Department of Computer Science and Engineering

University of Dhaka

Dhaka-1000
Professional Masters in Information and Cyber Security

Subject: Software Security
Code: 803

Assignment-1
Threat Model of Online Banking System

Name: Abu Syeed Sajid Ahmed
Roll: 30023

1/ a)
Describe, at a high level, the design of the system, including: the kinds of data it would handle or store, any servers or services it would contain, and any network devices it would contain.

Ans:

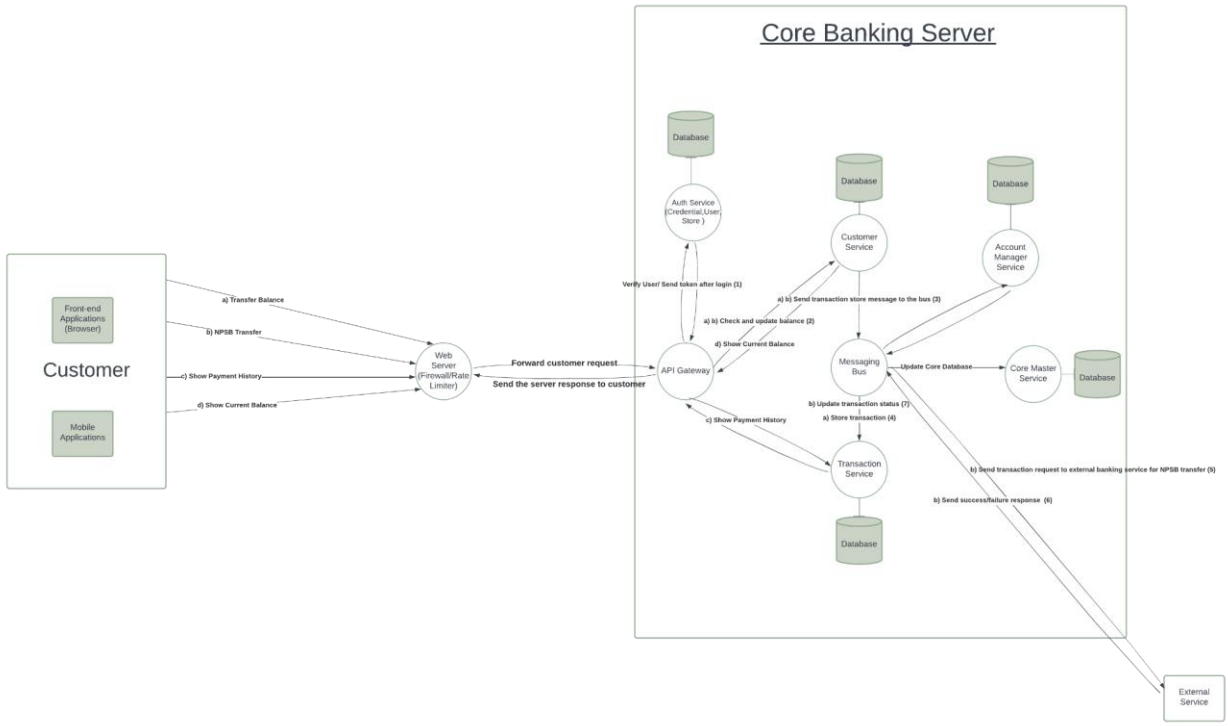

Fig-1: HLD of Online Banking System using Microservice Architecture

In figure-1, There are mainly 3 core components in the System.These are:

1) Client Side (Mobile App, Web App)
2) Backend/Server Side
3) External Services (Other banking services, cloudflare etc.)

- **Client Side:**

This layer is mainly used by customers through mobile or web applications to perform various actions like “Show Balance”, “Show Transaction History”, “Balance Transfer”, “NPSB Transfer” etc. Customers send HTTP

requests from this layer to the backend server to perform these actions.

- **Backend/Server Side:**

This layer accepts requests from client side applications to perform several actions like “Show Balance”, “Show Transaction History”, “Balance Transfer”, “NPSB Transfer”.
This layer is designed using microservice architecture. According to the microservice architecture, the backend server is built using many small services. Each service is attached with a database server. These servers communicate directly with each other while fetching data from each other. But when a service wants to send some data after some events, it will have to send the data to the messaging bus like Kafka/RabbitMQ first, then the messaging bus will send the data to the desired service. In this way, reliability of the data transfer is ensured.

