### ****Threat Model: Payment Process in an E-commerce Web App****

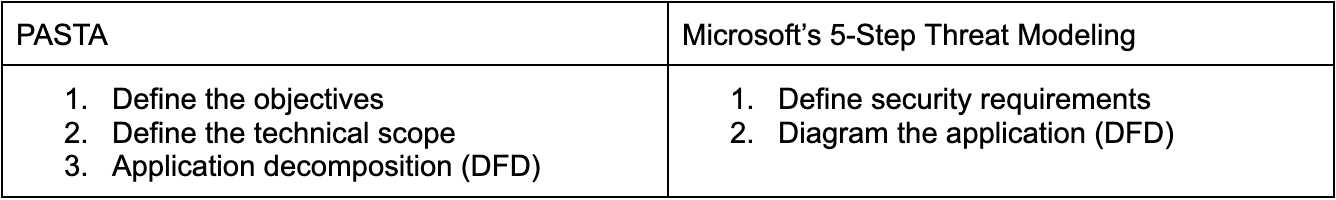
#### ****System Overview:****

This document outlines the threat model for the payment process in an e-commerce web application. The goal is to identify potential security threats, evaluate their impact, and implement mitigation strategies to protect sensitive data (e.g., cardholder information, user credentials) and ensure secure transactions. The threat model follows the four-question approach and aligns with industry standards such as OWASP ASVS and NIST CVSS.

The e-commerce web application allows users to browse products, add them to their cart, and complete purchases using various payment options, including credit/debit cards, PayPal, and digital wallets. The payment process involves communication with external payment gateway APIs.

**Note**: Using the four Question approach to apply Threat model on our example

**Part 1: What Are We Working On?**



**Objective**: Secure the e-commerce Web app payment process by identifying threats, evaluating potential attack vectors, and ensuring process is secure.

**Goal**: Protect sensitive data (e.g:cardholder information, user credentials) and prevent unauthorized access or manipulation of transactions.

**Scope:**

* Secure integration with the payment gateway
* Protection of sensitive data during transmission and storage

