CS431:CNS

Security Guard Design Document

Project 4

Team Members:

1. Aditya Mehta (aditya.mehta@iitgn.ac.in)
2. Anura Mantri (anura.mantri@iitgn.ac.in)
3. Daksh Jain (daksh.jain@iitgn.ac.in)
4. Harshit (harshit.harshit@iitgn.ac.in)
5. Pavani Tirunagari (venkata.sathya@iitgn.ac.in)
6. Reddybathuni Venkat (reddybathuni.venkat@iitgn.ac.in)
7. Soumya Malviya (soumya.malviya@iitgn.ac.in)

Overview:

This project implements a secure logging system to track the entry and exit of guests and employees within the IIT Gandhinagar campus. The system is designed to ensure the privacy and integrity of the logs, even in the presence of an adversary. Two main programs, logappend and logread, are developed to manage and query these logs.

Languages/ Frameworks:

C++: The core implementation of the secure log system will be done using C++. It is chosen for its performance, fine-grained control over system resources, and ability to interface with MongoDB using appropriate libraries.

MongoDB: Each log entry will be stored as a document in MongoDB. A document in MongoDB is a JSON-like structure that can include various fields, such as timestamp, person_id, role (guest or employee), action (entry, exit, or movement within the campus), location (building/room), and authentication_token.

OpenSSL and Crypto++: To ensure the security of the log data, OpenSSL and Crypto++ will be utilized for encryption and cryptographic operations. OpenSSL provides a robust library for SSL/TLS protocols, ensuring secure communication between the application and the MongoDB server. Crypto++ will be used for additional cryptographic functionalities such as generating secure tokens, hashing, and implementing custom encryption algorithms where needed. These libraries help protect the data both during transmission and while it is stored, ensuring that sensitive information remains confidential and tamper-proof.

Architecture:

The system consists of the following key components:

1. logappend (Client side)

- **Role:** This component is responsible for collecting data from the user, such as name, room number, timestamp, etc, and then sending it to the buffer for further encryption, which is then sent to the server.
- **Responsible Member**: Anura and Harshit

2. logread (Client side)

- **Role:** The component handles all the queries for retrieving log data from the database. It sends requests to the buffer, encrypts them, and then sends them to the database to fetch the required information.
- **Responsible Member**: Anura and Harshit

3. Network Server (Aditya, Pavani)

- **Role:** This central control hub manages all incoming client requests, whether appending new logs or reading data. The server also validates tokens and forwards requests between clients, the buffer, and the database.
- **Responsible Member**: Aditya and Pavani

4. Buffer (Daksh, Venkat)

- **Role:** The buffer is the middle ground between the server and the database. It ensures data is properly encrypted, decrypted, verified, and rate-limited before reaching the database. The buffer is where the data goes first before it can enter the database.
- **Internal Modules:**

    - **Rate Limiting:** The buffer controls how many requests can pass to MongoDB within a specific time frame (for example, no more than 10,000 entries per second).
    - **Data Verification:** It checks the integrity of the data by comparing the checksum of what it received with what was initially sent. This makes sure no tampering has occurred.
    - **Encryption:** Before sending data to the database, the buffer encrypts sensitive information to keep it secure.
    - **Decryption:** Decrypts data when queried by the logread client.
- **Responsible Member**: Daksh and Venkat

5. MongoDB Database (Soumya, Venkat)

- **Role:** This is where all the log data is securely stored. The buffer is the only component with direct access to read and write data into MongoDB.
- **Responsibilities:**
    - **Data Storage:** It safely stores all log entries in an encrypted format, ensuring they cannot be easily accessed or modified.
    - **Query Handling:** When a query comes in from the buffer, MongoDB retrieves the necessary data and sends it back.
- **Responsible Member**: Soumya and Venkat

Command Execution Pipeline:

logappend command:

- logappend (Client) → Buffer(Client-side) → Network Server → Buffer(Server-side) → MongoDB → Buffer(Server-side) → Network Server→ Buffer(Client-side) → logappend (Client)

logread command:

- logread (Client) → Buffer(Client-side) → Network Server → Buffer(Server-side) → MongoDB → Buffer(Server-side) → Network Server → Buffer(Client-side) → logread (Client)

Functionality:

logappend program

The `logappend` command is used to manage a log of events, such as arrivals and departures of employees or guests at a campus or in specific rooms. The command

records these events with a timestamp and ensures the log's consistency by validating each entry. Here is a brief overview:

Arguments:

