setup a 2FA key with this account (in case their password is leaked, malicious users don’t have complete access to their 2fa keys). Once they set their 2FA key , their account is created and they can sign in.

Login:

User will login with their username and password. Once logged in, if the device is not recognized the user will have to provide a 2FA generated password created when making their account. Once that is completed, the server will send the private components of the 2FA keys stored on the web service to the user. The user locally decrypts these keys (SHA256) and arranges them in the website.

Adding 2FA codes:

The user will be able to scan QR codes or enter the codes for the 2FA codes manually. These codes are encrypted with the password of the user on the client side and get sent to the server. The server stores the encrypted values to be used for later use.

Removal is a very straight forward process.

Displaying them is explained in login procedure.

Changing password:

All decrypted 2FA keys will have to be re encrypted with the key provided. All known sessions will be forgotten and the user will have to re login on all clients with the 2FA code provided at sign up.

Changing 2FA key used on registration:

The 2FA key will only be changeable via the web interface and by providing the password of the client. The 2FA key will not be disabled in any way, either by email or other methods of communication. This will worsen the UX but for a security aware product, we can’t remove the 2FA key if the user has lost that 2FA key. The users will be heavily encouraged to print the backup codes provided to them at registration.