
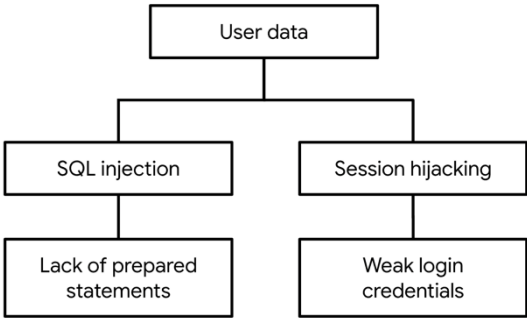


## PASTA Threat Model Framework

Stages	Sneaker Company
1. Define Business and Security Objectives	<ul style="list-style-type: none"> <li>• Users can create member profiles internally or by connecting external accounts.</li> <li>• Seamless connectivity between sellers and buyers.</li> <li>• Data privacy and the security of user information.</li> <li>• Clear and quick sales processing with multiple payment options, ensuring proper payment handling while in compliance with PCI-DSS.</li> </ul>
2. Define Technical Scope	<ul style="list-style-type: none"> <li>• API – APIs are gateways that allow different software components to communicate. Given the app's functionality, it will likely use various APIs for user registration, product listings, messaging, and payment processing. If not secured, APIs can provide an avenue for data breaches.</li> <li>• PKI – This encryption framework ensures the secure exchange of data. Ensuring that sensitive data exchanges, such as those during payments, are encrypted is crucial.</li> <li>• SHA-256 – Used for hashing, especially sensitive user data like passwords. While it offers good security, ensuring it's implemented correctly is crucial. Using salt with SHA-256 hashes can prevent rainbow table attacks.</li> <li>• SQL – Databases using SQL are essential in such applications for storing product listings, user data, and transaction records. Without proper security measures, these databases can be vulnerable to attacks like SQL injections.</li> </ul> <p>I would first evaluate the API. APIs often interact with all the other mentioned technologies and form the foundation for application functionality. If the API is compromised, it could expose multiple parts of the system, from user data in the database to vulnerabilities in the encryption mechanisms.</p>
3. Decompose Application	<p>Data Flow Diagram (Simplified):</p>  <pre> graph LR     User[User] -- "Searching for sneakers for sale." --&gt; Process((Product search process))     Process -- "Listings of current inventory." --&gt; Database[Database]   </pre> <ul style="list-style-type: none"> <li>• User – The starting point where an individual interacts with the application interface.</li> <li>• Searching for Sneakers for Sale – The user initiates a search query through the app's interface. This is where the user input gets processed and sent to the server.</li> </ul>

	<ul style="list-style-type: none"> <li>• Product Search Process – The server processes the user's search query, translating it into database-readable commands to fetch the relevant data. This stage involves logic to sort, filter, and select the correct listings based on user preferences and search criteria.</li> <li>• Listings of Current Inventory – This represents the aggregated results from the database that match the search query. The server packages these results into a format that the app can display effectively to the user.</li> <li>• Database – The foundational storage where all the product details are kept. The database responds to the commands from the product search process and sends the relevant data back to be presented as listings.</li> </ul> <p>By decomposing the data flow in this manner, we can pinpoint areas of concern:</p> <ul style="list-style-type: none"> <li>• The Searching for Sneakers for Sale and Product Search Process stages can be vulnerable to SQL injection attacks if user inputs are not properly sanitized.</li> <li>• The Database itself needs to be secure to ensure that unauthorized entities cannot gain direct access, which would bypass the intended data flow and lead to data breaches.</li> <li>• Communication between the stages should be encrypted to ensure data integrity and confidentiality.</li> </ul>
4. Threat Analysis	<ul style="list-style-type: none"> <li>• Injection Attacks – The app uses SQL databases, it's susceptible to such threats. Proper input validation can deter these attacks.</li> <li>• Session Hijacking – The app's mechanism of transmitting cookies across layers can be exploited to hijack sessions.</li> <li>• Man-in-the-Middle Attacks – As the application communicates data, attackers could intercept this data, leading to data breaches or unauthorized manipulations.</li> <li>• Phishing Attacks – Users might be targeted with counterfeit interfaces to steal their credentials.</li> </ul>
5. Vulnerability Analysis	<ul style="list-style-type: none"> <li>• Lack of Prepared Statements – This shortfall can expose the SQL database to injection attacks.</li> <li>• Cookie Mishandling – Poor management or insecure transmission of cookies can make session hijacking feasible.</li> <li>• Weak API Token Management – If API tokens are not securely generated and managed, it can lead to unauthorized access.</li> <li>• Insecure Data Storage – Data stored without proper encryption or in predictable locations can be a goldmine for attackers.</li> </ul>

6. Attack Modeling	<p>Attack Tree Diagram (Simplified):</p>  <pre> graph TD     A[User data] --&gt; B[SQL injection]     A --&gt; C[Session hijacking]     B --&gt; D[Lack of prepared statements]     C --&gt; E[Weak login credentials] </pre> <p>User Data: This is the ultimate target for threat actors.</p> <ul style="list-style-type: none"> <li>• SQL Injection – One of the primary attack vectors to gain unauthorized access to user data. <ul style="list-style-type: none"> <li>○ Lack of Prepared Statements – An insecure coding practice that doesn't bind variables securely in SQL statements, making it susceptible to SQL injection.</li> </ul> </li> <li>• Session Hijacking – Another method where attackers take over a user's session to impersonate them and gain unauthorized access. <ul style="list-style-type: none"> <li>○ Weak Login Credentials – This condition can make it easier for attackers to guess a user's credentials, log in as them, and then hijack their session.</li> </ul> </li> </ul> <p>For our Sneaker Company application, this attack tree highlights two critical areas for security enhancement, improving SQL command structures and strengthening authentication and session management mechanisms.</p>
7. Risk Analysis and Impact	<p>Given the vulnerabilities and threats identified, the following security controls can reduce risk:</p> <ul style="list-style-type: none"> <li>• MFA – Require MFA for both users and administrators, ensuring a second layer of authentication.</li> <li>• Regular Security Audits – Conducting routine penetration testing and vulnerability assessments to identify and patch potential weaknesses.</li> <li>• Data Encryption – Encrypting user data both at rest and in transit using robust encryption algorithms.</li> <li>• Rate Limiting – Implementing rate limiting on the API to prevent DDoS attacks and limit data scraping.</li> </ul> <p>By applying these controls and continuously monitoring the system for anomalies, the application's security can be enhanced, thereby reducing the potential for data breaches or other malicious activities.</p>