# Threat Modeling Report

**Threat Model Name: Online Banking Application Security Threat Model**

**Owner: Security Analyst**

**Reviewer: Lead Security Architect**

**Contributors:**

* Software Engineer
* DevOps Engineer
* Compliance Officer

**Description:**

This threat model analyzes the security risks associated with the Online Banking Application, focusing on potential vulnerabilities that could be exploited by attackers. The model will evaluate threats related to user authentication, data integrity, and unauthorized access to sensitive financial information.

**Assets:**

* User credentials
* Financial data
* Application Programming Interface (API)
* Customer PII
* Backend servers

**Trust Boundaries:**

* Users and web application: The interface where users interact with the banking application.
* The web application and the backend servers (SQL database): Communication between the web server and the database that stores sensitive financial information.
* The client and network (HTTPS data flows): The data flow over the network between the client device and web server.
* The application and third-party services: External APIs and services, such as third-party payment processors or security monitoring tools.

**Data Flows:**

* User inputs data through application interface.
* Data is transmitted securely via HTTPS to the backend servers.
* Sensitive data is processed and stored in the SQL database.

**Questions:**

* What measures are in place to ensure data integrity across trust boundaries?
* How is sensitive data encrypted at rest and during transmission?
* Are there monitoring mechanisms to detect unauthorized data flows?

**Assumptions:**

* Users will access the application from various devices, including mobile phones and desktop computers.
* The application will utilize standard web technologies (HTML, CSS, JavaScript) and communicate with backend services via APIs.
* Users will have varying levels of security awareness and may fall victim to social engineering attacks.
* The application will store sensitive user data, including personal identification information (PII) and financial records.

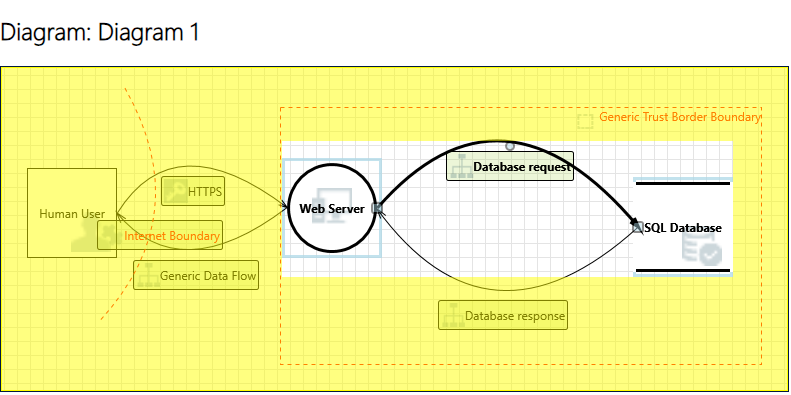
**External Dependencies:**

* Third-party payment processors.
* Security monitoring services for threat detection.

### Threat Model Summary:

|  |  |
| --- | --- |
| Not Started | 0 |
| Not Applicable | 0 |
| Needs Investigation | 0 |
| Mitigation Implemented | 28 |
| Total | 28 |
| Total Migrated | 0 |

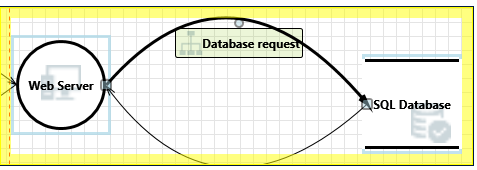
## Diagram: Diagram 1



### Diagram 1 Diagram Summary:

|  |  |
| --- | --- |
| Not Started | 0 |
| Not Applicable | 0 |
| Needs Investigation | 0 |
| Mitigation Implemented | 28 |
| Total | 28 |
| Total Migrated | 0 |

### Interaction: Database request



#### 1. Spoofing of Destination Data Store SQL Database  [State: Mitigation Implemented]  [Priority: Medium]

|  |  |
| --- | --- |
| **Category:** | Spoofing |
| **Description:** | SQL Database may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of SQL Database. Consider using a standard authentication mechanism to identify the destination data store. |
| **Justification:** | Authentication mechanism.  Regular security audit. |

