Web Application Security

TA-2 Assignment

Submitted By

Raj Shekhar

Enrolment no.: 240545002004

M.Sc. in Cyber Security

Semester I

Guided By

Mrs Meena Lakshmi

(Assistant Professor)

School of Cyber Security & Digital Forensics

National Forensic Science University

Bhopal Campus, M.P., 462001, India

A Sample Threat Modeling Scenario in a Web Application

Threat modeling is a proactive approach used to identify and address potential security risks in an application. This report provides a sample threat modeling scenario for a typical web application, detailing the threats, vulnerabilities, and mitigation strategies.

Web Application Overviews

I. It can be e-commerce platform where users can browse and search products, make purchases, and manage their profiles. It consists of :

- A front-end interface (HTML, CSS, JavaScript),
- A back-end API (RESTful),
- A database storing sensitive information like user credentials and payment details.

II. It can be a social media platform that allows users to create profiles, share posts and interact with others. It consists of same a front-end interface and a back-end API but with a database for managing user data, posts, comments, and interactions.

There are more web applications like CMS (Content Management System), Online Banking platforms, Online Learning platforms, Customer Support (Helpdesk), etc.

In this report we cover an **e-commerce platform** as a sample to perform threat modeling analysis.

Purpose & Objective

It is to conduct a threat modeling exercise and identify assets to address potential security risks using a methodology like STRIDE and suggest mitigation strategies, without tampering any confidentiality, Integrity, Availability of sensitive user data, and preventing any unauthorised access to any accounts or admin panels, which complies with GDPR.

Identifying Assets

It is the list of valuable assets within the web application that requires protection :

1. User data : Personal Information & Credentials, Payment information & orders,

2. Session Management : Cookies and Authorisation token (CSRF), Session data,

3. Source Code : Frontend (UI) and backend (Business logic) scripts and queries,

4. Server Infrastructure : Web servers, Database servers, Load balancers, firewalls,

5. Database : store user details, products, orders, inventory & other transactions data,

6. Admin Interface : to manage users, products, orders, and with admin credentials.

Threat Modeling Methodologies

There are many threat modeling frameworks used like :

- **STRIDE** (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) used for general threat categories,

- **DREAD** (Damage Potential, Reproducibility, Exploitability, Affected Users, Discoverability) used for prioritizing risks based on severity,

- **PASTA** (Process for Attack Simulation and Threat Analysis) used for attack simulation and risk analysis,

- **OCATVE** (Operationally Critical Threat, Asset, and Vulnerability Evaluation) used for Asset-based threat analysis and organizational risk,

- **VAST** (Visual, Agile, and Simple Threat Modeling) used for visual, agile, and simple threat modeling,

- **Attack Tree**, is used for mapping attack paths and methods,

- **LINDDUN** (Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of information, Unawareness, Non-compliance) used for privacy threat modeling.

Identifying the Threats using STRIDE

In this scenario, we will use **STRIDE model** for the e-commerce web application, that covers :

- **Spoofing**: Attackers may impersonate a legitimate user or admin using phishing techniques to gain unauthorized access to the application.

  Mitigation :

  - using Strong authentication mechanisms (e.g. multi-factor authentication),
  - ensuring Session tokens are securely generated and managed (e.g. HttpOnly, Secure cookies),
  - implementing Role-Based Access Control for admin functionalities.

- **Tampering**: Attackers could modify API requests to alter the data or logic of the application (e.g., changing a stock check API request before submission to access admin panel as localhost).

  e.g.: **"stock-check" : "http://localhost:8080/admin/manage"** in the request body.

Mitigation :

- storing logs securely,
- implementing logging for URL inputs and internal redirects.

- **Repudiation**: Users or admins could perform actions (like deleting data) and deny doing so, potentially causing integrity issues in the app.

  Mitigation :

  - Implementing a proper **logging** of URL inputs and internal redirects, including search queries and page interactions.

- **Information Disclosure**: Sensitive data, such as user passwords or financial information, could be exposed due to weak encryption or improper access control (means able to perform SQLi). e.g.: **‘ OR 1=1 --** in URL with search params.

  Mitigation :

  - storing passwords using strong encryption (e.g., bcrypt) and never store plaintext credit card details (use tokens instead),
  - using input sanitization/escaping, also parameterized queries and avoid storing sensitive data in an easily accessible format.

- **Denial of Service** (DoS): Attackers could overload the server with traffic, preventing legitimate users from accessing the application.

  Mitigation :

  - implementing rate limiting and web application firewalls (WAFs) like Cloudflare.

- **Elevation of Privilege**: A normal user might find a way to escalate their privileges to become an admin or perform actions outside their role. e.g.: **"stock-check" : "http://localhost:8080/admin/delete/user/1"** in stock check API request.

  Mitigation :

  - applying **least privilege** principles across the application, ensuring that users only have access to the data and functions they need and also at database levels.

