Target system architecture

Password Vault group

August 10, 2022

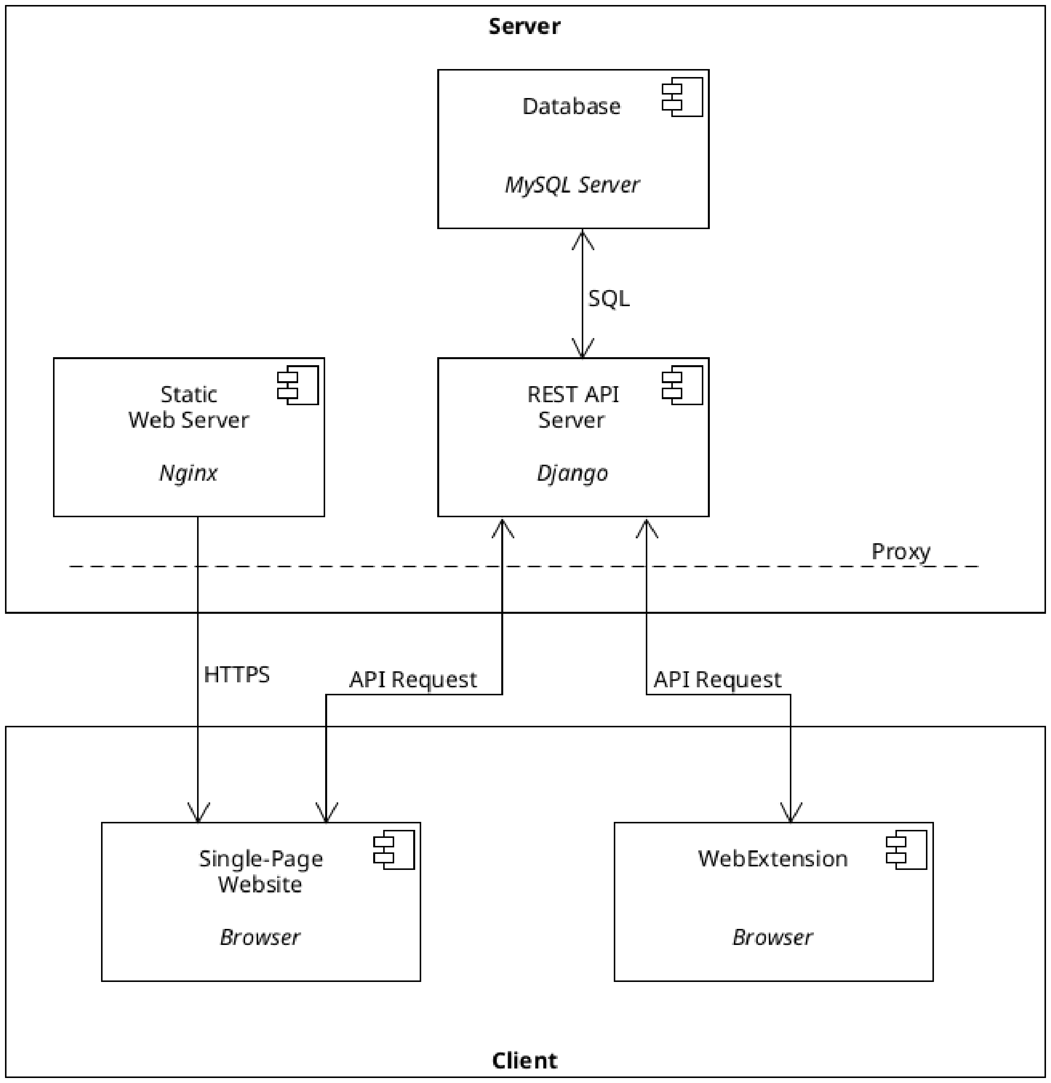

Figure 1: Overview of system components

1 Database

A relational SQL database will be a place where all users and their vaults are stored. As a SQL engine the PassV will use the MySQL server.