#### 2. Potential SQL Injection Vulnerability for SQL Database  [State: Mitigation Implemented]  [Priority: Medium]

|  |  |
| --- | --- |
| **Category:** | Tampering |
| **Description:** | SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. Any procedure that constructs SQL statements should be reviewed for injection vulnerabilities because SQL Server will execute all syntactically valid queries that it receives. Even parameterized data can be manipulated by a skilled and determined attacker. |
| **Justification:** | Regular vulnerability scanning. |

#### 3. Potential Excessive Resource Consumption for Web Server or SQL Database  [State: Mitigation Implemented]  [Priority: Medium]

|  |  |
| --- | --- |
| **Category:** | Denial Of Service |
| **Description:** | Does Web Server or SQL Database take explicit steps to control resource consumption? Resource consumption attacks can be hard to deal with, and there are times that it makes sense to let the OS do the job. Be careful that your resource requests don't deadlock, and that they do timeout. |
| **Justification:** | Use network access-control lists to control incoming and outgoing traffic |

#### 4. Weak Credential Storage [State: Mitigation Implemented]  [Priority: Medium]

|  |  |
| --- | --- |
| **Category:** | Information Disclosure |
| **Description:** | Credentials held at the server are often disclosed or tampered with and credentials stored on the client are often stolen. For server side, consider storing a salted hash of the credentials instead of storing the credentials themselves. If this is not possible due to business requirements, be sure to encrypt the credentials before storage, using an SDL-approved mechanism. For client side, if storing credentials is required, encrypt them and protect the data store in which they're stored |
| **Justification:** | Encryption.  Implement principle of least privileges. |

#### 5. Risks from Logging [State: Mitigation Implemented]  [Priority: Medium]

|  |  |
| --- | --- |
| **Category:** | Tampering |
| **Description:** | Log readers can come under attack via log files. Consider ways to canonicalize data in all logs. Implement a single reader for the logs, if possible, in order to reduce attack surface area. Be sure to understand and document log file elements which come from untrusted sources. |
| **Justification:** | Use centralized log management.  Use integrity checks and hashing.  Log review and monitoring. |

#### 6. Lower Trusted Subject Updates Logs [State: Mitigation Implemented] [Priority: Low]

|  |  |
| --- | --- |
| **Category:** | Repudiation |
| **Description:** | If you have trust levels, is anyone other outside of the highest trust level allowed to log? Letting everyone write to your logs can lead to repudiation problems. Only allow trusted code to log. |
| **Justification:** | Authentication control.  Use of digital signatures and cryptography. |

#### 7. Data Logs from an Unknown Source [State: Mitigation Implemented] [Priority: Low]

|  |  |
| --- | --- |
| **Category:** | Repudiation |
| **Description:** | Do you accept logs from unknown or weakly authenticated users or systems? Identify and authenticate the source of the logs before accepting them. |
| **Justification:** | Authentication mechanism. |

#### 8. Insufficient Auditing [State: Mitigation Implemented]  [Priority: Low]

|  |  |
| --- | --- |
| **Category:** | Repudiation |
| **Description:** | Does the log capture enough data to understand what happened in the past? Do your logs capture enough data to understand an incident after the fact? Is such capture lightweight enough to be left on all the time? Do you have enough data to deal with repudiation claims? Make sure you log sufficient and appropriate data to handle repudiation claims. You might want to talk to an audit expert as well as a privacy expert about your choice of data. |
| **Justification:** | Regular review and audits of logs.  Standardized log format.  Implement data retention policy. |

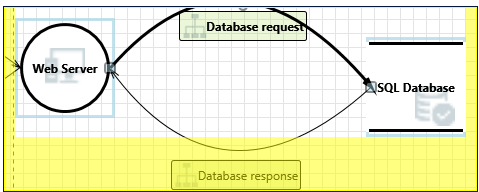
#### 9. Potential Weak Protections for Audit Data [State: Mitigation Implemented]  [Priority: Medium]

|  |  |
| --- | --- |
| **Category:** | Repudiation |
| **Description:** | Consider what happens when the audit mechanism comes under attack, including attempts to destroy the logs, or attack log analysis programs. Ensure access to the log is through a reference monitor, which controls read and write separately. Document what filters, if any, readers can rely on, or writers should expect |
| **Justification:** | Access control mechanism |

