# Security threat model

**Owner**:
Sandra
**Reviewer**:
Thomas
**Contributors**:
**Date Generated**: Wed Aug 21 2024

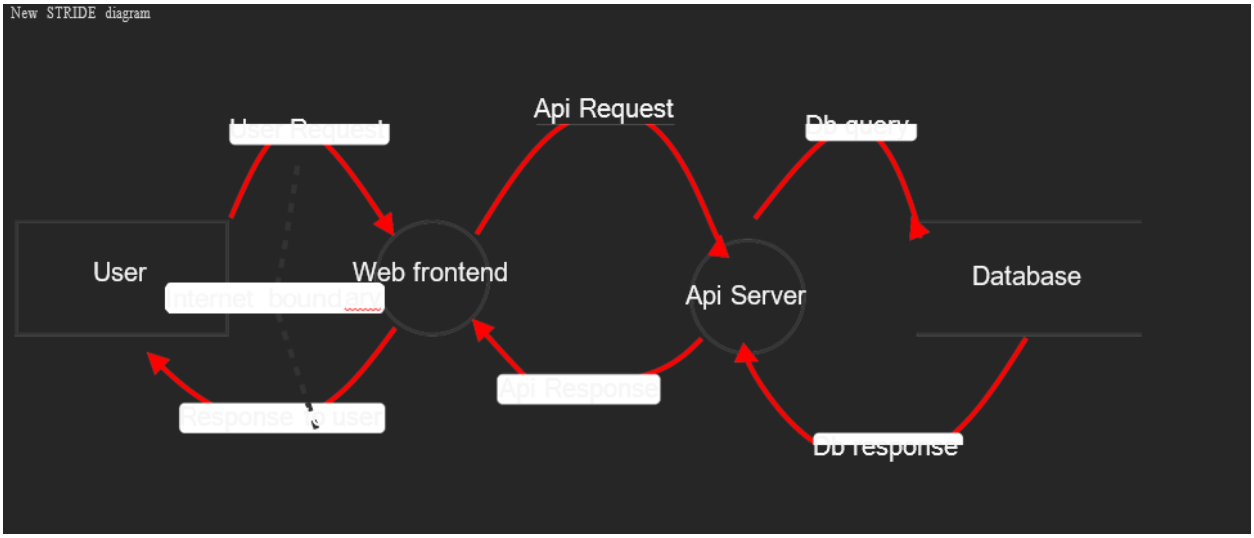
# Executive Summary

High level system description

threat modelling

Summary

| | |
|---|---|
| **Total Threats** | |
| **Total Mitigated** | |
| **Not Mitigated** | |
| **Open / High Priority** | |
| **Open / Medium Priority** | |
| **Open / Low Priority** | |
| **Open / Unknown Priority** | |




# Threat Model Diagram

User (Actor)

| No. | Title | Type | Priority | Status | Description | Mitigations |
|---|---|---|---|---|---|---|
| 12 | Threat model diagram | Repudiation | Medium | Mitigated | Repudiation refers to a security threat where an individual denies performing an action or transaction that has been recorded in a system. Without proper mechanisms to ensure the integrity and traceability of actions, users or attackers could falsely claim that they did not perform a particular action, leading to disputes, fraud, or difficulty in tracking malicious activities. | - **Strong Authentication**: Implement multi-factor authentication (MFA) to ensure that actions are tied to verified users.<br>- **Comprehensive Logging**: Maintain secure, tamper-proof logs with timestamps, IP addresses, and other relevant details.<br>- **Non-Repudiation Controls**: Use digital signatures or cryptographic tokens for definitive action attribution.<br>- **Audit Trails**: Implement audit trails for traceability.<br>- **Log Integrity**: Protect logs from tampering using hashing, encryption, and secure storage. |

**Web Frontend (Process)**

| No. | Title | Type | Priority | Status | Description | Mitigations |
|---|---|---|---|---|---|---|
| 13 | Threat model diagram | Elevation of Privilege | Medium | Mitigated | Elevation of Privilege (EoP) is a security threat where an attacker gains unauthorized access to resources or performs actions they would not normally be allowed to do by exploiting a system vulnerability. This could result in the attacker gaining higher-level privileges, such as administrative rights, potentially compromising the entire system or accessing sensitive data. | - **Secure Coding Practices**: Use input validation and proper error handling to prevent vulnerabilities.<br>- **Least Privilege Principle**: Ensure users and processes have only the necessary privileges.<br>- **Regular Security Audits**: Conduct audits and code reviews to identify vulnerabilities.<br>- **Access Control Mechanisms**: Implement robust access control enforcing separation of duties.<br>- **Patch Management**: Regularly update the OS, application software, and third-party components. |

