Building and Securing REST API Report

Intro to API security

An API, by definition, is a set of rules or protocols that enables software applications to communicate with each other to exchange data, features, and functionality. They allow communication between systems and users. They are, however, often prone to attacks. Attacks include:

- Data leaking: exposing data in APIs
- Rate abuse: Getting your api used too much, too fas,t which can cause issues

API security is all about regulating who can access what data to avoid leaks and other harmful events. In order to protect your api, you can implement the following:

- Authentication: letting the owners know who is accessing the api. Can be done with Oauth, JWT, etc….
- Authorization: having restrictions on what can be accessed
- Secure channels of data transport (https)
- Monitoring user activity

Documentation of endpoints

Users Endpoints

List All Users

- **Endpoint:** GET /users
- **Description:** Retrieve all users
- **Error Codes:**
    - 401 Unauthorized
    - 500 Internal Server Error

Get User by ID

- **Endpoint:** GET /users/{user_id}
- **Description:** Retrieve a single user by ID
- **Error Codes:**
  - 404 Not Found
  - 500 Internal Server Error

Create New User

- **Endpoint:** POST /users
- **Description:** Create a new user with a mobile number
- **Error Codes:**
  - 400 Bad Request
  - 500 Internal Server Error

Update User

- **Endpoint:** PUT /users/{user_id}
- **Description:** Update a user’s information
- **Error Codes:**
  - 401 Unauthorized
  - 400 Bad Request
  - 404 Not Found
  - 500 Internal Server Error

Delete User

- **Endpoint:** DELETE /users/{user_id}
- **Description:** Delete a user by ID
- **Error Codes:**
    - 404 Not Found
    - 500 Internal Server Error

Transactions Endpoints

List All Transactions

- **Endpoint:** GET /transactions
- **Description:** Retrieve all transactions
- **Error Codes:**
    - 401 Unauthorized
    - 500 Internal Server Error

Get Transaction by ID

- **Endpoint:** GET /transactions/{transaction_id}
- **Description:** Retrieve a single transaction by ID
- **Error Codes:**
    - 401 Unauthorized
    - 404 Not Found

    - 500 Internal Server Error

Create New Transaction

- **Endpoint:** POST /transactions
- **Description:** Create a new transaction record
- **Error Codes:**
    - 401 Unauthorized
    - 400 Bad Request
    - 500 Internal Server Error

Update Transaction

- **Endpoint:** PUT /transactions/{transaction_id}
- **Description:** Update transaction details
- **Error Codes:**
    - 401 Unauthorized
    - 400 Bad Request
    - 404 Not Found
    - 500 Internal Server Error

Delete Transaction

- **Endpoint:** DELETE /transactions/{transaction_id}
- **Description:** Delete a transaction by ID

- **Error Codes:**
    - 401 Unauthorized
    - 404 Not Found
    - 500 Internal Server Error