#### 10. Authorization Bypass [State: Mitigation Implemented] [Priority: Medium]

|  |  |
| --- | --- |
| **Category:** | Information Disclosure |
| **Description:** | Can you access SQL Database and bypass the permissions for the object? For example by editing the files directly with a hex editor, or reaching it via filesharing? Ensure that your program is the only one that can access the data, and that all other subjects have to use your interface. |
| **Justification:** | Access control implemented.  Encryption of sensitive data. |

### Interaction: Database response



#### 11. Spoofing of Source Data Store SQL Database [State: Mitigation Implemented]  [Priority: Low]

|  |  |
| --- | --- |
| **Category:** | Spoofing |
| **Description:** | SQL Database may be spoofed by an attacker and this may lead to incorrect data delivered to Web Server. Consider using a standard authentication mechanism to identify the source data store. |
| **Justification:** | Authentication mechanism |

#### 12. Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Tampering |
| **Description:** | The web server 'Web Server' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input. |
| **Justification:** | Validating input to prevent the processing of malicious payloads.  Use SSL/TLS for secure transmission |

#### 13. Persistent Cross Site Scripting  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Tampering |
| **Description:** | The web server 'Web Server' could be a subject to a persistent cross-site scripting attack because it does not sanitize data store 'SQL Database' inputs and output. |
| **Justification:** | Sanitize user inputs.  Encode output data.  Regular security audit |

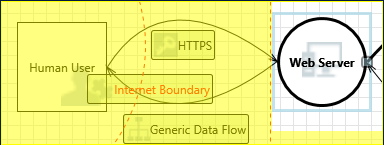
#### 14. Weak Access Control for a Resource  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Information Disclosure |
| **Description:** | Improper data protection of SQL Database can allow an attacker to read information not intended for disclosure. Review authorization settings. |
| **Justification:** | Implement role based access control (RBAC).  Apply principle of least privileges.  Encrypt data both at rest and in transit. |

#### 15. Risks from Logging [State: Mitigation Implemented]  [Priority: Low]

|  |  |
| --- | --- |
| **Category:** | Tampering |
| **Description:** | Log readers can come under attack via log files. Consider ways to canonicalize data in all logs. Implement a single reader for the logs, if possible, in order to reduce attack surface area. Be sure to understand and document log file elements which come from untrusted sources. |
| **Justification:** | Centralized log management implemented.  Use integrity checks and hashing.  Log review and monitoring. |

### Interaction: Generic Data Flow



#### 16. Spoofing of the Human User External Destination Entity  [State: Mitigation Implemented]  [Priority: Low]

|  |  |
| --- | --- |
| **Category:** | Spoofing |
| **Description:** | Human User may be spoofed by an attacker and this may lead to data being sent to the attacker's target instead of Human User. Consider using a standard authentication mechanism to identify the external entity. |
| **Justification:** | Multi-factor authentication.  Security awareness practices. |

#### 17. External Entity Human User Potentially Denies Receiving Data  [State: Mitigation Implemented]  [Priority: Low]

|  |  |
| --- | --- |
| **Category:** | Repudiation |
| **Description:** | Human User claims that it did not receive data from a process on the other side of the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data. |
| **Justification:** | Digital signature.  Regular log review.  User awareness education. |

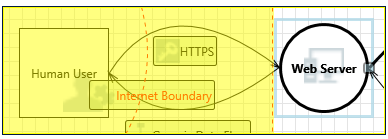
#### 18. Data Flow Generic Data Flow Is Potentially Interrupted  [State: Mitigation Implemented]  [Priority: Medium]

|  |  |
| --- | --- |
| **Category:** | Denial Of Service |
| **Description:** | An external agent interrupts data flowing across a trust boundary in either direction. |
| **Justification:** | Data encryption.  Develop incident response plan. |

#### 19. Authenticated Data Flow Compromised [State: Mitigation Implemented]  [Priority: Low]

|  |  |
| --- | --- |
| **Category:** | Tampering |
| **Description:** | An attacker can read or modify data transmitted over an authenticated dataflow. |
| **Justification:** | Encrypt data.  Use HTTPS for secure communication. |