**API Server (Process)**

| No. | Title | Type | Priority | Status | Description | Mitigations |
|---|---|---|---|---|---|---|
| 14 | Threat model diagram | Spoofing | Medium | Mitigated | Spoofing is a security threat where an attacker masquerades as a legitimate entity or user to gain unauthorized access to sensitive information, systems, or resources. By impersonating a trusted source, the attacker can deceive users or systems into granting access or divulging confidential information. This can occur in various forms, such as IP, email, identity spoofing. | - **Strong Authentication**: Implement MFA to verify user identities and reduce the risk of credential-based spoofing.<br>- **Digital Certificates and SSL/TLS**: Use certificates and encryption to verify identities and ensure secure communication.<br>- **Email Security Measures**: Implement SPF, DKIM, and DMARC to verify emails.<br>- **Network Security Controls**: Use IP filtering, firewalls, and intrusion detection systems to detect and prevent spoofing. |

**Database (Store)**

| No. | Title | Type | Priority | Status | Description | Mitigations |
|---|---|---|---|---|---|---|
| 15 | Threat model diagram | Denial of Service | Medium | Mitigated | Denial of Service (DoS) refers to an attack that aims to make a service, network, or application unavailable to its intended users by overwhelming it with illegitimate requests, consuming system resources, or exploiting vulnerabilities that cause the service to crash or become unresponsive. This can lead to significant downtime, loss of revenue, and damage to the organization's reputation. | - **Rate Limiting**: Control the number of requests a user or IP can make in a timeframe.<br>- **Load Balancing**: Distribute traffic across multiple servers.<br>- **WAF (Web Application Firewall)**: Filter out malicious traffic and protect against common DoS patterns.<br>- **Resource Monitoring**: Continuously monitor resources for signs of exhaustion.<br>- **Timeouts and Retry Mechanisms**: Set appropriate timeouts and retry mechanisms.<br>- **Redundancy and Failover**: Ensure services remain available during an attack. |

**API Response (Data Flow)**

| No. | Title | Type | Priority | Status | Description | Mitigations |
|---|---|---|---|---|---|---|
| 8 | Threat model diagram | Information Disclosure | Medium | Open | Information disclosure refers to the unintended or unauthorized exposure of sensitive information to unauthorized users. This can occur through various means, such as improper data handling, insufficient access controls, or vulnerabilities that allow attackers to access confidential data. | - **Encrypt Sensitive Data**: Use TLS/SSL to encrypt data in transit and encrypt sensitive data at rest.<br>- **Implement Access Controls**: Enforce strict authentication and authorization mechanisms.<br>- **Sanitize Error Messages**: Return generic error messages to users and securely log detailed information.<br>- **Secure Logging Practices**: Avoid logging sensitive data and ensure secure storage.<br>- **Regular Security Audits**: Identify and fix potential information disclosure vulnerabilities. |

**DB Response (Data Flow)**

| No. | Title | Type | Priority | Status | Description | Mitigations |
|---|---|---|---|---|---|---|
| 7 | Threat model diagram | Denial of Service | Medium | Open | Denial of Service (DoS) refers to an attack that aims to make a service, network, or application unavailable to its intended users by overwhelming it with illegitimate requests, consuming system resources, or exploiting vulnerabilities that cause the service to crash or become unresponsive. This can lead to significant downtime, loss of revenue, and damage to the organization's reputation. | - **Rate Limiting**: Control the number of requests a user or IP can make in a timeframe.<br>- **Load Balancing**: Distribute traffic across multiple servers.<br>- **WAF (Web Application Firewall)**: Filter out malicious traffic and protect against common DoS patterns.<br>- **Resource Monitoring**: Continuously monitor resources for signs of exhaustion.<br>- **Timeouts and Retry Mechanisms**: Set appropriate timeouts and retry mechanisms.<br>- **Redundancy and Failover**: Ensure services remain available during an attack. |

**User Request (Data Flow)**

| No. | Title | Type | Priority | Status | Description | Mitigations |
|---|---|---|---|---|---|---|
| 10 | Threat model diagram | Information Disclosure | Medium | Open | Information disclosure refers to the unintended or unauthorized exposure of sensitive information to unauthorized users. This can occur through various means, such as improper data handling, insufficient access controls, or vulnerabilities that allow attackers to access confidential data. | - **Encrypt Sensitive Data**: Use TLS/SSL to encrypt data in transit and encrypt sensitive data at rest.<br>- **Implement Access Controls**: Enforce strict authentication and authorization mechanisms.<br>- **Sanitize Error Messages**: Return generic error messages to users and securely log detailed information.<br>- **Secure Logging Practices**: Avoid logging sensitive data and ensure secure storage.<br>- **Regular Security Audits**: Identify and fix potential information disclosure vulnerabilities. |