- **-T `<timestamp>`:** Records the time of the event. The timestamp must be a positive integer and must be greater than the last recorded event's timestamp.
- **-K `<token>`:** Authenticates the log. The token is a unique string that must match the token used when the log was first created.
- **-E `<employee-name>` / -G `<guest-name>`**: Specifies the name of the employee or guest involved in the event. Names are case-sensitive and consist of alphabetic characters.
- **-A/ -L:** Indicates whether the event is an arrival or departure. This can be for the campus as a whole or a specific room.
- **-R `<room-id>`:** Identifies the room involved in the event. Room IDs are non-negative integers, and each person can only be in one room at a time.
- **`<log>`:** The path to the log file. If the log doesn't exist, it will be created. If it exists, the token must match for the event to be recorded.
- **Batch Processing:** The `-B` option allows multiple commands to be processed from a file sequentially.

Consistency Checks: Ensures that events are logically consistent (e.g., a person cannot leave a room they never entered). If inconsistencies are detected, the command returns an error and does not modify the log.

Error Handling: If any error occurs (e.g., invalid timestamp, conflicting arguments, or incorrect token), the command prints "invalid" and exits without altering the log. In summary, `logappend` manages a secure and consistent log of events, ensuring authorized access and preventing tampering through careful validation of each log entry.

logread program

`logread` is a command-line tool used to query the state of the campus based on a log file that records the movements of employees and guests. It allows the user to check who is on campus, track where individuals have been, calculate the time they have spent on campus, and find out which rooms were occupied by specific individuals at the same time.

Arguments:

- **-K (Authentication token):** The token is used to verify the log file's integrity. If the token doesn't match, the command will print "integrity violation" and return an error code 255.
- **-S (State):** Prints the current state of the campus such as, a list of employees and guests on campus as well as room-by-room information, showing who is in each room.
- **-R (Rooms):** Provides a list of rooms that a specified employee or guest has entered in chronological order.

- **-T (Total Time):** Displays the total time spent on campus by a specified employee or guest.
- **-I (Intersecting Rooms):** Lists the rooms that were occupied by all specified employees/guests at the same time during the log's history.
- **-E `<employee-name>` / -G `<guest-name>`**: Specifies the name of the employee or guest involved in the event. Names are case-sensitive and consist of alphabetic characters.
- **log:** The path to the file log used for recording events.

Error Handling:

- If conflicting or invalid command-line arguments are given, it prints "invalid" and exits with code 255.
- If a log file is corrupted or tampered with, it prints "integrity violation" and exits with code 255.

Output:

- The output for `-S` is multi-line, with lists separated by commas and sorted.
- For `-R` and `-T`, the output is straightforward, either a list of room IDs or the total time as an integer.

Optional Feature (`-I`): To be implemented.

Prints the rooms, as a comma-separated list of room IDs, that were occupied by all the specified employees and guests at the same time over the complete history of the campus. Room IDs are printed in ascending numerical order. If a specified employee or guest does not appear in the campus, it is ignored. If no room ever contains all of the specified persons, then nothing is printed .

Return Codes:

`0` for successful execution.
`255`for any errors or integrity violations.

Security properties:

1. **Token Verification**: Ensures that only authorized users (e.g., guards) can access the system, log entries, and perform specific actions, preventing unauthorized access.
2. **Encryption**: Protects the confidentiality of sensitive data, such as log entries and user information, both in storage and during transmission, ensuring that even if data is intercepted, it remains unreadable to attackers.

3. **Checksum in Buffer**: Verifies the integrity of log entries before appending them to the main file, preventing tampering and guarding against attacks like man-in-the-middle by ensuring data has not been altered.
4. **Rate Limiting**: Prevents database overflow and protects against denial-of-service (DoS) attacks by limiting the number of requests a user can make in a given time frame, ensuring system stability and security.
5. **Multi-Factor Authentication (MFA):** Consider implementing MFA for critical operations, such as reading the log file, to add an additional layer of security. Even if the token is compromised, the attacker must bypass MFA to succeed.
6. **Timestamp Consistency Check:** Before appending a new entry, the program will verify that the timestamp is greater than the most recent timestamp in the log. If not, it will output "invalid" and exit with an error.

References:

1. https://www.mongodb.com/docs/manual/
2. https://www.mongodb.com/developer/products/mongodb/getting-started-mongodb-cpp/
3. https://www.freecodecamp.org/news/understanding-website-encryption/
4. https://www.youtube.com/watch?v=yPoH5cBJzkk
5. https://www.openssl.org/
6. https://github.com/weidai11/cryptopp