SECURESNAP: DOCUMENTATION

Total lines of Python code: 2,277

A. THREAT MODEL

- Threats:
    - Dolev-Yao Attacker Model
        - Our system is premised on the Dolev-Yao attacker model, which assumes that a malicious actor can view, edit, and create information transmitted across a network. To address these data concerns, our system makes use of encryption and integrity protocols to protect user information, even if it is viewed/altered by a malicious actor.
    - DOS Attacks
        - As a network-based system, our system is prone to DOS attacks. To address this, our authentication and authorization features have been designed to control the volume of requests that can be made by a user.
- Non threats:
    - Trusted hardware
        - We assume that our server hardware is trustworthy and secure, and located in a secure physical environment removed from malicious actors.
    - Trusted superadmin/admins
        - We trust that our superadmin/admins are trusted parties that does not have malicious intents, and that their interests lie in auditing and supervising user actions in our system.

B. SECURITY GOALS AND CORRESPONDING FEATURES

1. Authentication
    a. Overview
        - The system can verify the identity of each user attempting to access our system by implementing a login system. This is in the form of usernames and (salted and hashed) passwords stored on a local csv file.

    b. Passwords
        - Generating and storing passwords:
            - The *secrets* module in python is used to generate a unique salt for each password, which is concatenated with the password to create a salted password. This salted password is then hashed using the SHA-256 hashing algorithm
        - Password policies
            - Our password policy is defined as follows: minimum 5 characters, 1 special character, 1 number, and 1 uppercase character. Passwords are stored in their corresponding record within the

userinfo.csv file, in the format *username, userID, (salted and hashed) password, salt*

- Authenticating passwords
    - When a user inputs a password, a salt is generated with the secrets module and concatenated with the user input to create a salted user input. The salted user input is then hashed using the SHA-256 hashing algorithm. The (salted and hashed) user input is compared with the (salted and hashed) stored password. If both values match, the user is authenticated and can access the system

2. Verification
    a. User verification via email
        - When a user creates an account, they are prompted to enter their username, which is the email they used to create an account. A 6-digit PIN is sent to their email and the user is prompted by the system to enter their PIN. If the entered PIN and the sent PIN match the email is verified and the user record is created. The PIN resets every user session.

    b. Password recovery via email
        - Password recovery utilizes the same verification system as when an account is created. There is one additional check where the system checks if that the username/email exists within the system.

3. Authorization
    a. Overview
        - The system can determine and enforce the actions allowed for an authenticated user. It ensures users can only perform actions they're permitted to, such as viewing or uploading photos.

    b. Authorization levels
        - Users
            - Send and receive photos
            - Block Users
            - Delete their account
            - Password Recovery
        - Admin
            - Create other admin
            - Create other users
            - Delete user accounts
            - Reset user passwords
            - View Logs
        - Superadmin
            - Create other admin
            - Create other users
            - Delete user accounts

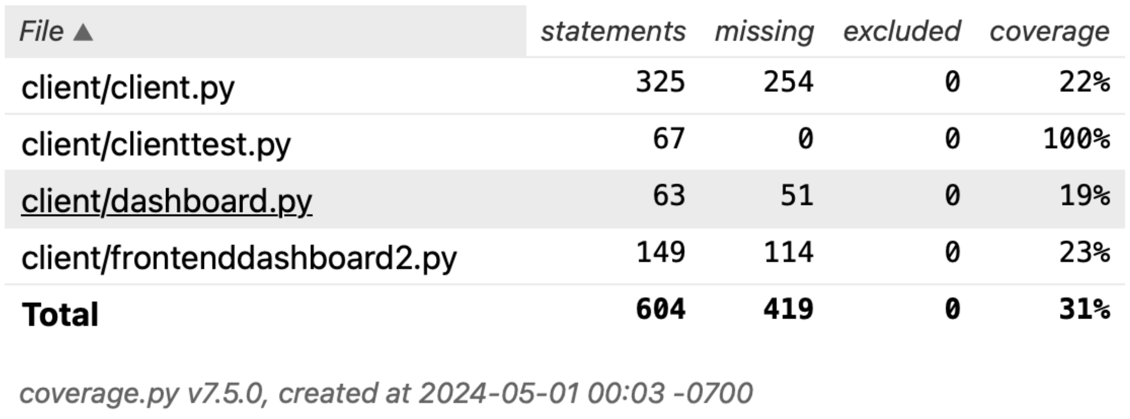
        - Delete admin accounts
        - Reset user passwords
        - Reset admin passwords
        - View Logs
   c. Examples of unauthorized actions:
      - Users cannot send photos to admin/superadmin
      - Users cannot delete admin/superadmin accounts
      - Limit the size of the image that can be sent over
      - We also make sure to check that the file being sent over ends in a .png, .jpg, and .jpeg extension