**Data Flow Diagram(DFD):**

**1. Entities:**

**Web Client:** Allows users to input payment details and confirm transactions.

**Payment Gateway API:** Processes payment requests and returns the transaction status.

**Database**: Stores order details but does not store sensitive payment information.

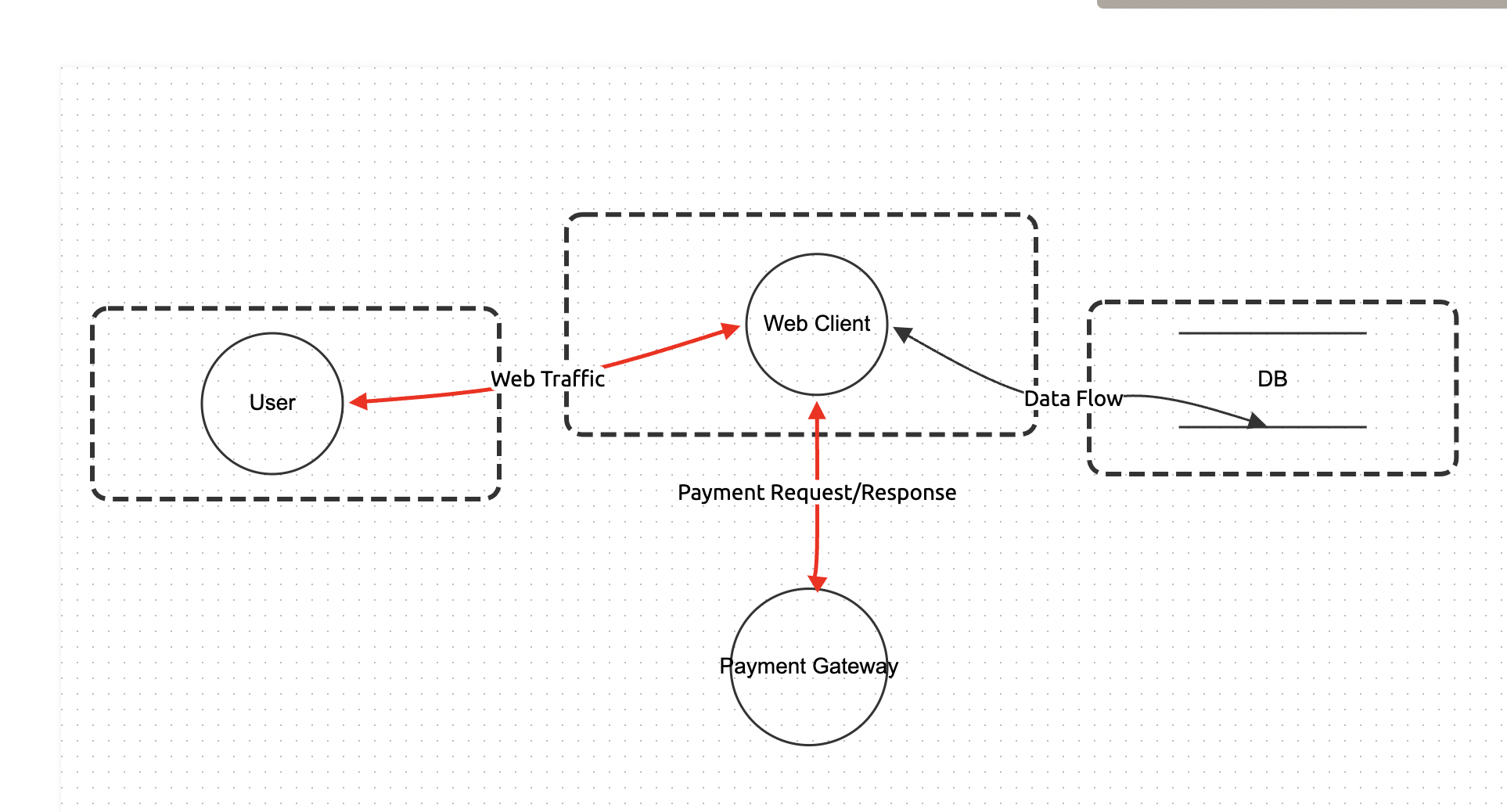
**User Device**: Hosts the app and may cache sensitive information unintentionally.

2. Visual Representation:

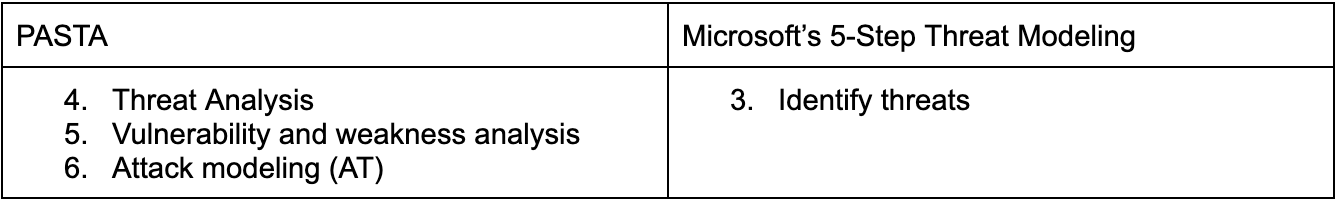
**User → Web Client:** Data submission and transaction initiation

**Web Client → Payment Gateway:** Secure API call for payment processing

**Web Client → Backend Database:** Storage of transaction history

****

**Part 2: What Can Go Wrong?**



The following concern is raised based on security standards like [OWASP ASVS](https://mas.owasp.org/MASVS/" \t "/Users/mayadaabdelsalam/Documents\\x/_blank)

**Current situation:**  
The checkout process includes sending user payment information (credit card number, CVV...) to the Payment Gateway API. Right now, we're including these details in plaintext in the request body.

**Threat Analysis:**  
Even though the request is sent over HTTPS, sensitive information in plaintext in the request body could be exposed if HTTPS is downgraded or intercepted via a Man-in-the-Middle (MitM) attack ,OWASP recommends encrypting sensitive data at the application layer, even over secure transport.

