Tracing Security Requirements

Prompt: trace ALL security requirements to where they are implemented in the application. You must also explain how your system implements the three principles of the reference monitor: always invoked, tamperproof, and verifiable.

Identification:

Unique User ID <Integrity>

- When combined with authentication, ensures that actions taken can be logged and associated with a user
- Implementation: This policy is enforced in the AddUser class. If a userID already exists, then an exception is thrown and the creation is disallowed.

No Backdoors <Integrity>

- All interfaces of the software that are accessed for performing any action shall have the capability to recognize the user ID.
- Identification must be applied across all system interfaces. In the event that a “backdoor” exists through which access is granted with no identification, the security of the system would be compromised
- Implementation: There is a standard authorization module, which is invoked with any request to interact with the system.

Authentication:

Credential Security <Confidentiality>:

- The system shall store the information used for authentication in a secure manner, using public and widely accepted crypto algorithms.
- Authenticating information must be stored in such a way so that a third party without authorization to do so cannot easily obtain it.
- Implementation: All information is encrypted when stored inside a database. No information is left in the open for people to see. Authentication is implemented through a set of levels. All actions require a user authentication to be performed so that no sensitive information is leaked.

Protect Credential Guessing <Integrity, Confidentiality>

- Restrict number or time of attempts on credentials for any given user ID. This serves to prevent brute force attacks on the system
- Implementation: Time is exponentially longer after login is failed for a specific username after a set number of attempts. This will eliminate brute force tactics after repetitive attempts.

Protection of credentials <Confidentiality>

- The application shall not divulge in clear text the static authenticator (e.g., password, PIN number, token seed, smart card seed, etc.) of one user to any other user, including administrators.

- Passwords shall not be transmitted, stored, or echoed in cleartext. Administrators must have the ability to make changes to authentication information, but must not be able to easily impersonate the user.
- Implementation: Everything is encrypted as soon as it enters the system. Not even the admin is able to see the information in the system.

Password Complexity <Confidentiality>

- The system shall require that the authentication information is configurable to administrator- specified characteristics for minimum length, alphabetic characters and numeric or special characters.
- Use of trivial and predictable authenticators makes it easier for a third party to obtain an authenticator through brute-force attacks, such as dictionary attacks and other cracking methods.
- Implementation: There is a check on new passwords for new users that ensures that their chosen string surpasses a certain length to ensure security strength. It is also checked against a list of common passwords so that a hacker can not guess easy passwords that may be used often by users.

Authorization:

Access Rights <Integrity, Non-repudiation>

- The system shall not allow access to system resources without checking the assigned rights and privileges of the authenticated user.
- Authorization is useless unless tied to something that maps identification to rights or privileges. Authorization controls must be applied across all users, resources, and interfaces. (Monitor)
- Implementation: This system enforces Role Based Access Control Policy (RBAC). Consequently, each user entity has a role associated with it both in the active and database state. This RBAC is enforced with every request to interact with the system via the Authorization module.

Account Lock-out <Confidentiality>

- If several consecutive incorrect login attempts are made, the system shall generate an alarm and also lock-out the account (for a specified period of time or indefinitely, depending on the criticality of the role and application) after an administrator-specifiable number of attempts.
- This serves to safeguard against brute force type attacks on a specific user ID.
- Implementation: After a specified amount of failed login attempts, the system will lock the user out for an amount of time exponentially proportional the number of failed login attempts.

User and Group Privileges <Confidentiality>

- The system shall have features to assign user and group privileges (i.e., access permissions) to user IDs.
- This will assist in the creation of a reference monitor which will be an integral part of the product’s security

- Implementation: This system enforces Role Based Access Control Policy (RBAC). Consequently, each user entity has a role associated with it both in the active and database state. This RBAC is enforced with every request to interact with the system via the Authorization module.

Role-Based Access Control <Integrity, Confidentiality>

- The system shall provide an enforceable mechanism through which users can be segmented into roles, involving access to security features and other administrative functions.
- This will assist in the creation of a reference monitor which will be an integral part of the product's security
- Implementation: This system enforces Role Based Access Control Policy (RBAC). Consequently, each user entity has a role associated with it both in the active and database state. This RBAC is enforced with every request to interact with the system via the Authorization module.

Resource Control Mechanism <Confidentiality>

- The system shall provide a resource control mechanism that grants or denies access to a resource based on user and interface privilege.
- This will assist in the creation of a reference monitor which will be an integral part of the product's security
- Implementation: This system enforces Role Based Access Control Policy (RBAC). Consequently, each user entity has a role associated with it both in the active and database state. This RBAC is enforced with every request to interact with the system via the Authorization module.

Security Auditing:

Audit Log <Non-repudiation>:

- The system shall maintain an audit log that provides adequate information for establishing audit trails on security breaches and user activity
- The event log is useful in finding the root of an attack or incident and holding the accountable user or attacker responsible. It can also help to find suspicious patterns.
- Implementation: There exists an AuditLog class that implements an audit log. This audit log is encrypted and access is enforced by RBAC policy. The audit log tracks all relevant information to the auditor (administrator), as specified in the Command Line interface document.

Action on Audit Log Failure <Non-repudiation>:

- The system shall give the administrator a variety of actions to perform in the circumstance that the Audit Log fails. Those actions include termination of the program and a reset of the log
- Although audit log information could be critical for forensic purposes and for detection of inappropriate or unauthorized activity, the system must allow the administrator to specify whether or not the system should continue to function when the log function is no longer able to perform.