1.1 Database's functionalities

- Creating a user with password.
- Changing user's password.
- Inserting user's various credentials into the vault.
- Updating user's various credentials stored in the vault.
- Extracting user's vault so that it can be used securely offline.
- Removing user's vault completely so that it cannot be undone.

1.2 Database's security specification

- User's server access password will be stored in a secure way using one way hash function.
- User will not be able to access, view or modify any other data stored in the database that is not his whatsoever.

The SQL database will be managed via connection with implemented API Server.

User storing implementation Alpha version.

Vault storing implementation Beta version.

Whole database implementation Final version.

2 Web servers

2.1 NGINX Web Server

For our project we have decided to go with NGINX web server, which is a very fast and configurable web server that is very well suited to serve static files and act as a reverse proxy for other applications.NGINX and NGINX Plus provide a number of features that enable it to handle most SSL/TLS requirements.

There are number of other security features that support the decision, including (but not limited to) the following:

- End-to-End Encryption - NGINX can do both decryption and encryption, you can achieve end-to-end encryption of all requests with NGINX still making Layer 7 routing decisions.
- Client Certificates - Client certificates are a way of restricting access to your systems to only pre-approved clients without requiring a password, and you can control the certificates by adding revoked certificates to a certificate revocation list (CRL), which NGINX checks to determine whether a client certificate is still valid.
- SSL/TLS protocols – You can specify which protocols are enabled, including SSLv2, SSLv3, TLSv1, TLSv1.1, and TLSv1.2.

- Chained certificates – NGINX supports certificate chains, used when the website's certificate is not signed directly by the root certificate of a CA (Certificate Authority), but rather by a series of intermediate certificates. The web server presents a 'certificate chain' containing the intermediate certificates, so that the web client can verify the chain of trust that links the website certificate to a trusted root certificate.
- HTTPS server optimizations – NGINX can be tuned to maximum its SSL/TLS performance by configuring the number of worker processes, using keepalive connections, and using an SSL/TLS session cache.

2.2 Single-Page Website

The main description of the website used from the browser side. Additional security guidelines and information about the communication process.

Implementation time Beta version

2.2.1 Page performance and security requirments

- Linear oversimplified structure - the structure of the website should be relatively easy to use and facilitate the possibility of modification
- Direct access for the user to modify his master password
- Allowing the user to delete his account
- Option for the user to modify their credidentials from within the website
- Direct and permanent user access to his Vault

2.3 Proxy server

All communication outside of the server environment will utilize TLS encryption. To achieve this a separate NGINX instance will be deployed and configured as a proxy for both the web application as well as the API server.

Implementation time Beta version

2.3.1 Proxy usage and technicals

- Proxy server throughout communication - the communication procedure between the browser and the feedback in the context of the application's request to the server should run smoothly while maintaining the necessary security protocols.
- Applied proxy server operation - In addition to forwarding the request from the website, it should also allow to increase the overall efficiency of the processes being carried out. Improving the use of bandwith and improving the quality of network operation.

3 API Server

The API server is the central component of the server infrastructure of PassV. It is responsible for managing user accounts and storing vault data to allow for seamless synchronization between different client devices.

The API server will be implemented using the Django Web Framework.

Authentication of the user is handled by the server via JSON Web Tokens. The API Server keeps sessions with connected clients and accepts the refresh tokens to allow extension of the session.

On the backend the server uses an SQL connector to store and retrieve user information from the SQL database.

