1. Introduction to API Security

APIs are powerful tools that allow software applications to communicate and exchange data. However, because they often expose sensitive information such as financial transactions or user details, securing them is essential.

In our MoMo SMS Data Processing system, API security ensures that only authorized users can access or manipulate transaction data. Without proper security, attackers could intercept, modify, or misuse data.

For this assignment, we implemented **Basic Authentication**, where the client must send a username and password with every request. This provides a first layer of security, but it has significant weaknesses:

- Credentials are sent with every request and can be intercepted if not encrypted.
- If reused across multiple services, one stolen password can compromise all systems.
- Passwords do not expire and must be manually changed.

Although Basic Auth is easy to implement, stronger authentication methods (JWT or OAuth2) are better for real-world systems.

2. API Documentation

Our API was implemented in plain Python using `http.server`. It supports the following CRUD endpoints:

GET /transactions

- **Description:** Returns all transactions.

Request Example:

```
curl -u admin:password http://localhost:8000/transactions
```

Response Example:

```
[
 {
  "id": 1,
  "sender": "Alice",
  "receiver": "Bob",
  "amount": 5000,
  "timestamp": "2024-06-18 14:08:05"
 }
]
```

- **Error Codes:**
    - 401 Unauthorized – Invalid credentials
    - 500 Internal Server Error – Unexpected issue

GET /transactions/{id}

- **Description:** Returns a single transaction by ID.

Request Example:

```
curl -u admin:password http://localhost:8000/transactions/1
```

Response Example:

```
{
 "id": 1,
 "sender": "Alice",
 "receiver": "Bob",
 "amount": 5000,
 "timestamp": "2024-06-18 14:08:05"
}
```

- **Error Codes:**
    - 404 Not Found – No transaction with that ID

POST /transactions

- **Description:** Creates a new transaction.

Request Example:

```
curl -u admin:password -X POST http://localhost:8000/transactions \
    -H "Content-Type: application/json" \
    -d '{"sender":"Frank","receiver":"Grace","amount":3000,"timestamp":"2024-06-20 10:30:00"}'
```

Response Example:

```
{"message": "Transaction added successfully"}
```

PUT /transactions/{id}

- **Description:** Updates an existing transaction.

Request Example:

```
curl -u admin:password -X PUT http://localhost:8000/transactions/1 \
    -H "Content-Type: application/json" \
    -d '{"amount":5500}'
```

Response Example:

```
{"message": "Transaction updated successfully"}
```

DELETE /transactions/{id}

- **Description:** Deletes a transaction by ID.

Request Example:

```
curl -u admin:password -X DELETE http://localhost:8000/transactions/1
```

Response Example:

```
{"message": "Transaction deleted successfully"}
```

3. Results of DSA Comparison

We tested two ways of searching for transactions by ID:

1. **Linear Search (List Scan)**
    - Steps through every record until it finds the match.
    - Time complexity: **O(n)**.
    - Example: If there are 1000 transactions, worst case is checking all 1000.
2. **Dictionary Lookup (Hash Map)**
    - Stores transactions in a dictionary `{id → transaction}`.
    - Time complexity: **O(1)** average.
    - Example: Even with 1000 transactions, it jumps straight to the result in 1 step.

Test Results with 20 Records:

- Linear search took ~0.002s.
- Dictionary lookup took ~0.00001s.

Conclusion: Dictionary lookup is far more efficient. For larger datasets (10,000+ records), this difference becomes critical. An even better structure for range queries could be a **Binary Search Tree (BST)** or **B-Tree**, which supports fast lookups and sorted order.

4. Reflection on Basic Auth Limitations

While Basic Authentication provided a simple way to secure our endpoints, it is not sufficient for real-world applications. Its key limitations include:

- Passwords are static and reused for every request.
- If intercepted, attackers can replay them indefinitely.
- No built-in mechanism for expiration, roles, or token revocation.

Better Alternatives:

- **JWT (JSON Web Tokens):** Tokens expire, can embed user roles, and are signed to prevent tampering.
- **OAuth2:** Industry-standard used by Google, Facebook, and payment providers. Provides token refresh and delegated permissions.

Reflection:
For our MoMo SMS system, we used Basic Auth to meet the assignment requirements. However, in a production environment handling financial transactions, JWT or OAuth2 would be essential to ensure data integrity, user privacy, and compliance with security standards.