- Implementation: Audit Log recovery feature is implemented in the AuditLog class, in the form of a specialized user interface interacting with UserInput class.

Confidentiality Requirements:
Sensitive Information Protection<Confidentiality, Integrity>

- The system shall be able to protect information from those who are not granted access to the sensitive data
- A loss of integrity and confidentiality would occur if the data was breached in the system. It would also cause the information to be released to people who do not permit the authentication to view it
- Implementation: Only users have the encrypted key to their address databases and they may only edit, read, or delete records from their own database. Users need to give their password in order to access their databases and the passwords are encrypted in the user database.

Cryptographic Key Security <Confidentiality>

- If keys are generated, the system should be able to protect the keys from all people who are not granted access.
- These keys are essential to cryptography so it is of the utmost importance to protect these keys. To maintain the confidentiality of a system and the purpose of the key, the keys must be kept safe.
- AES keys are only accessible by unique users only when they are logged in. These keys are however not stored anywhere in the system and are unobtainable by other users and the admin.
- Implementation: AES keys are protected with encryption based on the password of the user.

Cryptographic Key Strength <Confidentiality>

- An algorithm is needed that will produce keys that are unpredictable and of good length.
- Cryptography is only as strong as the key strength. The stronger the key, the stronger the system is against vulnerabilities.
- Implementation: Keys are generated with the help of the following website: https://www.novixys.com/blog/how-to-generate-rsa-keys-java/. Bcrypt is used because it is a reputable algorithm that produces unpredictable keys.

Integrity Requirements:
The integrity of Sensitive Information <Integrity>

- The system shall support protocols that bind the integrity of sensitive information with the integrity of the associated protocol information.
- Integrity of network data (e.g., protocol headers) should be tied to integrity of packet data, to further ensure the integrity of the transmission.

- Implementation: Only users can add or edit their address databases. Users are expected to enter the correct address, but the code will check that the fields of the address are in a proper format

Integrity of Logs <Integrity>

- The system shall have the capability to protect the integrity of audit log records by generating integrity checks (e.g., checksums or secure hashes) when the log records are created, and by verifying the integrity check data when the record is accessed.
- A common technique, as part of an attack, is to alter the log and audit records on a system to hide unauthorized activity. Integrity checks on these records can help prevent such activity.
- Implementation: Log integrity is enforced with RSA encryption, ensuring that the admin alone has access to it. Additionally, actions associated with a user are signed by that user with RSA encryption, to maintain integrity and source identification.

Integrity Checks <Integrity>

- The system shall have the capability to protect data integrity by performing data integrity checks and reject the data if the integrity check fails.
- Data integrity is a large issue, and threats take many forms. The system should take steps to ensure that the integrity of data is maintained at all relevant points.
- Implementation: The data is protected by encryption of a protected user key in the User and Address Databases.

Immunity Requirements:

Default Deny <Confidentiality>

- The system shall deny access unless a user has permission to access a resource.
- By taking the default approach of denying unless authorized, we minimize the risk of an unauthorized back door attack
- Implementation: The system limits who can access what information by setting implementing a system of classifications. Administrators have the most access over the system but can not access the sensitive information of each user.

Survivability Requirements:

Logging through System Restarts <Integrity, Non-repudiation>

- The system shall allow the audit log and its control mechanisms to maintain integrity and completeness through system restarts.
- System restarts must not clear the log. Since a shutdown or restart can be associated with a security-related event (or even be one on its own), it is necessary for logs to note them as an event and to resume normal operations upon restart.
- Implementation: Databases are stored in hidden files that are reopened when the application is started and stored when the application is exited.

Privacy:

Personal Information Identification and Classification <Confidentiality>:

- The system shall identify all personal information in order to keep it private in the processes of moving or backing up data.
- The system shall provide a classification between user and admin data in order to decide what should be viewed as private data and what should be viewed as public data.
- Implementation: Classifications exist to limit what information is revealed to the admin. Although the admin has many controls over the application, they do not have access to the encrypted data that is only disclosed to the unique user.

Logical Access Controls <Confidentiality, Integrity, Non-repudiation>

- Identifying and authenticating internal personnel and individuals
- Making changes and updating access profiles
- Preventing individuals from accessing anything other than their own personal or sensitive information
- Limiting access to personal information to only authorized internal personnel based upon their assigned roles and responsibilities
- Restricting access to system configurations, super-user functionality, master passwords, powerful utilities, and security devices (for example, firewalls)
- Preventing the introduction of viruses, malicious code, and unauthorized software
- Reasoning: The eighth principle of the Generally Accepted Privacy Principles (GAPP) is Security for Privacy. This principle requires that the entity protect personal information against unauthorized access (both physical and logical).
- Implementation: The user needs to log in to access any address records and the address database uses the userid hash as a key to access that user's address records.

Three principles of Reference Monitors

a. Isolation
    i. Admin and users have an asymmetric key.
    ii. Each private key is encrypted with an intermediate key which is encrypted with a salted hash of the user's password (to prevent plaintext private keys)
    iii. Private keys are only decrypted when they need to be used
b. Completeness
    i. Every access to the data must go through the RM (ie. non-bypassability)
    ii. Implemented in the form of Authorization class, which must be passed for every command to execute.
        1. RBAC is enforced in conjunction with the Authorization class to this end.
c. Verifiability

i. NO SPAGHETTI CODE (Keep it simple, stupid). Also known as consolidating the code into modules that make features easier to verify. For example, there is a single class of Authorization, which all Authorization requests pass through.
ii. High cohesion, low coupling