CS 431: Computer Networks and Security

Semester 1, 2024-25

Security Guard Service

Design Document | 4th September 2024

Group - 8			
Email Address	**Name**	**Roll No.**	**Department**
shubham.agrawal@iitgn.ac.in	Shubham Agrawal	22110249	CSE
vraj.shah@iitgn.ac.in	Vraj Shah	22110292	CSE
kishan.ved@iitgn.ac.in	Kishan Ved	22110122	CSE
chandrabhan.patel@iitgn.ac.in	Chandrabhan Patel	22110057	CSE
pranjal.gaur@iitgn.ac.in	Pranjal Gaur	22110201	CSE
parth.govale@iitgn.ac.in	Parth Govale	22110087	CSE
sumeet.sawale@iitgn.ac.in	Sumeet Sawale	22110234	CSE

1. Introduction and Overview:

The goal of this project is to implement a secure log system to track the state of an institute. This includes tracking the entry and exit of employees and guests and storing this information in a log file that guarantees privacy and integrity. The project involves two programs, logappend and logread, both using an authentication token to verify the correctness and prevent unauthorized access.

The system will be designed to ensure that only authorized users with knowledge of the token can modify or read the log. The log will store information about guests and employees in a secure and tamper-proof format.

2. Requirements:

- Logappend: This will append new data (e.g., entries and exits) to the log securely.
- Logread: This will query and read the log, returning the state of the institute.
- Security Model: The logs need to be encrypted and authenticated with an authentication token to ensure privacy and integrity.

3. Log Format:

The log format is structured to support encryption while remaining human-readable when decrypted. It stores the following fields:

- Type: Guest or Employee
- Name: Name of the individual
- Action: Entry or Exit
- Location: Room or building number
- Timestamp: Time of event (Unix timestamp)
- Event ID: Unique identifier for the event
- Sample Log Format (Encrypted)

4. Logappend Program:

Functionality:

- Append data to a log file using fixed command flags as explained below.
- For a particular security guard, create the log file if it does not exist.
- Use regular expressions to validate the data being entered to ensure data consistency and validity.

Command-line Options:

- -T <timestamp>: Specify the event timestamp (seconds since the campus opened).
- -K <token>: Authentication token for log access.
- -E <employee-name>: Employee name (alphabetic, case-sensitive, no spaces).
- -G <guest-name>: Guest name (alphabetic, case-sensitive, no spaces).
- -A: Arrival event.

- -L: Departure event.
- -R <room-id>: Room ID (non-negative integer, no spaces, leading zeros dropped).
- <log>: Path to the log file.
- -B <file>: Batch file of commands to process.

Validation:

- Ensure the timestamp is increasing and within the domain; the clock will be server-side.
- If needed in the future, maintain a domain of rooms on the campus (to limit their number).
- Validate token consistency based on the security guard.

5. Logread Program:

Functionality:

- Read and query the log file upon validation of the token.
- Display the state of the campus based on queries (e.g., total number of guests on the campus).
- Output is in the CLI itself; various options can be chosen using suitable flags.
- Entries will be printed in decreasing order of timestamps.

Command-line Options:

- -K <token>: Authentication token for log access.
- -S: Print the current state of the campus.
- -R (-E <name> | -G <name>): List rooms visited by the specified person.
- -T (-E <name> | -G <name>): Total time spent by the specified person.
- -I (-E <name> | -G <name>) [(-E <name> | -G <name>) ...]: Rooms occupied simultaneously by all specified persons (optional).
- <log>: Path to the log file.

Validation:

- Validate the token and based on this, access the respective log file.
- A pagination mechanism will be used to divide the log into segments.
- Check for file integrity and consistency.
- Ensure valid command-line arguments.

6. Security Model:

Token-Based Access:

- Ensure only authorized users can access or modify the log.
- Each log file is associated with a unique token.

File Integrity:

- Ensure the log file is in a valid state (e.g., no invalid transitions).
- Detect unauthorized modifications or corruption.

Action Validity:

- Validate logical actions (e.g., no room exits before entry).
- Flag and reject invalid actions.

Mitigation:

- Implement a delay between entries to reduce brute-force attack risk.
- Limit access to the log file.

7. Programming Aspects:

CLI Application:

- Use C++ for backend development.
- Utilize relevant C++ libraries.
- Employ a Makefile for building the final ELF.

Non-Volatile Database:

- Use JSON / CSV / XML for log data storage and management.

Testing & Batch Files:

- Utilize shell scripts for testing and batch file operations.

8. Build Script:

Makefile:

- Define build instructions for compiling and linking the application.

Automation of Compilation & Testing:

- Automate the build and testing process to ensure consistency and efficiency.

9. Test Application:

Unit Testing:

- Test individual components for correct functionality.

Integration Testing:

- Test the integration of components and overall system behavior.

Security Testing:

- Test for security vulnerabilities and robustness against attacks.

10. Hosting the Application:

- Use multiple threads to handle several endpoints of the application.
- Implement a class-based structure, handling objects for each endpoint.

11. Modules and Work Split:

Our team consists of seven members. This section lists several tasks and milestones. However, this task allocation is flexible, and most of the work will be done collaboratively as needed. The assignments primarily serve as a reference to outline key project milestones.

11.1 Encryption and Authentication Module

- Assigned to: Shubham Agrawal (22110249)
- Implement encryption and decryption techniques, and hashing for log integrity.
- Manage the token-based authentication system and secure command-line token handling.

11.2 Logappend Implementation

- Assigned to: Vraj Shah (22110292)
- Implement log appending with decryption, modification, re-encryption.
- Handle command-line inputs for log entries (name, event type, room, token).

11.3 Logread Implementation

- Assigned to: Kishan Ved (22110122)
- Implement log reading and querying with decryption and hash verification to prevent tampering.
- Handle user queries such as current occupants, list of visitors, and optional features.

11.4 Testing and Validation

- Assigned to: Chandrabhan Patel (22110057)
- Create and run test scenarios for log integrity, authentication, and optional features.
- Develop the Makefile for building and testing the project.

11.5 Log Format and Integrity Check Design

- Assigned to: Pranjal Gaur (22110201)
- Design log format supporting encryption, and ensure valid transitions (no exit before entry).
- Implement file integrity checks during log reads and writes.

11.6 Batch Processing and Command-Line Integration

- Assigned to: Parth Govale (22110087)
- Implement batch processing for multiple log appends/reads and validate command-line inputs.
- Design and integrate command-line interface flags for `logappend` and `logread`.

11.7 Project Documentation and Reporting

- Assigned to: Sumeet Sawale (22110234)
- Document the design, usage instructions, and test results.
- Compile the final report and ensure submission meets project requirements.

12. Conclusion:

This design should ensure a secure, tamper-proof logging system for tracking guests and employees. The system's modular design hopes to allow for efficient implementation and future extensions.

13. References:

1) We used ChatGPT to get a basic idea and the outline of the project
2) Lecture Slides of CS:431 Computer Networks and Security
3) Project description mentioned in the GitHub repo