Web Application Threat Model

Aditya Pangavhane

August 27, 2025

Abstract

In cybersecurity, finding vulnerabilities is not just about scanning or exploiting; it begins with structured steps that build awareness of the target system. After planning the approach and gathering information about the application, the next step is threat modeling. At this stage we shift perspective from attacker to protector: instead of asking "how can we break the system," we ask "where could the system break, and how can we defend it."

Threat modeling works like mapping the weak points of a house before a storm. We examine how attackers might attempt entry, what they could achieve, and the possible business impact if they succeed. This report studies established methodologies such as STRIDE, PASTA, OWASP threat categories, and NIST risk guidance. Each framework provides a structured way to recognize potential weaknesses in a web application before attackers exploit them.

We describe threats in everyday terms—such as using stolen credentials to impersonate a user or injecting malicious input to expose private data—and connect them directly to business risks like loss of user trust, downtime, or regulatory penalties. For each identified threat, we highlight realistic security controls including stronger authentication, data validation, encryption, and active monitoring. The purpose of this report is not only technical defense but also to provide non-technical stakeholders with clarity on why these risks matter.

1 System Overview

1.1 Architecture and Components

Figure **??** shows the high-level architecture of our web application:

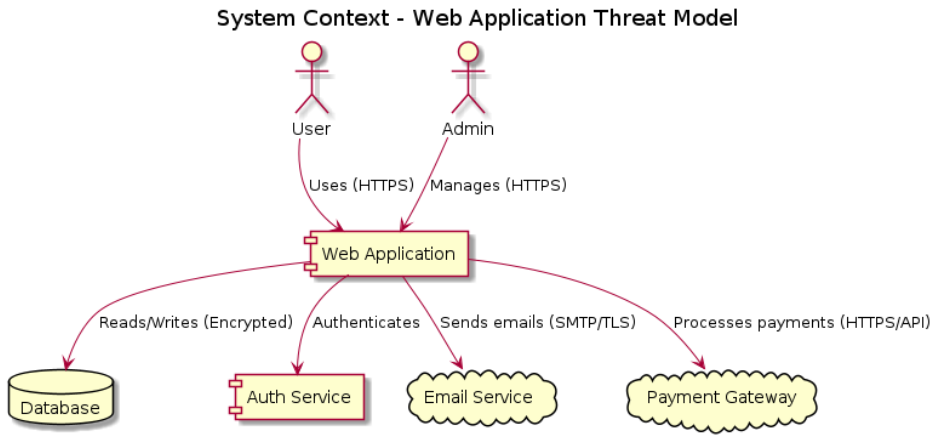

Figure 1: System Context Diagram

1.2 STRIDE Threat Analysis

Using Microsoft's STRIDE methodology, we identified the following potential threats (Figure **??**):