### Interaction: HTTPS



#### 20. Spoofing the Human User External Entity  [State: Mitigation Implemented]  [Priority: Medium]

|  |  |
| --- | --- |
| **Category:** | Spoofing |
| **Description:** | Human User may be spoofed by an attacker and this may lead to unauthorized access to Web Server. Consider using a standard authentication mechanism to identify the external entity. |
| **Justification:** | Authentication implemented.  HTTPS for secure communication.  Validate user input. |

#### 21. Cross Site Scripting [State: Mitigation Implemented] [Priority: Low]

|  |  |
| --- | --- |
| **Category:** | Tampering |
| **Description:** | The web server 'Web Server' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input. |
| **Justification:** | Authentication implemented |

#### 22. Elevation Using Impersonation  [State: Mitigation Implemented]  [Priority: Medium]

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | Web Server may be able to impersonate the context of Human User in order to gain additional privilege. |
| **Justification:** | Implementing role based access control |

#### 23. Potential Data Repudiation by Web Server  [State: Mitigation Implemented]  [Priority: Low]

|  |  |
| --- | --- |
| **Category:** | Repudiation |
| **Description:** | Web Server claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data. |
| **Justification:** | Digital signature for verification.  Regular audit logs and review. |

#### 24. Potential Process Crash or Stop for Web Server [State: Mitigation Implemented] [Priority: Medium]

|  |  |
| --- | --- |
| **Category:** | Denial Of Service |
| **Description:** | Web Server crashes, halts, stops or runs slowly; in all cases violating an availability metric. |
| **Justification:** | Monitor network traffic. |

#### 25. Data Flow HTTPS Is Potentially Interrupted  [State: Mitigation Implemented]  [Priority: Medium]

|  |  |
| --- | --- |
| **Category:** | Denial Of Service |
| **Description:** | An external agent interrupts data flowing across a trust boundary in either direction. |
| **Justification:** | Encrypt data while in transit. |

#### 26. Web Server May be Subject to Elevation of Privilege Using Remote Code Execution  [State: Mitigation Implemented]  [Priority: Low]

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | Human User may be able to remotely execute code for Web Server. |
| **Justification:** | Principle of least privileges implemented. |

#### 27. Elevation by Changing the Execution Flow in Web Server [State: Mitigation Implemented]  [Priority: Medium]

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | An attacker may pass data into Web Server in order to change the flow of program execution within Web Server to the attacker's choosing. |
| **Justification:** | Input validation and sensitization.  Secure coding practice. |

#### 28. Cross Site Request Forgery [State: Mitigation Implemented] [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | Cross-site request forgery (CSRF or XSRF) is a type of attack in which an attacker forces a user's browser to make a forged request to a vulnerable site by exploiting an existing trust relationship between the browser and the vulnerable web site. In a simple scenario, a user is logged in to web site A using a cookie as a credential. The other browses to web site B. Web site B returns a page with a hidden form that posts to web site A. Since the browser will carry the user's cookie to web site A, web site B now can take any action on web site A, for example, adding an admin to an account. The attack can be used to exploit any requests that the browser automatically authenticates, e.g. by session cookie, integrated authentication, IP whitelisting. The attack can be carried out in many ways such as by luring the victim to a site under control of the attacker, getting the user to click a link in a phishing email, or hacking a reputable web site that the victim will visit. The issue can only be resolved on the server side by requiring that all authenticated state-changing requests include an additional piece of secret payload (canary or CSRF token) which is known only to the legitimate web site and the browser and which is protected in transit through SSL/TLS. See the Forgery Protection property on the flow stencil for a list of mitigations. |
| **Justification:** | Implement Cross-Site Request Forgery (CSRF) tokens.  Implement content security policy. |

**FAQ: Online Banking Application Security**

1. What key security concerns does this threat model address for the online banking application?

This threat model primarily targets security risks related to user authentication, data integrity, and preventing unauthorized access to sensitive financial information. It evaluates potential vulnerabilities across critical trust boundaries, including interactions between users and the web application, communication between the web application and backend servers, network data flow, and the application’s reliance on third-party services. Some of the specific threats addressed include SQL injection, cross-site scripting (XSS), and denial of service (DoS) attacks, weak credential storage and inadequate auditing practices.

