Fake Stack Overflow Threat Modeling

Authors: John Jacoby, Yue Wen Peter Li
Course: CS5500 - Foundations of Software Engineering Term: Spring 2024
Professor: Joydeep Mitra

Identify Five Possible Threats:

Threat 1 - “Spoofing” of User Credentials

Without proper safeguards, an attacker could spoof the user’s session and get the server to perform actions on that user’s behalf that they did not actually initiate. In our ‘User Profiles’ requirement, users need to be authenticated before viewing or updating their own profile(username, profile picture, and activity summary).

Threat 2 - “Elevation of Privilege” to Moderator Status

In our 'Post Moderation' requirement, we have a moderation system where specially privileged users, called 'Moderators', can perform actions that regular users cannot such as editing, removing, flagging, or locking posts that they do not own. This obviously gives these users extensive power to alter and/or damage website content. An attacker could exploit this by bypassing the authentication check or exploiting the absence of proper authorization to gain a moderator's privileges. Therefore, we need to establish robust access control measures to prevent malicious users from escalating their roles from a regular user to a mod, thus addressing this security concern.

Threat 3 - “Tampering” via Database Code Injection

In our project, requirements like 'Create New Posts', 'Search for Existing Posts', 'Commenting on Posts', 'Tagging Posts', and 'Post Moderation' all involve user input. These requirements implicitly require a persistent data store on the backend to manage data storage and relationships. An attacker can exploit this by entering database commands into fields such as 'username' or 'post title'. When this input is processed by the database, it could execute these commands instead of just storing them as part of the input string. Theoretically, this vulnerability could allow an attacker to perform any operation they desire on our database - altering user permissions, modifying user login information, editing existing posts and comments, or even deleting the entire database.

Threat 4 - “Denial of Service” Attack to the server

In our project, requirements including 'Create New Posts', 'Search for Existing Posts', 'Commenting on Posts', 'Tagging Posts', and 'Post Moderation' all involve user input. Attackers could cause the system to crash by inducing errors, for example, by sending specific requests that trigger unhandled exceptions or resource exhaustion, leading to system failure.

Threat 5 - “Information Disclosure” in middleware

In our project, all requirements involve communication between the client and server through middleware. However, unencrypted data may pose a potential threat of information disclosure. For example, if the login information is transmitted in plain text from the client to the server, an attacker intercepting the communication could easily access the user's login credentials. This issue highlights the risks associated with not encrypting sensitive data during transit and storage. Additionally, sensitive data leakage can occur if the codebase inadvertently exposes confidential information. To address these vulnerabilities, it is crucial to implement robust encryption mechanisms for data in transit and storage and to ensure that sensitive information is never exposed in the application's code.

Threat Specification:

Threat 1 - “Spoofing” of User Credentials

Revised User Profile Requirement:

Inputs: Request with the username, password.
Outputs: User profile information page (username, profile picture, and activity summary)
Action: The system allows users to input and update their profile information, which is stored in the

database. Users' activities, like posts and voting history, are aggregated and displayed on their profile page. **Security Enhancements**: Implement enhanced authentication mechanisms to prevent user credential spoofing. All defined cookies must set the httpOnly flag to true. All user input will be schema-validated and sanitized to prevent injection attacks.
Pre-condition: Users must be registered and authenticated to edit an account. New account creation requires being logged out, and the account must not already exist.
Post-condition: User's profile is created or updated with the provided information, which is then visible to the user and others.

Threat 2 - "Elevation of Privilege" to Moderator Status
Revised Post Moderation Requirement:

Inputs: Request with Post ID, moderator actions (edit, delete, flag, lock/unlock), moderator ID.
Outputs: Post – The state of the post after the moderator has performed an action on it..
Action: The system enables moderators to scrutinize posts and take necessary actions, including editing, deleting, flagging, or locking/unlocking them. **Security Enhancements**: Ensure that session IDs are randomly generated using a cryptographically secure method and are regenerated upon each login to prevent session hijacking. Apply the httpOnly flag to cookies to protect against XSS attacks and secure user sessions. Sanitize all user inputs to prevent code injection and maintain system integrity.
Pre-condition: Moderator must be authenticated and authorized to perform moderation tasks.
Post-condition: The post is edited, deleted, or flagged as per the moderator's action, influencing its visibility and status on the site.

Threat 3 - "Tampering" via Database Code Injection
Definition of Done:

Input Validation: Mongoose schemas will be used to define exactly what the structure of a document should be. Mongoose parameterized queries will be used exclusively instead of direct database access. All api endpoints should be tested, ensuring all user inputs, especially those used in database operations like creating new posts, searching, commenting, tagging, and post moderation, are rigorously validated and sanitized to prevent injection of malicious code.
.
This DoD applies to the following requirements: 'Create New Posts', 'Search for Existing Posts', 'Commenting on Posts', 'Tagging Posts', and 'Post Moderation'. All these functions involve user input and are thus subject to these security measures.

Threat 4 - "Denial of Service" Attack to the server
Definition of Done:

Implement Comprehensive Error Handling: Develop error handling strategies to prevent system crashes caused by specific requests that could lead to unhandled exceptions or resource exhaustion. Ensure that all potential errors are properly managed.
Conduct System Resilience Testing: To ensure system robustness, test various types of invalid inputs, including Special Characters (such as HTML tags like **<script>**), Null and Empty Strings, Script Injection, and HTTP links.