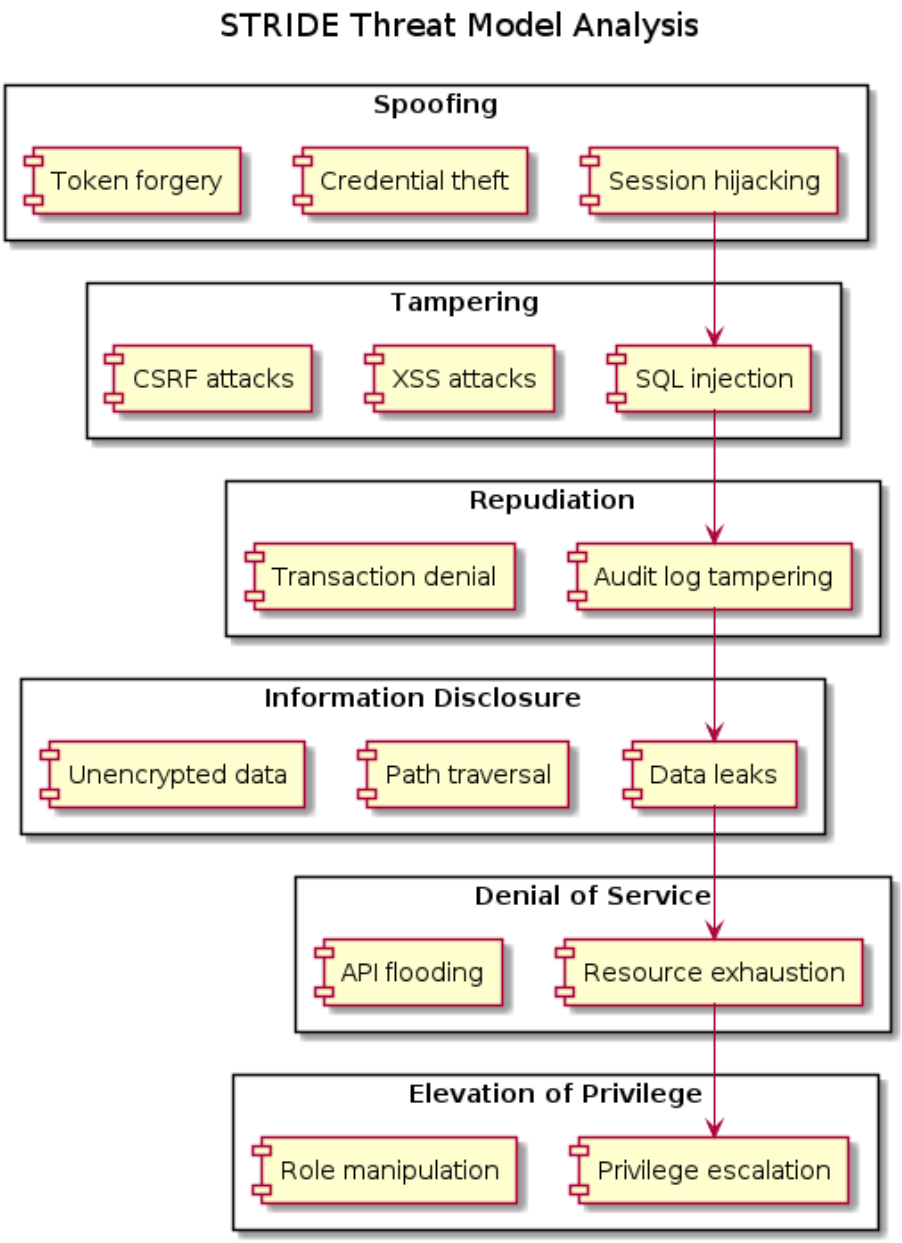

Figure 2: STRIDE Threat Analysis

2 Detailed Threat Analysis

2.1 Authentication and Authorization Threats

- **Threat:** Session hijacking through token theft
- **Impact:** Unauthorized access to user accounts
- **Controls:**
  - Implement secure session management
  - Use HTTP-only cookies
  - Employ token rotation

2.2 Data Integrity Threats

- **Threat:** SQL injection attacks
- **Impact:** Unauthorized data access or modification
- **Controls:**
  - Use parameterized queries
  - Input validation
  - Least privilege database access

2.3 Availability Threats

- **Threat:** DDoS attacks
- **Impact:** Service disruption
- **Controls:**
  - Rate limiting
  - DDoS protection services
  - Load balancing

3 Security Controls and Mitigations

3.1 Implemented Controls

1. **Authentication**
   - Multi-factor authentication
   - Password complexity requirements
   - Account lockout policies
2. **Authorization**
   - Role-based access control
   - Principle of least privilege
   - Regular permission audits
3. **Data Protection**
   - TLS 1.3 for data in transit
   - AES-256 for data at rest
   - Regular security audits

4 Risk Assessment Matrix

Threat	Likelihood	Impact	Risk Level
SQL Injection	High	Critical	High
Session Hijacking	Medium	High	High
DDoS Attack	High	Medium	Medium
Data Breach	Medium	Critical	High

Table 1: Risk Assessment of Identified Threats

5 Implementation Plan

5.1 Phase 1: Critical Controls

- Implement WAF (Web Application Firewall)
- Deploy MFA for all administrative access
- Establish secure coding guidelines

5.2 Phase 2: Enhanced Security

- Implement continuous security monitoring
- Deploy intrusion detection systems
- Regular penetration testing

5.3 Phase 3: Maintenance

- Regular security assessments
- Update threat models
- Security awareness training