Alexander Witwer, Tristan Besser

**Secure Database-as-a-Service System Report**

**1. Introduction**

Cloud-based database services have become essential, providing scalable and reliable solutions to everyday problems, this assignment is no different. However, security concerns arise due to potential unauthorized access by malicious insiders and this happens more often than not. This project aims to design, implement, and evaluate a secure database-as-a-service system, considering a semi-trusted cloud environment. The system focuses on healthcare information, including fields like first name, last name, gender, age, weight, height, and health history.

**2. Basic Setup**

**2.1 SQL Database System Setup**

We have chosen MySQL for the database system. While running the database on a local machine, we simulate the cloud environment.

**2.2 Table Creation and Data Population**

A table is created with the specified fields, and it's populated with at least 100 data items to ensure a robust testing environment.

**3. Security Features**

**3.1 User Authentication**

Implementation:

* We designed a custom authentication system using username/password.
* Passwords are securely stored by hashing before being stored in the database.

Reasoning:

* Custom authentication provides flexibility and enhances security.
* Hashing passwords adds an extra layer of protection, ensuring that even if the database is compromised, user passwords remain secure.

**3.2 Basic Access Control Mechanism**

Implementation:

* Two user groups, H and R, are established.
* Group H has full access, while group R is restricted from accessing first name and last name.

Reasoning:

* Access control ensures that users only see and modify data they are authorized to.
* Differentiating user groups allows for granular control over data access.

**3.3 Basic Query Integrity Protection**

Implementation:

* Single data item integrity is ensured by using cryptographic hashing for each data item.
* Query completeness is addressed by implementing a probabilistic detection mechanism for missing or removed data items.

Reasoning:

* Hashing guarantees the integrity of individual data items.
* Probabilistic detection adds a layer of security against incomplete query results.

**3.4 Basic Data Confidentiality Protection**

Implementation:

* Sensitive attributes, gender, and age, are encrypted.
* Protection measures are in place to prevent statistical leakage.

Reasoning:

* Encryption safeguards sensitive information.
* Statistical leakage prevention maintains data privacy.

**4. System Architecture**

A diagram of a service

Description automatically generated

**5. Team Contribution**

Each team member made significant contributions to different aspects of the project. Refer to the GitHub commit history for a detailed breakdown of individual contributions.

**6. Limitations**

**6.1 Security/Privacy-Related Limitations**

* **Encryption Overhead:** Encrypting sensitive attributes may introduce a performance overhead, impacting query response times.
* **Probabilistic Detection:** While effective, probabilistic detection of query completeness has limitations in certain edge cases.
* **Key Management:** The system assumes secure key management practices, and any compromise in this area could lead to vulnerabilities.

**7. Conclusion**

This project successfully addresses security concerns in a cloud-based database-as-a-service system. The implemented features ensure user authentication, access control, query integrity, and data confidentiality. The team acknowledges limitations and emphasizes continuous improvement for enhanced security.