4. Confidentiality
   a. Overview
      - The system ensures that only the administrator and the user themselves should be able to access that user’s login data and audit logs. The system also ensures that the file's contents can only be read by the recipient.
   b. Implementation
      - Users cannot view, edit, delete, or replay messages sent across the network between other users. Every chunk of data is encrypted using TLS when being sent over the network. Files stored on disk that have private information should be encrypted (e.g. passwords).
5. Integrity
   a. Overview
      - The system employs integrity checks to ensure that a file is not corrupted in transit or replaced with harmful content, addressing the Dolev-Yao threat model.
   b. Implementation
      - The system uses TLS Cipher Protocol version 1.2, and Message Authentication Codes (MACs) from the data and a secret key to confirm authenticity before transmission. The data and the MAC are both encrypted for secure transmission. Symmetric encryption is used w/ CBC or GCM for integrity checks. The data is verified upon receipt through decryption, and the MAC is recalculated to ensure no changes occurred during transit.
6. Audit
   a. Items logged:
      - Timestamp
      - User ID
      - username
      - Actions
        - Users/Admin/Superadmin log in successfully/unsuccessfully
        - Users/Admin/Superadmin attempt to login unsuccessfully more than 3 times

- Users/Admin/Superadmin logouts
- Admin/Superadmin create/delete accounts
- Blocked users

C. ASSURANCE

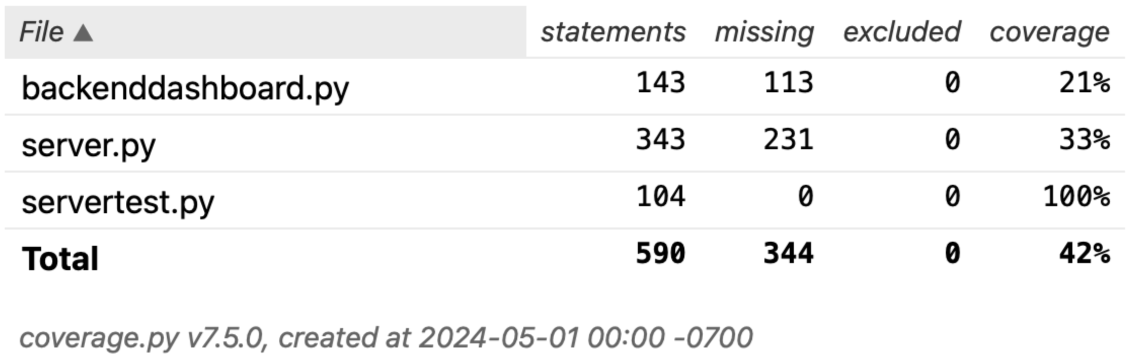
Unit testing on client side:



File ▲	statements	missing	excluded	coverage
client/client.py	325	254	0	22%
client/clienttest.py	67	0	0	100%
client/dashboard.py	63	51	0	19%
client/frontenddashboard2.py	149	114	0	23%
Total	**604**	**419**	**0**	**31%**

coverage.py v7.5.0, created at 2024-05-01 00:03 -0700

Unit testing on server side:



File ▲	statements	missing	excluded	coverage
backenddashboard.py	143	113	0	21%
server.py	343	231	0	33%
servertest.py	104	0	0	100%
Total	**590**	**344**	**0**	**42%**

coverage.py v7.5.0, created at 2024-05-01 00:00 -0700

D. ETHICAL CONSIDERATIONS

- User Privacy:
    - Ensuring user data privacy and confidentiality by implementing robust encryption and access controls.
    - Users' data privacy and confidentiality is our main consideration in our system, in both the images users send and user information that we store. All our security features outlined above are designed to ensure that user data is secure in transit (following the Dolev-Yao model) and in storage.
- Transparency

    - There are additional security features we chose not to implement due to time constraints, such as allowing users to accept or deny photos. Both existing features and limitations would need to be communicated with users to facilitate user trust in our system.
- Data security
    - Our system makes extensive use of existing cryptographic and cipher protocols. To maintain the security of our system, we would have to regularly update our system and the protocols we use to keep up to date with any new security flaws
- User accountability
    - Our system assumes that there are no malicious users on our network – realistically, there will be some bad actors on our system. Guided by our audit system, we would need to establish protocols/procedures/guidelines to maintain user accountability. This could include banning users for a certain period of time, or removing them from the system entirely if they misused their system privileges or did not follow our system guidelines.