## PASTA worksheet

|  |  |
| --- | --- |
| **Stages** | **Sneaker company** |
| **I. Define business and security objectives** | 1. User experience: The app should allow easy sign-up, login, and account management, ensuring data privacy for users.  2. Payment processing: Secure transaction handling is crucial to avoid legal issues, requiring compliance with payment standards.  3. Messaging and ratings: The app enables buyers to message sellers and rate them, necessitating secure communication and data protection. |
| **II. Define the technical scope** | It is recommended to prioritize the evaluation of the API. APIs play a critical role in connecting different components of the application and are often targeted by attackers due to potential weaknesses in authentication, authorization, and data handling. Ensuring the security of the API will mitigate risks of unauthorized access and data breaches, safeguarding the system's overall integrity. |
| **III. Decompose application** |  |
| **IV. Threat analysis** | **Internal Threats:**  1. **Insider Threat**: An employee or contractor with access to sensitive information could misuse their privileges to access customer data or payment details, possibly due to poor enforcement of the principle of least privilege.  2. **Misconfigured Database Access**: Internal developers or administrators could accidentally misconfigure database permissions, allowing unauthorized access to sensitive information like user passwords or financial data.  **External Threats:**  1. **API Exploits**: A threat actor could exploit vulnerabilities in the public API to gain unauthorized access to sensitive user data or perform actions on behalf of legitimate users.  2. **Man-in-the-Middle Attack (MITM)**: Attackers could intercept communication between the app and server during key exchange or payment processing if encryption protocols like PKI are not properly implemented or if outdated algorithms are used. |
| **V. Vulnerability analysis** | 1. **API Vulnerabilities**: The public API could have insufficient input validation or weak authentication mechanisms, leaving it vulnerable to attacks such as SQL injection or unauthorized access. This could allow attackers to manipulate or steal sensitive data, such as user credentials or payment details.  2. **Weak Database Security**: The SQL database may have misconfigured access controls, allowing unauthorized users to view or modify sensitive data. Additionally, if encryption for stored data (e.g., credit card numbers, passwords) is not properly implemented or out-of-date, attackers could exfiltrate this information in plaintext. |
| **VI. Attack modeling** |  |
| **VII. Risk analysis and impact** | 1. Implement Multi-Factor Authentication (MFA): Adding an additional layer of authentication for users logging in to the app can prevent unauthorized access, even if credentials are compromised.  2. API Security Best Practices: Secure APIs by enforcing strong authentication, validating all inputs, and implementing rate limiting to prevent brute-force attacks and unauthorized access to sensitive endpoints.  3. SQL Query Parameterization: Use prepared statements and parameterized queries to protect the application from SQL injection attacks, ensuring that user input is safely handled.  4. Encryption of Sensitive Data: Implement end-to-end encryption using strong algorithms (e.g., AES) for data at rest and in transit, especially for payment and personal data, to mitigate the risk of data interception or leakage. |