Security and Data Integrity Analysis

Thomas Bioren

Christopher Kim

TABLE OF CONTENTS

EXECUTIVE SUMMARY

This document describes the security and data integrity problems the MyID90 database faces and how we intend to resolve them.

In short, our users have their names, IDs, dependent relationships, and passwords stored in our database, and it is possible for an attacker to obtain them through SQL injection, connecting directly to the database to run arbitrary code, or social engineering.

PRIVACY ANALYSIS

Sensitive information related to a Person entity includes the person's name, ID, password, and relationship to other persons (as a dependent or vice versa). Users can view the personal information of their dependents. Database administrators can view all users' personal information. This information should remain inaccessible to other users because there is no legitimate reason for one user to know the personal information of other users; allowing such access could lead to privacy concerns.

The only sensitive information related to the Airline table is which employees are employed at which airline. A user can see only what airline they are listed as an employee under. Dependents can see what airline their employee is listed under as well. This information should remain private because an airline could potentially forbid their employees from using this service and could punish users if they had access to this data.

The ID of the person who submitted a load request is only available to the person who made the submission. To all other users, the person submitting the request is anonymous. This can potentially stop favoritism and refusing to fill load requests for certain users.

The ID of who requested and who fulfilled a load request will never be available to any non-administrative user. The only people who can see this information are database administrators.

All information related to the Flight relation is available to all users. There is no sensitive information in this relation.

SECURITY ANALYSIS

Database applications have many vulnerabilities. This section discusses three potential exploits and our approach to mitigating them.

SQL Injection: A potential attacker can inject malicious SQL scripts into our database to delete data or return sensitive information. SQL injection could potentially reveal sensitive information about users such as their passwords or relationships to other users. A breach of this kind can cause distrust among employees and their dependents. Furthermore, data breaches could cause airlines to ban their employees from using our service. Our solution to this potential issue is to use prepared statements and stored procedures wherever possible. In a situation where a user input string is required, we will sanitize the string in Java before sending the query to the database. This will ensure that SQL injection becomes impossible.

Social Engineering: A system's security is only as secure as the least secure person using it. A motivated attacker can use social engineering techniques to convince an administrator to give up sensitive information such as passwords. Just as the previous attack vector, if information is easily exposed, employees and their dependents may lose trust in our product. Additionally, airlines may ban our product from their employees. We will fix this by having strict policies in place for people with administrative privileges. Under no circumstances will an administrator provide a user's password or request a password. An administrator will only provide information to users that is accessible to them through the user's user interface.

Over-Privileged Application Accounts: If an attacker gets the username and password of MyID90's application account, they can wreak havoc on the database. This is a worst-case scenario. All security measures would become irrelevant. Employees and their dependents would have all their information leaked; airlines would be thoroughly upset that all sensitive information about their employees and flights is now public information. To address this, we are taking two separate measures. First, our application account's password is encrypted. This, in theory, will make it impossible for an attacker to get the password. Additionally, we have limited the permissions of the application account to hopefully limit any potential damage that can be done in the event of this worst-case scenario.

ENTITY INTEGRITY ANALYSIS

For our database, we will be utilizing IDs as our primary keys for the Employee, Dependent, and Flight tables. The values for these IDs will be integers. The Airline table will have a 3-letter character variable called ID as its primary key. Airline's primary key will represent the official ICAO airline code of any airline recognized with ICAO. The Destination table will use the official IATA codes for airports as its primary key. Lastly, a composite key of EmployeeID and FlightID will be used as the primary key for tables SubmitsLoadRequest and Updates. EmployeeID and FlightID will be foreign key references to the respective ID of an Employee and Flight.

There are some additional constraints for other keys. In the SubmitsLoadRequest table, the column DateTime cannot have a date that is in the future (*i.e.* a date that is greater than the current date). This principle will also apply to the table Flight and the column LastLoadUpdate.

There are also some constraints for other columns. In the Employee table, an employee cannot have less than 0 tokens. We will use a check constraint on the table to prevent invalid data in these cases.

REFERENTIAL INTEGRITY ANALYSIS

If an employee is removed from the database, then the database will cascade to a null value. A dependent will no longer be able to do anything useful without an employee to refer to.

The database will reject the deletion of an Airline that has any Employees or Flights dependent on it. This is because there cannot be any flights or employees that work for/operated by a null airline.

If a flight that a SubmitsLoadRequest or Updates has a dependency on is being removed, it will cascade to a null value. This is because it is assumed that the flight is being cancelled, thus negating the need for a load request or update to that load request.

When updating the Airline table, Destination table, and Employee table, the database will cascade to all foreign key relationships. This is because when an airline ID is updated, that ID information update needs to be reflected across the database (namely tables Flight and Employee). Similarly, when a Destination ID is updated, that change needs to be reflected in the Flight table, and when an Employee ID is updated, it will need to be reflected in all other foreign key relations in other tables (Dependent, SubmitsLoadsRequest, and Updates)

BUSINESS RULE INTEGRITY ANALYSIS

When a flight's date/time of departure has passed, no more load requests or updates may be made. This is because a past flight's load is irrelevant. This will be implemented with the stored procedures to create and update load requests.

REFERENCES

None

APPENDIX

Airline loads are highly volatile and may be negative at times. We are aware of this and chose to not have any restrictions on load values.

INDEX

GLOSSARY

ICAO: International Civil Aviation Organization

IATA: International Air Transport Association