- **External Service:**

In our system, there are many kinds of external services. One is Cloudflare which is acting as a Web server. This service protects our core banking system from DDos attacks, filtering malicious packets etc.
Another one is other banking services for executing NPSB transfers.Using external services one customer of a bank can transfer balance to the customer of another banking service.

1/ b) List the assets that the system contains that the bank that builds and runs the system would want to protect.

Ans:

A bank building and running a financial system would prioritize protecting several key asset categories:

1. Customer Data:

- This includes personal information like names, addresses, Social Security numbers, and contact details.
- Account details including transaction history, balances, and holdings.

2. Financial Data:

- All information related to deposits, loans, transfers, and other

financial transactions.
- This also includes sensitive data like credit scores and investment strategies.

3. System Security Infrastructure:

- Firewalls, intrusion detection systems, and encryption technologies that safeguard the entire system.
- Access controls and user authentication mechanisms to prevent unauthorized entry.

4. Core Banking Applications:

- Software programs that manage accounts, process transactions, and perform other critical banking functions.
- Backup and disaster recovery plans to ensure system continuity in case of outages.

5. Intellectual Property:

- The bank's proprietary software, algorithms, and business processes used in the financial system.

Protecting these assets is crucial for the bank to maintain customer trust, ensure financial stability, and comply with regulations

1/ c) Produce a data flow diagram of the system's design that contains users, data stores, active components (e.g., servers or processes), and the data that flows between them, as well as any trust boundaries.

Ans:

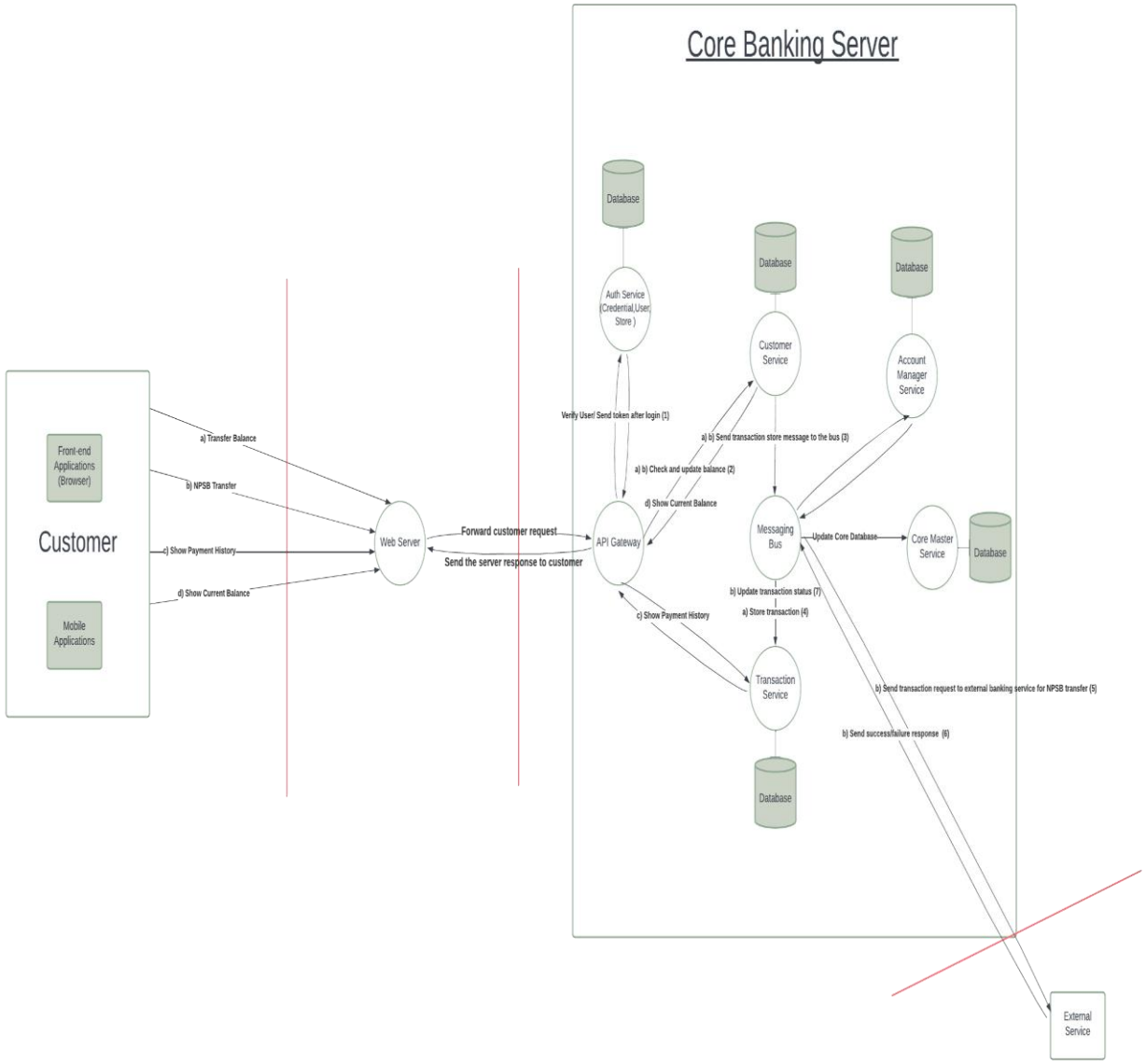

Fig-2: DFD of Online Banking System using Microservice Architecture

1/ d) Briefly describe the trust boundaries of the system.

Ans: In Fig-2, trust boundaries are marked as red straight lines.
There are 3 red lines:

- One is between client side application of the customer and Web Server.

Trust boundaries between client-side applications (such as web browsers) and web servers are essential for several reasons:

1) **Security**: Trust boundaries help protect sensitive data and prevent unauthorized access to server resources. Without proper boundaries, malicious actors could exploit vulnerabilities in client-side applications to access or manipulate server data.
2) **Privacy**: Trust boundaries ensure that user data is handled securely and in accordance with privacy regulations. By restricting access to sensitive information to only authorized entities (such as the server), trust boundaries help maintain user privacy.
3) **Integrity**: Boundaries ensure the integrity of data transmitted between the client and server. By enforcing strict controls over data transmission and processing, trust boundaries help prevent data tampering or manipulation.
4) **Authentication**: Trust boundaries facilitate proper authentication

mechanisms, ensuring that only authenticated users or applications can access sensitive server resources. This helps prevent unauthorized access and misuse of server functionalities.

5) **Resource Management**: By delineating the responsibilities and capabilities of client-side applications and web servers, trust boundaries help optimize resource management and ensure efficient use of computing resources.

- Second one is between Web Server and API Gateway.

Trust boundaries between a web server and an API gateway are crucial for several reasons:

6) **Security Isolation**: API gateways often serve as a central entry point for client requests, handling authentication, authorization, and routing to backend services. Trust boundaries help ensure that the API gateway can authenticate and authorize requests securely before forwarding them to backend servers. This isolation prevents unauthorized access to backend resources.
7) **Traffic Management**: API gateways manage and distribute incoming traffic to backend services based on various criteria such as load balancing, routing rules, and API versioning. Trust boundaries enable the API gateway to enforce traffic management policies effectively while protecting backend services from overload or denial-of-service attacks.
8) **Protocol Transformation**: API gateways may need to convert incoming requests from one protocol to another (e.g., HTTP to HTTPS) or translate between different data formats (e.g., JSON to XML). Trust boundaries ensure that these transformations are performed securely and accurately, preventing data corruption or injection attacks.
9) **Rate Limiting and Throttling**: API gateways often enforce rate limiting and throttling policies to prevent abuse or misuse of backend services. Trust boundaries help ensure that these policies are enforced consistently and that malicious actors cannot bypass them

to overload backend servers or disrupt service availability.

10) **Monitoring and Logging**: API gateways typically provide logging and monitoring capabilities to track incoming requests, response times, error rates, etc. Trust boundaries enable the gateway to collect and analyze this data securely without exposing sensitive information or compromising system integrity.
11) **Caching**: API gateways may implement caching mechanisms to improve performance by storing frequently accessed data closer to the client. Trust boundaries ensure that cached data is managed securely and invalidated or refreshed as necessary to maintain data consistency and accuracy.

- Third one is between our internal server network and external service.