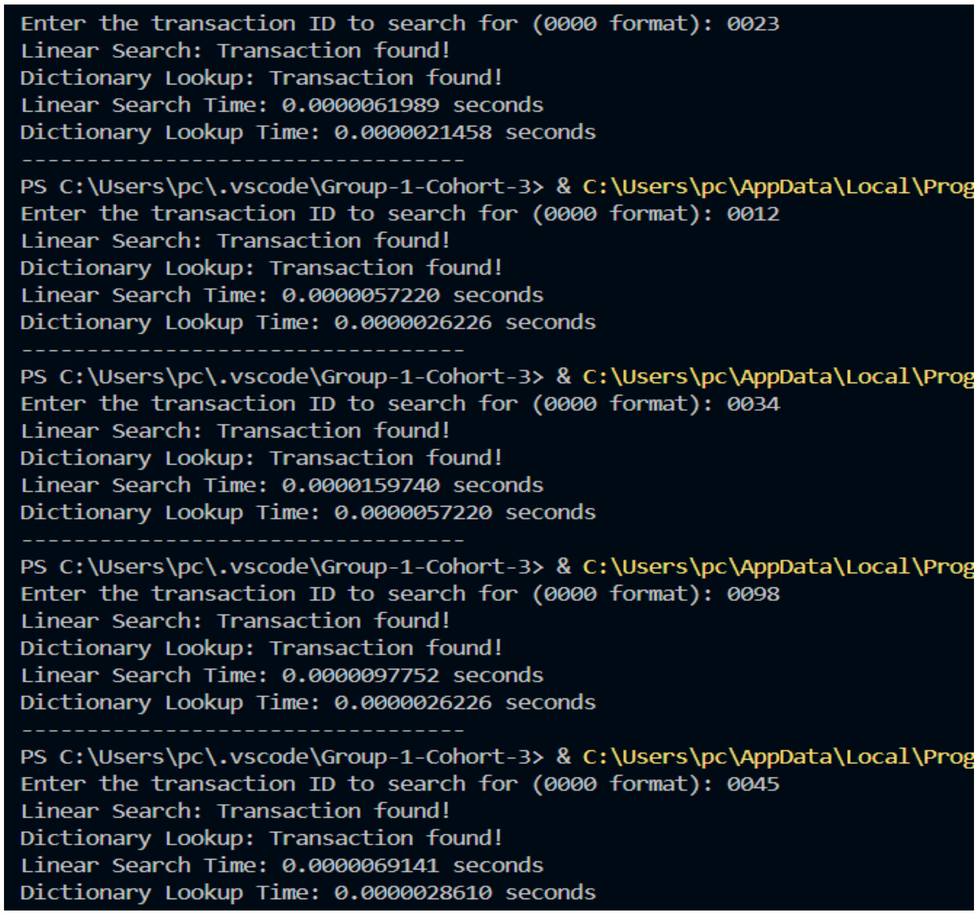
Result of DSA comparison

We implemented two different search methods to locate transactions by ID: Linear search and Dictionary lookup.

Linear Search: Iterates through each transaction, with a time complexity of O(n). This makes it slower the further the item is in a larger set of data.

Dictionary Lookup: Uses an inbuilt hash function to quickly associate values, with a time complexity of O(1). This means it's super fast consistently.

After the analysis of both methods, it is pretty clear which method was faster when we compared them for 20 records. Below is a screenshot of 5 of 20, just as a confirmation.



```
Enter the transaction ID to search for (0000 format): 0023
Linear Search: Transaction found!
Dictionary Lookup: Transaction found!
Linear Search Time: 0.0000061989 seconds
Dictionary Lookup Time: 0.0000021458 seconds
----------------------------------
PS C:\Users\pc\.vscode\Group-1-Cohort-3> & C:\Users\pc\AppData\Local\Prog
Enter the transaction ID to search for (0000 format): 0012
Linear Search: Transaction found!
Dictionary Lookup: Transaction found!
Linear Search Time: 0.0000057220 seconds
Dictionary Lookup Time: 0.0000026226 seconds
----------------------------------
PS C:\Users\pc\.vscode\Group-1-Cohort-3> & C:\Users\pc\AppData\Local\Prog
Enter the transaction ID to search for (0000 format): 0034
Linear Search: Transaction found!
Dictionary Lookup: Transaction found!
Linear Search Time: 0.0000159740 seconds
Dictionary Lookup Time: 0.0000057220 seconds
----------------------------------
PS C:\Users\pc\.vscode\Group-1-Cohort-3> & C:\Users\pc\AppData\Local\Prog
Enter the transaction ID to search for (0000 format): 0098
Linear Search: Transaction found!
Dictionary Lookup: Transaction found!
Linear Search Time: 0.0000097752 seconds
Dictionary Lookup Time: 0.0000026226 seconds
----------------------------------
PS C:\Users\pc\.vscode\Group-1-Cohort-3> & C:\Users\pc\AppData\Local\Prog
Enter the transaction ID to search for (0000 format): 0045
Linear Search: Transaction found!
Dictionary Lookup: Transaction found!
Linear Search Time: 0.0000069141 seconds
Dictionary Lookup Time: 0.0000028610 seconds
```

Reflection on basic auth limitation

Basic auth is weak because it uses base64, which can easily be reversed or decoded. Also, with every request, credentials are sent and can be intercepted. Once they are intercepted, whoever intercepted them can just use them over and over again since they do not expire or change.

I would recommend more secure authentication like OAuth because it uses limited-time tokens, which can change if intercepted, and it doesn't require credentials to be sent, similar to JWT.