# PASTA Worksheet Portfolio

| Stages | Sneaker company |
|---|---|
| **I. Define business and security objectives** | ● **Seamless Connection**: The app's main goal is to seamlessly connect sellers and shoppers. This implies that the user interface should be user-friendly, allowing easy navigation and interaction between buyers and sellers.<br>● **Data Privacy**: Data privacy is a significant concern for the company. This means that the app should prioritize robust security measures to protect user data, including authentication and encryption, and comply with relevant data privacy regulations.<br>● **Payment Handling**: Proper payment handling is crucial to avoid legal issues. This indicates that the app needs to have secure payment processing mechanisms in place, possibly using well-established and trusted payment gateways. |
| **II. Define the technical scope** | List of technologies used by the application:<br>● *API*<br>● *PKI*<br>● *AES*<br>● *SHA-256*<br>● *SQL* |

| | |
|---|---|
| | In Stage II of the PASTA framework, one of the technologies that I would prioritize for evaluation is the Structured Query Language (SQL). The reason for this priority is that SQL is extensively used for interacting with and managing the database, which is a core component of the application. SQL injection is a prevalent attack vector, and given the sensitive nature of the data the app handles (e.g., user information, payment details), any vulnerabilities in SQL queries or database interactions could lead to data breaches, unauthorized access, or data manipulation. Thus, a thorough assessment of SQL-related security measures, including input validation and query parameterization, is essential to ensure the security of user data and the overall application. |
| **III. Decompose application** | Sample data flow diagram<br><br>In this data flow diagram, we have a simplified representation of a single process within the application: the Product Search Process. Let's analyze how the technologies evaluated in Stage II relate to protecting user data in this process:<br><br>**Structured Query Language (SQL**): SQL plays a central role in this process, as it's used to interact with the database (represented as element C). When a user (element A) searches for sneakers for sale, their query is processed by the Product Search Process (element B), which likely involves SQL queries to retrieve relevant data from the database. Ensuring secure coding practices, input validation, and proper use of SQL queries are essential to prevent SQL injection attacks, which could compromise the confidentiality and integrity of user data.<br><br>**Application Programming Interface (API):** Although not explicitly |

| | |
|---|---|
| | represented in this simplified diagram, APIs could be used in the background to facilitate communication between different components of the application. For instance, an API might be responsible for fetching data from the database or providing search results to the user. Securing API endpoints is crucial to prevent unauthorized access and data leakage.<br>**Public Key Infrastructure (PKI):** PKI, specifically the use of encryption algorithms like RSA, is vital for securing data in transit. When users search for sneakers, the data transmitted between their devices and the application's servers should be encrypted using strong encryption methods. This helps protect user queries and search results from eavesdropping and interception.<br>**SHA-256**: While SHA-256 is not directly involved in this user query process, it's likely used to secure sensitive user data stored in the database. For instance, user passwords and other critical information should be hashed using SHA-256 before storage to enhance data security. |
| **IV. Threat analysis** | Here are two types of threats that are risks to the information being handled by the app:<br>● **SQL Injection Attacks**: Threat actors may attempt SQL injection attacks on the app's database, exploiting vulnerabilities in the SQL queries used for various processes. If successful, these attacks could lead to unauthorized access to, modification, or theft of sensitive user data, such as personal information and transaction records. |

| | |
|---|---|
| | ● **Phishing and Social Engineering**: Users and employees of the company could be targeted with phishing emails or other social engineering tactics. If an attacker successfully tricks an employee into revealing their credentials or gains unauthorized access to a user's account through deceptive means, it could result in data breaches, fraudulent transactions, and compromised user accounts. |
| **V. Vulnerability analysis** | Here are two types of vulnerabilities that could be exploited:<br>● **Inadequate Data Encryption**: If the app fails to implement strong encryption mechanisms, particularly for sensitive data such as credit card information during payment transactions, it could be vulnerable to data interception and theft. Attackers could exploit this vulnerability to eavesdrop on user data and compromise their financial information.<br>● **Insufficient Input Validation**: Inadequate input validation on user inputs, especially in forms and search queries, can expose the application to various attacks, including SQL injection and cross-site scripting (XSS). Attackers might input malicious code or payloads that the app fails to properly sanitize, leading to potential code execution or data manipulation. |
| **VI. Attack modeling** | Sample attack tree diagram<br>Below is an example of an attack tree for the sneaker company's app, focusing on potential threats related to user |

| | |
|---|---|
| | data:<br><br>**Attack Tree: User Data Compromise**<br><br>**User Data**<br><br>This is the primary target for attackers, as it contains valuable information like user profiles, payment details, and personal data.<br><br>**Exploitation Vectors**<br>**a. SQL Injection (2a)**<br><br>Attackers might attempt SQL injection attacks to manipulate the database and retrieve user data.<br><br>**Subcategories:**<br>**Lack of Prepared Statements (3a):** If the app doesn't use prepared statements in SQL queries, it's vulnerable to SQL injection.<br><br>**Weak Login Credentials (3b):** If users have weak passwords, attackers could easily gain access to their accounts and steal data.<br>**b. Session Hijacking (2b)**<br><br>Attackers may attempt to hijack user sessions to gain unauthorized access to their accounts.<br><br>**Subcategories:**<br>**Insecure Session Management:** If the app doesn't properly |

| | |
|---|---|
| | manage user sessions, it could be vulnerable to session hijacking attacks.<br>This simplified attack tree outlines two main attack vectors, SQL injection and session hijacking, that could lead to the compromise of user data |
| **VII. Risk analysis and impact** | Here are four security controls that can help mitigate threats:<br>**Access Control**: Implement strong access controls to ensure that only authorized users can access sensitive data and system functions. This includes user authentication, role-based access control (RBAC), and proper permission settings.<br>**Encryption:** Use encryption mechanisms to protect data at rest and in transit. Employ techniques like Transport Layer Security (TLS) for secure communication and encryption algorithms (e.g., AES) to safeguard sensitive data stored in databases.<br>**Input Validation:** Implement thorough input validation and output encoding to prevent common vulnerabilities like SQL injection, cross-site scripting (XSS), and command injection. Validating and sanitizing user inputs helps filter out malicious input.<br>**Logging and Monitoring:** Set up comprehensive logging and monitoring solutions to detect and respond to security incidents in real-time. Monitor system logs, network traffic, and user activities for signs of suspicious behavior or unauthorized access. |