Codie Heaney (bh97rt)

23/05/2024  University of Sunderland

CET324 Design Rational

Table of Contents

[Design Decisions 2](#_Toc167396401)

[References 7](#_Toc167396402)

# Design Decisions

A screen shot of a computer

Description automatically generatedPython was chosen to develop the system due to the access it has to different libraries for ease of development and that allow for, encryption (“cryptography”), hashing (“hashlib”), and creating tokens (“jwt”). These libraries provide the necessary classes and functions to allow for implementing all the security features, and generating login tokens.

Figure 1:Library imports.

Once an account have been created with a username, password, and an access level this data is used to create a login token. Along with saving this information to a table to allow for logging in.

A screen shot of a computer code

Description automatically generated

Figure 2:Create new user function.

This token will contain all the information to sign a user into their account, with the username and password stored within it checked against a table to see if it matches. The username and password is hashed with a salt to ensure no plaintext is stored in the token.

A screen shot of a computer code

Description automatically generatedDuring the creation of the token each part of the body is encrypted using the previously mentioned AES Fernet encryption for additional security to prevent revealing the hashed values in the token.

Figure 3:Token creation function.

The token is a JSON Web Token (JWT) which is able to contain JSON objects to store all the necessary data for logging in and was chosen because of this for data storage. Additionally, each token comes with a signature which contains the information of the header and body of the token concatenated with a predefined key and hashed (Poddar, 2022). This allows for verification when processing the token to ensure it hasn’t been altered in anyway. This was chosen for this extra layer of security to prevent users making changes to their tokens to attempt to receive higher levels of access, by using the same key to hash the token header and body, and check it against the signature. In the event the body of the token has been altered the hash calculated will be different to the signature and it will be rejected. For additional security, in the event the key is known to the user and the signature is changed to correctly reflect the altered body of the token, a hash of the token is stored in a table and the token is checked against it to check the token hasn’t changed from its original form.

A computer screen with colorful text

Description automatically generatedTo login with a token the file path is inputted and the previously mentioned signature hash check and token hash check is done to verify it.

Figure 4:Token hash check.

A computer screen shot of text

Description automatically generated

Figure 5:Token input with hash and signature hash check.

If these checks pass the username and password are decrypted and checked against the stored usernames and passwords.

A black background with white text

Description automatically generated

Figure 6:Check login details.

A computer screen shot of a program code

Description automatically generated

Figure 7:Check login details from stored file.

If the username and password both match, the function returns true and the expiry date is checked against the current time, in the event of expiration the user is prompted to renter their login details which will be checked using the same function as in Figure 7, but with the token argument set to false so the user input is salted before checking.

A screen shot of a computer screen

Description automatically generated

Figure 8:Login in with login details.

If the details entered are correct, a new token is created using the outdate token information with an updated expiration date and replaces the current token file.

A computer screen with colorful text

Description automatically generated

Figure 9:Token expiration check.

With all checks passing, or a new token being created if it has expired, the user is informed of the amount of time left until their token expires as well a message to inform them that the login was successful and further functionality can be implemented to take them to the desired system the company uses.

To achieve the hashing shown in the previous figures, the “hashlib” library was used and the “SHA256” algorithm was selected which creates 256-bit hash values. This algorithm was chosen instead of another method such as MD5 due it not suffering to many types of collision attacks other algorithms fail to (Gilbert, *et al.* 2004). Due to its 256-bit size outputs the chances of collisions are indefinitely small, and although MD5 has been shown to be faster by Rachmawati, *et al*. the security benefits outweigh the slight increase in time taken.

Two implementations where used for this system, a function to hash text with a randomly generated salt which returns the hash and the salt used, and a function to hash text without generating a salt intended to be used for text that either doesn’t require a salt or for text that has been previously salted before requiring hashing.

A screen shot of a computer program

Description automatically generated

Figure 10:Hash class containing all hash functionalities.

With relation to encryption the system uses Fernet to encrypt all data which uses the Advanced Encryption Standard (AES) in cipher block chaining mode with a 128-bit key (Cryptography, ND). This method is used to prevent brute force attacks on decrypting data due to the larger key size, as compared to DES which uses keys of 52-bit (Baivab, 2023).

To implement this, a key is generated and saved to a file, or if a key already exists its loaded into the key variable and is used for all encryption and decryption.

A screen shot of a computer code

Description automatically generated

Figure 11:AES key generation.

To encrypt and decrypt, the Fernet library provides a function that uses the key to do so and returns a byte string with the encrypted or decrypted text which is returned as a result of the overall function.

A screen shot of a computer code

Description automatically generated

Figure 12:Encrypt and decrypt text.

# References

Bavib, J. (2023) “What Is AES Encryption and How Does It Work?”, Available at: <https://www.simplilearn.com/tutorials/cryptography-tutorial/aes-encryption> [Accessed: 10/05/2024]

Cryptography. (ND) “Fernet (symmetric encryption)”, Available at: https://cryptography.io/en/latest/fernet/ [Accessed: 10/05/2024]

Poddar, R. (2022) “What is JWT? Understanding JSON Web Tokens”, Available at: <https://supertokens.com/blog/what-is-jwt> [Accessed: 10/05/2024]

Gilbert, H., Handschuh, H. (2004) “Security Analysis of SHA-256 and Sisters”, *Selected Areas in Cryptography*, vol. 3006, pg. 175-193.

Rachmawati, D., Tarigan, J., Ginting, A. (2018) “A comparative study of Message Digest 5(MD5) and SHA256 algorithm”, *Journal of Physics: Conference Series*, vol. 978.