Risk Assessment and Prioritization

Based on the identified threats, here’s the risk assessment for this e-commerce web application scenario :

STRIDE Category	Threat Description	Risk Level	Mitigation Priority
Spoofing	Attacker impersonates a legitimate user or admin.	**Medium**	**High**
Tampering	attacker allowed to manipulate server requests to access internal endpoints, malicious JavaScript execution.	**High**	**High**
Repudiation	Attacker denies performing malicious actions due to lack of traceability	**Medium**	**Medium**
Information Disclosure	Exposure user cookies, session tokens, or sensitive database data	**High**	**High**
Denial of Service	Overloading the system with malicious search queries.	**Medium**	**Medium**
Elevation of Privilege	Attack leads to higher-level access or privilege escalation.	**High**	**High**

Conclusion

In conclusion, securing an e-commerce web application requires a comprehensive approach to mitigate a wide range of security risks, including Server-Side Request Forgery (SSRF), Cross-Site Scripting (XSS), and SQL Injection (SQLi) vulnerabilities. Threat modeling with the STRIDE framework provides a structured way to identify potential threats to critical assets such as user data, payment information, the database, and the admin interface. For SSRF, the primary concern is unauthorized access to internal services, such as the admin panel, which can lead to privilege escalation and data breaches. To address these risks, key mitigation strategies include input validation, output encoding, parameterized queries, network segmentation, strong authentication, and access control. Effective session management, logging, and monitoring also play vital roles in identifying malicious activity early and ensuring accountability. By understanding and addressing these risks through proactive security measures, e-commerce platforms can better protect sensitive user data, preserve system integrity, and maintain user's healthy trust.

References

- **OWASP Top 10** : A comprehensive list of the most critical web application security risks, including XSS, SQLi, and SSRF.
- **Microsoft STRIDE Framework**: A threat modeling methodology to identify and address security threats in a system.
- **ThreatModeler**: A tool that helps organizations visualize and analyze threats to their web applications using automated threat modeling.

Threat Modeling Diagram Example

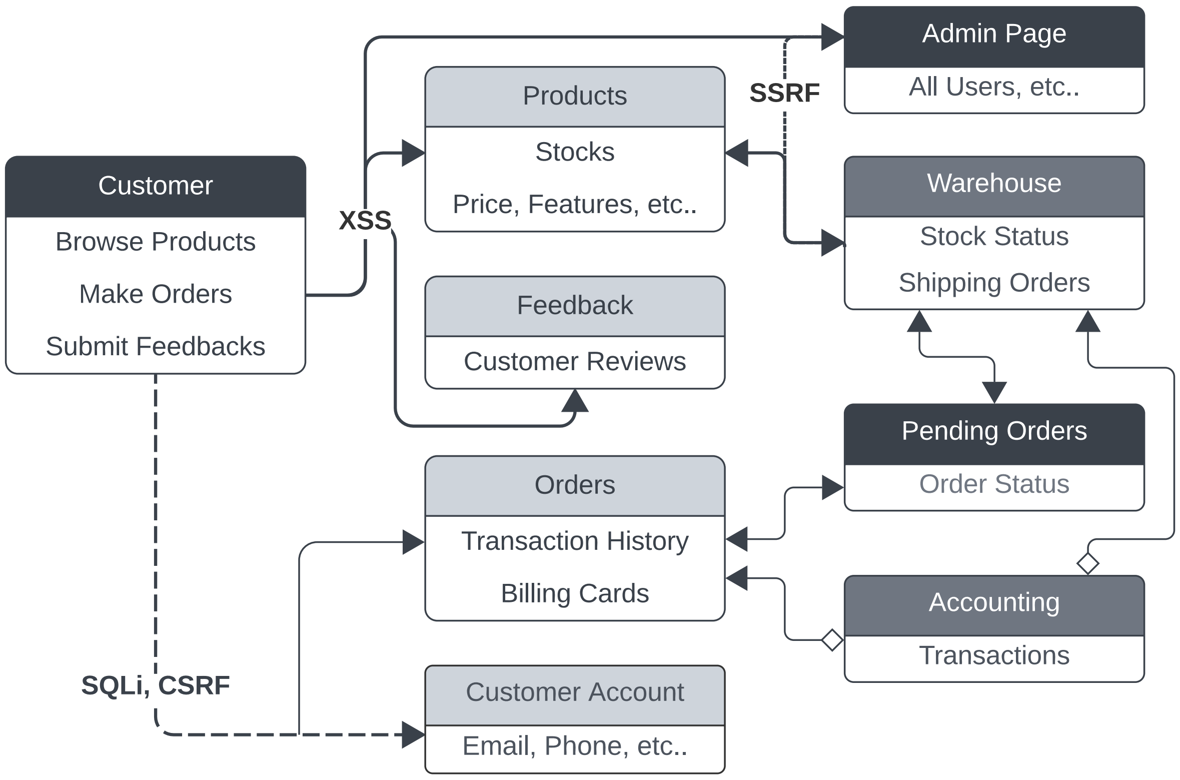

Fig: A dataflow diagram in an ecommerce website for Threat Intelligence