**Threat Identified:**

**1.** Plaintext Payment Data Exposure

* **Threat Title:** Sensitive payment details sent in plaintext.
* **Threat Type:** Information Disclosure.
* **Definition:** Payment details (e.g., card number, CVV) are transmitted in plaintext over HTTPS. If intercepted or logged, attackers could access sensitive data.
* **Affected Entity**: Data Flow between Web client and Payment Gateway API.
* **Mitigation:**
  + Encrypt sensitive payment data at the application layer before sending
  + Use secure tokens instead of raw card details
  + Ensure HTTP headers do not contain sensitive information

#### ****2 . Man-in-the-Middle (MitM) Attack****

* **Threat Title:** Data interception during transmission.
* **Threat Type:** Spoofing
* **Definition:** An attacker intercepts or alters payment details transmitted between the web client and the payment gateway API.
* **Affected Entity:** Data Flow between user and Web client
* **Mitigation:**
  + Enforce TLS for all communications.
  + Enable HSTS (HTTP Strict Transport Security) to prevent HTTPS downgrades.

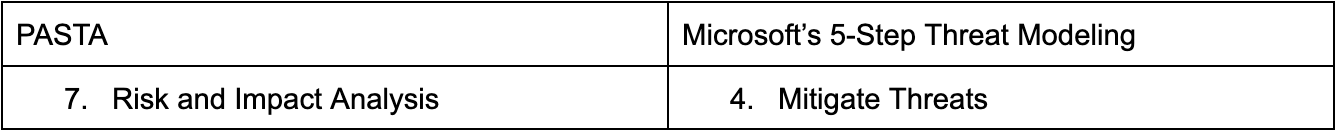
#### ****3 . Unauthorized Payment Modification****

* **Threat Title:** Payment request tampering.
* **Threat Type:** Tampering.
* **Definition:** An attacker intercepts and modifies payment requests (e.g:changing amounts or recipient details)
* **Affected Entity:** Data Flow between Web client and Payment Gateway API.
* **Mitigation:**
  + Digitally sign payment requests using a secure hashing algorithm
  + Validate signatures on the server-side.
  + Use a unique session or transaction ID to prevent replay attacks.

Apply [NIST CVSS (Common Vulnerability Scoring System)](https://www.first.org/cvss/" \t "/Users/mayadaabdelsalam/Documents\\x/_blank) for our threat:

|  |  |  |
| --- | --- | --- |
| **Threat** | **CVSS Score** | **Severity** |
| Plaintext Payment Data Exposure | 8.6 | High |
| Man-in-the-Middle (MitM) Attack | 7.5 | High |
| Unauthorized Payment Modification | 7.2 | High |

#### ****Part 3: What Are We Going to Do About It?****

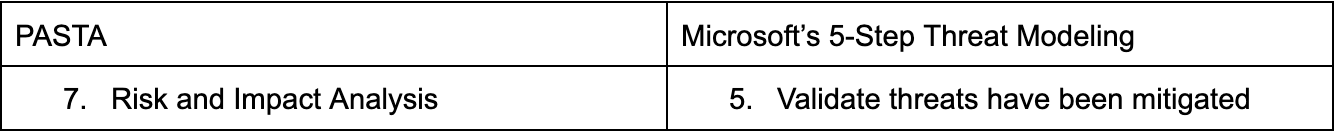


To address these threats, we refer to the **OWASP Application Security Verification Standard (ASVS)for best practices.**

### ****Comparison and Prioritization of Risks:****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ****Threat**** | **Impact** | **Likelihood** | **Risk Level** | **Priority** |
| Plaintext Payment Data Exposure | High (Compliance, Reputation) | High | Critical | 1 |
| Man-in-the-Middle (MitM) Attack | High (Confidentiality) | Moderate to High | High | 2 |
| Unauthorized Payment Modification | Moderate to High (Integrity) | Moderate to High | High | 3 |

#### ****Part 4: Did We Do a Good Job?****



After applying the mitigation:

* Update the **Mitigations field** of the identified threat at dragon app
* Change the **Status** of the threat from **Open** to **Mitigated**.
* Review the Threat Modeling diagram to ensure all entities, data flows, and boundaries align with the updated system.

Happy Threat Modeling.