Trust boundaries between an internal server network and external services are essential for several reasons:

1. **Security Isolation**: External services are often accessed over public networks, such as the internet, which are more susceptible to security threats like hacking, malware, and unauthorized access attempts. Trust boundaries help isolate the internal server network from these external threats, reducing the risk of security breaches and data leaks.
2. **Access Control**: By establishing trust boundaries, organizations can enforce strict access controls to regulate the flow of data between internal servers and external services. This ensures that only authorized entities can interact with external services, mitigating the risk of unauthorized access and data exposure.
3. **Data Protection**: Trust boundaries enable organizations to implement encryption and other security measures to protect sensitive data transmitted between internal servers and external services. This helps safeguard confidential information from interception or tampering by malicious actors during transit over public networks.
4. **Compliance Requirements**: Many industries and regions have

regulations and compliance standards governing the protection of sensitive data, such as GDPR in Europe or HIPAA in the United States. Trust boundaries help organizations comply with these requirements by ensuring that data shared with external services is handled securely and in accordance with relevant regulations.

5. **Service Reliability**: External services may experience downtime, network congestion, or other performance issues that could impact the reliability and availability of internal systems. Trust boundaries allow organizations to implement failover mechanisms, load balancing, and other strategies to maintain service continuity and minimize disruptions caused by external service outages.
6. **Monitoring and Auditing**: Trust boundaries facilitate the implementation of monitoring and auditing mechanisms to track the flow of data between internal servers and external services. This allows organizations to detect and investigate suspicious activities, unauthorized access attempts, or compliance violations, enhancing overall security and accountability.

1/ e) For each mitigation, list each of the threats that your procedures and policies would help prevent, along with a brief justification as to why those mitigations should work.

Ans:
A threat to a system refers to any potential danger or risk that can exploit vulnerabilities in the system's security and compromise its integrity, availability, or confidentiality.

Man-in-the-Middle Attack (Public Internet): Attacker intercepts communication between UI and API Gateway.

- **Mitigation:** Implement HTTPS with strong ciphers and certificate validation on both UI and API Gateway.

SQL Injection Attack (API Gateway, Microservices): Attacker injects malicious code into database queries.

- **Mitigation:** Use prepared statements, sanitize user inputs, and implement input validation.

Unauthorized Access (Authentication Service): Attacker steals credentials or exploits vulnerabilities to gain unauthorized access.

- **Mitigation:** Implement strong password policies, two-factor authentication, and regular security updates.

Privilege Escalation (Authorization Service): Attacker exploits a flaw to gain higher privileges than authorized.

- **Mitigation:** Implement least privilege principle, role-based access control (RBAC), and monitor user activity logs.

Account Takeover (Customer UI, Account Service): Attacker gains control of a customer's account.

- **Mitigation:** Secure session management, implement real-time fraud detection, and offer user education on phishing attempts.

Data Breach (Transaction Service, Core Banking Systems): Attacker steals sensitive financial information.

- **Mitigation:** Encrypt data at rest and in transit, implement data loss prevention (DLP), and monitor system access logs.

Denial-of-Service Attack (Public Internet, API Gateway): Attacker floods the system with requests to render it unavailable.

- **Mitigation:** Implement rate limiting, intrusion detection/prevention systems (IDS/IPS), and have a DDoS mitigation plan.

2/ Taking into account the mitigations, briefly describe the residual risk of the design.

Ans:
The residual risk of a design refers to the level of risk that remains after implementing mitigations or controls to address known threats and vulnerabilities. It represents the risk that an organization is willing to accept or cannot feasibly reduce further through additional measures.
There are some common residual risks that remain integrating with external services. These are listed below:

- **Security breaches:** Even if a third party has strong security practices, there's always a chance of a data breach. This could expose your data or your customers' data.

- **Service disruptions:** If a third-party service goes down, it can disrupt your own operations. This can lead to lost revenue and productivity.

- **Financial instability:** If a third-party goes out of business, it could disrupt your service and force you to find a new provider.

- **Compliance issues:** Changes in regulations or the way a third party handles data could put you out of compliance.

- **Vendor lock-in:** If you become too reliant on a single third-party provider, it can be difficult and expensive to switch to a different one.

By understanding these residual risks, we can take steps to mitigate them. Here are some things you can do:

- Conduct thorough due diligence on third-party providers before integrating with them.
- Negotiate strong contracts that outline security requirements and service level agreements (SLAs).
- Regularly monitor the performance and security of third-party services.
- Have a plan for what to do if a third-party service is disrupted.
- Diversify your vendor portfolio to avoid becoming too reliant on a single provider.