User Authentication and Access Control System for justInvest Security

Course: SYSC 4810A: Software and Network Security
Instructor: Hala Assal
Submission Date: December 3 2024
Submitted by: Kyle Taticek 101193550

Problem 1: Access Control

Access Control Mechanism (1a)

I used the **Role-Based Access Control (RBAC)** model to implement the access control mechanism. RBAC was chosen because it aligns well with the justInvest system requirements by associating roles with predefined permissions. This simplifies policy management, reduces errors in individual permission assignments, and provides a clear separation of concerns.

Access Control Policy(1b)

Below is a role-permissions matrix representing the access control policy:

Role	Operation 1	Operation 2	Operation 3	Operation 4	Operation 5	Operation 6	Operation 7
Client	✔	✔	X	✔	X	X	X
Premium Client	✔	✔	✔	✔	✔	X	X
Financial Advisor	✔	✔	✔	X	X	X	✔
Financial Planner	✔	✔	✔	X	X	✔	✔
Teller	✔	✔	X	X	X	X	X

Annotations:

- Operations:
  - (1) View account balance
  - (2) View investment portfolio
  - (3) Modify investment portfolio
  - (4) View Financial Advisor contact info

    - (5) View Financial Planner contact info
    - (6) View money market instruments
    - (7) View private consumer instruments
- Business hours restrict Tellers to operations 1 and 2 only during 9 AM to 5 PM.

Test Cases for Access Control (1c)

Tested Scenarios:

1. **Role Validation**:
    - Attempted operations for each role to confirm authorized actions were allowed and unauthorized ones were denied.
    - Example: A Client attempting operation 3 (Modify portfolio) received "Access Denied!".
2. **Business Hours Enforcement**:
    - Tested Teller access outside of 9 AM to 5 PM, ensuring operations were denied.
3. **Role-Specific Operations**:
    - Validated Premium Client's ability to perform operations 1–5, but not 6 or 7.

Coverage Justification:
The test cases cover all roles, operations, and conditions (e.g., business hours), ensuring comprehensive validation of the RBAC policy.

Problem 2: Password File

Chosen Hash Function and Parameters (2a)

Hash Function: **BLAKE2b**
Justification:
BLAKE2b provides high performance, a customizable digest size, and robust cryptographic security. It is resistant to collision and pre-image attacks, making it ideal for password hashing.

Parameters:

- **Hash Length**: 32 bytes, ensuring sufficient entropy for secure storage.
- **Salt Length**: 16 bytes, chosen to prevent precomputed attacks (e.g., rainbow tables).
- **Salt Generation**: Randomly generated using secrets.token_bytes, a cryptographically secure source of randomness.

Password File Format(2b)

The password file passwd.txt stores user credentials in the following format:

<username>,<hashed_password>,<salt>,<role>

Hashing Process (2c)

1. A salt (16 bytes) is randomly generated using secrets.token_bytes.
2. The password is hashed using BLAKE2b with the generated salt.
3. The salt and hash are stored alongside the username and role.

Example Record

testuser,dbce1f112af996542a591f61dee6085d10ab240cbe002ae9dd62db53f5d87a8f,c5263c493f57ee6611e0167e6852565a,Client

Components:

- **Username**: Identifies the user.
- **Hashed Password**: Securely stores the password, ensuring confidentiality.
- **Salt**: Ensures uniqueness of hashed passwords, even if two users choose the same password.
- **Role**: Links the user to a predefined role in the access control policy.

Test Cases for Password File (2c, 2d)

Tested Scenarios:

1. **Salt Uniqueness**:
    - Verified that salts were unique for each record.
2. **Password Verification**:
    - Tested login with valid and invalid passwords to ensure correct validation.
3. **File Integrity**:
    - Inspected passwd.txt to confirm all components were stored correctly.

Coverage Justification:
The test cases validate the integrity, correctness, and security of the password file, covering all required aspects.

Problem 3: Proactive Password Checker

Test Cases for Enrollment and Password Policy (3a, 3b)

Tested Scenarios:

1. **Valid Password**:
    - Verified acceptance of compliant passwords (e.g., "ValidPass1!").
2. **Length Violations**:
    - Tested rejection of passwords shorter than 8 or longer than 12 characters.

3. **Composition Violations**:
    - Tested rejection of passwords missing required elements (e.g., no uppercase letters).
4. **Username Inclusion**:
    - Verified rejection of passwords containing the username.
5. **Weak Passwords**:
    - Ensured common passwords (e.g., "123456") were rejected.

Coverage Justification:
These cases cover all rules in the password policy, ensuring no invalid passwords are accepted.

Problem 4: Login Interface and Access Rights

Test Cases for Login and Access Validation (4a, 4b, 4c)
Tested Scenarios:

1. **Successful Login**:
    - Confirmed users with valid credentials could log in and see allowed operations.
2. **Unauthorized Operations**:
    - Verified attempts to perform unauthorized actions displayed "Access Denied!".
3. **Invalid Credentials**:
    - Ensured login was denied for incorrect usernames or passwords.

Coverage Justification:
The test cases validate the functionality and security of the login process and ensure access rights are enforced.