**API Request (Data Flow)**

| No. | Title | Type | Priority | Status | Description | Mitigations |
|---|---|---|---|---|---|---|
| 4 | Threat model diagram | Repudiation | Medium | Open | Repudiation refers to a security threat where an individual denies performing an action or transaction that has been recorded in a system. Without proper mechanisms to ensure the integrity and traceability of actions, users or attackers could falsely claim that they did not perform a particular action, leading to disputes, fraud, or difficulty in tracking malicious activities. | - **Strong Authentication**: Implement multi-factor authentication (MFA) to ensure that actions are tied to verified users.<br>- **Comprehensive Logging**: Maintain secure, tamper-proof logs with timestamps, IP addresses, and other relevant details.<br>- **Non-Repudiation Controls**: Use digital signatures or cryptographic tokens for definitive action attribution.<br>- **Audit Trails**: Implement audit trails for traceability.<br>- **Log Integrity**: Protect logs from tampering using hashing, encryption, and secure storage.<br>- **User Notifications**: Send notifications for critical actions as additional evidence. |

In conclusion, Web application security is a complex and critical topic that requires continuous attention and effort. By understanding the fundamentals, common vulnerabilities, and best practices, you can significantly improve the security posture of your web applications. Implement a holistic approach that incorporates secure development practices, regular testing, and ongoing monitoring to safeguard your applications and protect your users' data.

Below is the summary of the OWASP Top Ten

1. **Injection (A01:2021)**
    - **Description**: Attackers exploit vulnerabilities by injecting malicious code or commands into a system (e.g., SQL, NoSQL, OS commands) to manipulate or corrupt the application's behavior.
    - **Mitigations**: Use parameterized queries, input validation, and escape special characters.
2. **Broken Authentication (A02:2021)**
    - **Description**: Flaws in authentication mechanisms allow attackers to compromise passwords, session tokens, or exploit other weaknesses to assume the identity of other users.

    - **Mitigations**: Implement strong authentication practices like MFA, secure session management, and regular audits.
3. **Sensitive Data Exposure (A03:2021)**
    - **Description**: Inadequate protection of sensitive data, such as unencrypted storage or transmission, can lead to exposure to unauthorized parties.
    - **Mitigations**: Encrypt data in transit and at rest, enforce secure access controls, and minimize data exposure.
4. **XML External Entities (XXE) (A04:2021)**
    - **Description**: Attackers exploit vulnerabilities in XML parsers to execute malicious commands, access internal files, or launch denial-of-service attacks.
    - **Mitigations**: Disable external entity processing, use less complex data formats like JSON, and validate input.
5. **Broken Access Control (A05:2021)**
    - **Description**: Inadequate enforcement of access control policies allows unauthorized users to access restricted resources or perform actions beyond their permissions.
    - **Mitigations**: Implement strong access control mechanisms, follow the principle of least privilege, and regularly test access controls.
6. **Security Misconfiguration (A06:2021)**
    - **Description**: Improperly configured security settings, such as default credentials, unnecessary features enabled, or exposed stack traces, increase the risk of compromise.
    - **Mitigations**: Regularly review and update configurations, remove unused services, and enforce secure defaults.
7. **Cross-Site Scripting (XSS) (A07:2021)**
    - **Description**: Attackers inject malicious scripts into web applications, which are then executed in the browser of unsuspecting users, leading to data theft or session hijacking.
    - **Mitigations**: Use input validation, output encoding, and content security policies to mitigate XSS risks.
8. **Insecure Deserialization (A08:2021)**
    - **Description**: Deserializing untrusted data allows attackers to execute arbitrary code, tamper with serialized objects, or launch denial-of-service attacks.
    - **Mitigations**: Avoid deserialization of untrusted data, use secure libraries, and implement integrity checks.
9. **Using Components with Known Vulnerabilities (A09:2021)**
    - **Description**: Relying on outdated or vulnerable third-party libraries, frameworks, or components can expose the application to known exploits.
    - **Mitigations**: Regularly update components, use vulnerability scanning tools, and avoid unmaintained libraries.
10. **Insufficient Logging and Monitoring (A10:2021)**
    - **Description**: Lack of adequate logging and monitoring allows attackers to remain undetected after compromising a system, delaying response efforts and increasing damage.
    - **Mitigations**: Implement comprehensive logging, real-time monitoring, and establish a robust incident response plan.