## PASTA worksheet

### **PASTA Threat Model for Sneaker App**

#### **Stage I: Define Business and Security Objectives**

* The app should provide a seamless buying and selling experience for sneaker enthusiasts.
* Secure transactions and multiple payment options must be available.
* Strong data privacy practices must be in place to protect user information.

#### **Stage II: Define the Technical Scope**

The app uses the following technologies:

* **API** for interactions between app components.
* **PKI (AES and RSA)** for encrypting sensitive data.
* **SHA-256** for hashing passwords and credit card information.
* **SQL** for storing and retrieving sneaker listings and user data.

**Technology prioritization:** I would prioritize **SQL security** first because it is a major target for attacks like **SQL injection**, which could expose user data. The second priority is **API security**, as poorly secured APIs can be exploited to gain unauthorized access to app functionality.

#### **Stage III: Decompose Application**

The **data flow diagram** provided in the presentation focuses on the **product search process**:

* Buyers search for sneakers in the app.
* Listings are pulled from the SQL database and displayed to users.

To ensure secure data flow:

* Search queries should be sanitized to prevent **SQL injection**.
* The API should enforce **authentication and access control**.
* Sensitive user data should be **hashed and encrypted** before storage.

#### **Stage IV: Threat Analysis**

**Potential threats to the sneaker app:**

1. **SQL Injection** – Attackers could manipulate input fields to access or modify database information.
2. **Session Hijacking** – If session tokens are not properly secured, attackers can steal user sessions.

#### **Stage V: Vulnerability Analysis**

1. **Lack of Prepared Statements** – If the app does not use parameterized SQL queries, it becomes vulnerable to **SQL injection**.
2. **Weak Login Credentials** – If users are allowed to create weak passwords, it increases the risk of **brute-force attacks**.

#### **Stage VI: Attack Modeling**

The **attack tree** outlines possible attack paths:

* SQL injection due to lack of prepared statements.
* Session hijacking from poorly secured authentication mechanisms.

#### **Stage VII: Risk Analysis and Security Controls**

To mitigate risks, implement the following security controls:

1. **Use parameterized queries** to protect against SQL injection.
2. **Enforce strong authentication mechanisms** like MFA to prevent session hijacking.
3. **Encrypt user data** using AES for sensitive information like payment details.
4. **Secure APIs with authentication and rate limiting** to prevent abuse.