The public facing interface of the server is a REST API which exposes the following endpoints:

3.1 User management

`POST /user` - Create new user account (Alpha version)

- `user` - user email
- `password` - password

`GET /user/activate` - Activates user account (Final version)

- `token` - User activation token

`DELETE /user` - Delete user account (Beta version)
Header:

- `token` - session JWT

`PATCH /user/passwd` - Change user account password (Beta version)
Header:

- `token` - session JWT

3.2 Login session

`POST /token` - Return new session token (Alpha version)

- `user` - user email
- `password` - password

`POST /token/refresh` - Return refresh session token (Alpha version)
Header:

- `token` - session JWT

3.3 Vault data access

`GET /vault/key` - Return encrypted vault key (Beta version)
Header:

- `token` - session JWT

`POST /vault/key` - Change encrypted vault key (Final version)
Header:

- `token` - session JWT

Params:

- `key` - base64 encoded key data

Below endpoints allow the client to store and retrieve vault entries - a group of the basic component of the password vault. For the purpose of data interchange entries are stored in a JSON array, with each element consisting of an unique identifier, type string, access time as well as opaque data chunk in form of a base64-encoded text.

```
vault entries:
[
    {
        type: "credential"/"note"/"removed"
        id: <uuid>
        data: <credential domain,user,password>/<note name,text> (AES encrypted, base64 encoded)
        atime: <timestamp>
    },
    ...
]
```

`GET /vault` - Return vault data (Beta version)

Header:

- `token` - session JWT

Params:

- `since` (optional) - credentials modification timestamp
- `id` (optional) - vault entry id

`PUT /vault` - Update vault data (Beta version)

Header:

- `token` - session JWT

Params:

- `entries` - vault entries

4 Browser Extension

The PassV browser extension is the main client interface to the system.

4.1 Client and server side login

During startup the extension will inquire the user about the master password. After receiving said password the extension will use it combined with user's email address as well as server domain to derive two keys - one used for server access and one for local vault decryption.

Implementation time Alpha version

4.2 Password access

Web extension will provide access to the password via filling up the login forms and/or storing password in clipboard.

4.2.1 Login form completion

The extension will attempt to detect website login forms. Upon detection it will lookup the website domain name in the database and if an entry is present, fill out both the username as well as the password field. If the user logs in using a different username or password the extension will offer to update them in the vault.

Implementation time Alpha version

4.2.2 Clipboard access

Extension will allow the user to copy the password into the clipboard without revealing it on the screen. In addition it will provide the ability to clear the password from clipboard after a configurable period.

Implementation time Alpha version

4.3 Password generation

Web extension will provide method for generating strong secure random passwords during account setup and password changing. The generator will allow customizing password attributes (length, character set, etc.).

4.3.1 Password generator customization

The password generation interface will contain a set of controls allowing the user to change password attributes to satisfy website requirements.

Implementation time Beta version

4.3.2 Password rule parser

When available the password generator will detect and parse the `passwordrules`[1] attribute of a password input box and configure the password parameters to comply with those requirements. , which will fulfills the web page passwords requirements from password attributes .

Implementation time Final version

4.4 Local password storage

Web extension as a PassV client will provide a method for accessing the vault in offline mode i.e. without connecting to PassV server.

Implementation time Alpha version

4.5 Shared secret publishing

Web extension will provide a method for sharing/publishing secrets as a password protected web sites. User will be able to generate such secret and share it, giving it the password, via password protected website, which will be generated on the server.

Implementation time Final version

4.6 Credentials management

Web extension will provide a method for management the content of the vault. The user will be able to manage the content of the vault e.g. add, remove or change credentials. Client will also provide a user friendly interface, which allow the user to browse content of the vault. Moreover user will be able to copy the username or password to the clipboard or fill up login form on current open web page.

Implementation time Alpha version

4.7 Master password change

Web extension will provide a method for changing the master password of the vault in order to enhance the vault security.

Implementation time Beta version

4.8 Vault import/dump

Web extension will provide a method for dumping and importing the whole vault as a encrypted or plain-text file.

Implementation time Beta version

[1]Proposal for HTML passwordrules attribute - WHATWG GitHub issue

4.9 Server vault sync

Upon connection to a server, the extension will compare locally stored password database with the remote one. In case of differences between the two databases, the extension will merge them by picking newest credentials.

Implementation time Beta version