2. What are the key assets identified that this security model aims to protect?

The security model aims to protect the following key assets such as user credentials, financial data, the application programming interface (API), customer personally identifiable information (PII), and backend servers. These assets need to be protected from unauthorized access, tampering and disclosure by implementing robust security measures.

3. How does the threat model address the trust boundaries within the online banking system?  
The threat model defines specific trust boundaries, acknowledging that data flows across these areas are potential attack vectors. These boundaries include interactions between users and the web application, communications between the web application and backend servers, network data flows between client devices and the web server (secured with HTTPS), and the application’s integration with third-party services. It identifies risks such as spoofing, tampering, and denial of service attacks at these boundaries and proposes mitigation strategies to ensure the confidentiality, integrity and availability of data during transitions across them.

4. What measures are implemented to protect sensitive data at rest and in transit?

The threat model emphasizes the importance of data encryption both at rest and during transmission. For data in transit, HTTPS is used to secure communications between the client and the web server. For data at rest, strong encryption like AES (Advanced Encryption Standard) is used, Role-Based Access Control (RBAC) is implemented to ensure only authorized users can access the data. This helps to prevent unauthorized access to sensitive data while at rest on in transit.

5. What authentication and authorization measures are implemented to prevent unauthorized access?  
The model incorporates multiple authentication mechanisms, including multi-factor authentication for users, authentication of destination and source data stores to prevent spoofing, and standardized authentication methods to verify external entities. Role-Based Access Control (RBAC) is implemented to restrict access to specific functionalities based on user roles. Additionally, the principle of least privilege is enforced to ensure users only have access to the minimal data or services necessary to perform their tasks. These measures reduce the risk and impact of compromised accounts.

FAQ: Online Banking Application Security

1. What key security concerns does this threat model address for the online banking application?

This threat model primarily targets security risks related to user authentication, data integrity, and preventing unauthorized access to sensitive financial information. It evaluates potential vulnerabilities across critical trust boundaries, including interactions between users and the web application, communication between the web application and backend servers, network data flow, and the application’s reliance on third-party services. Some of the specific threats addressed include SQL injection, cross-site scripting (XSS), and denial of service (DoS) attacks, weak credential storage and inadequate auditing practices.

2. What are the key assets identified that this security model aims to protect?

The security model aims to protect the following key assets such as user credentials, financial data, the application programming interface (API), customer personally identifiable information (PII), and backend servers. These assets needs to be protected from unauthorized access, tampering and disclosure by implementing robust security measures.

3. How does the threat model address the trust boundaries within the online banking system?  
The threat model defines specific trust boundaries, acknowledging that data flows across these areas are potential attack vectors. These boundaries include interactions between users and the web application, communications between the web application and backend servers, network data flows between client devices and the web server (secured with HTTPS), and the application’s integration with third-party services. It identifies risks such as spoofing, tampering, and denial of service attacks at these boundaries and proposes mitigation strategies to ensure the confidentiality, integrity and availability of data during transitions across them.

4. What measures are implemented to protect sensitive data at rest and in transit?

The threat model emphasizes the importance of data encryption both at rest and during transmission. For data in transit, HTTPS is used to secure communications between the client and the web server. For data at rest, strong encryption like AES (Advanced Encryption Standard) is used, Role-Based Access Control (RBAC) is implemented to ensure only authorized users can access the data. This helps to prevent unauthorized access to sensitive data while at rest on in transit.

5. What authentication and authorization measures are implemented to prevent unauthorized access?  
The model incorporates multiple authentication mechanisms, including multi-factor authentication for users, authentication of destination and source data stores to prevent spoofing, and standardized authentication methods to verify external entities. Role-Based Access Control (RBAC) is implemented to restrict access to specific functionalities based on user roles. Additionally, the principle of least privilege is enforced to ensure users only have access to the minimal data or services necessary to perform their tasks. These measures reduce the risk and impact of compromised accounts.