This DoD applies to the following requirements: 'Create New Posts', 'Search for Existing Posts', 'Commenting on Posts', 'Tagging Posts', and 'Post Moderation'. Since these functionalities involve user input, they are subject to these security measures.

Threat 5 - "Information Disclosure" in middleware
Epic: Transport Layer Security (TLS)

Deploy TLS to secure transmission and storage of sensitive data, by input validation(or filtering) and output encryption mechanisms. Public-private key encryption protocols to safeguard data during

transit. HTTPS will be enforced across all client-server communications to encrypt data. Sensitive information, such as passwords, will be encrypted on the client side prior to transmission. The strategy extends to data sanitization and validation to ensure all user inputs are filtered, and secure coding practices will be adopted to prevent confidential information from being exposed in the codebase. To test the TLS,

Three Top-Priority Threats to Address

Threat 1 - "Spoofing" of User Credentials:
The significance of this issue stems from the potential for unauthorized access and inadequate or weak authentication, which can lead to session hijacking and identity theft. This threat is directly related to user privacy; improper handling could result in a loss of user trust. Moreover, it might also lead to privilege escalation. Ensuring secure authentication and session management is crucial to mitigating these risks.

Threat 2 - "Elevation of Privilege" to Moderator Status
An unauthorized user with moderator privileges could wreak havoc on any target user's post by deleting it or changing it to say anything they'd like, including offensive or illegal material that violates the site's policies. This is also a 'many birds with one stone' threat to address as XSS attacks can come in a variety of flavors that could all be damaging to target users in different ways. For instance, an attacker could also redirect a user to a phishing site that steals their login credentials. We also avoid Threat 1 by solving this threat.

Threat 3 - "Tampering" via Database Code Injection
This is the most vital to address because the failure to address this threat is the most costly. An attacker could delete the entire website's database or give any user permission to perform any action. Sending passwords in plain text over the internet is extremely exposing. It's a low bar for even amateur attackers to achieve, meaning that this will be one of the most common types of attack. Users losing their accounts to hackers will create an unacceptable user experience.

Final List of Requirements

1. View Posts

Function: Display Post
Description: User clicks on a question title hyperlink. Database then retrieves corresponding post and display on the web page from the database.
Inputs: URL and User ID.
Source: User input through the interface, Post data stored in the site's database.
Outputs: Contents of the Displayed Post
Destination: User's web page.
Action: When a user clicks the hyperlink, the system fetches post from the database. The posts are then formatted and displayed on the user's web page. The post may include questions, answers, and relevant metadata like author, date, vote count, and tags.
Requirements: Database access to retrieve post data. Properly formatted post data including metadata. Question titles and hyperlinks need to be presented on the homepage or in search results.
Pre-condition: Database contains the post that can be retrieved and displayed, and the web page hyperlink displayed on the homepage or in search results.
Post-condition: User is presented with the post.
Side effects: None.

2. Create New Posts

Function: Post Submission
Description: Allows users to submit new posts to the site. The function handles the collection of post content, tags, and submission to the database for storage.
Inputs: Post content, selected tags, user ID, datetime.
Source: User input through the site interface, datetime service.
Outputs: NewPost – the newly created post that is submitted to the database.

Destination: Site's database for storing posts and the webpages for the posts.
Action: When a user submits a new post, the system associates it with the user's ID, captures the datetime of the post, and stores it in the database. The post is then made available for viewing by other users.
Requirements: Text editor for post composition and user authentication to associate posts with user accounts.
Pre-condition: User must be authenticated and authorized to create a post. Database does not contain the post.
Post-condition: A new post is added to the database and becomes visible on the site, i.e. will show up in response to the 'search' and 'view' requirement functions.
Side effects: Too many posts at the same time may overwhelm the system

3. Search for Existing Posts

Function: Post Search
Description: Enables users to search for existing posts based on keywords, tags, author names, and/or datetime ranges. This function uses a search algorithm to retrieve and display a list of posts that match the search terms, as well as sort the returned list.
Inputs: Search query (keywords, tags, author names, datetimes), optional sorting parameters (alphabetical by username, datetime, number of up-votes).
Source: User input through the site's search interface.
Outputs: SearchResults – a list of posts that match the search criteria.
Destination: User's device screen.
Action: The system processes the search query, queries the database, and retrieves posts that match the criteria. The results are then formatted and displayed to the user.
Requirements: A search algorithm, access to the post database, filtering and sorting mechanisms.
Pre-condition: The database contains indexed posts that can be searched.
Post-condition: The user is presented with a list of posts that match the search criteria.
Side effects: Too many requests at the same time may overwhelm the system. Web crawler may steal the information from the database using the method.

4. Commenting on Posts

Function: Post Comment
Description: Allows users to add comments to posts. This function handles the collection, validation, and submission of user comments, as well as their display under the corresponding post.
Inputs: Comment text, user ID, post ID (the ID of the post being commented on).
Source: User input through the site's interface.
Outputs: NewComment – the user's comment, which is added to the site under the specified post.
Destination: Database entry and the webpage associated with the specific post.
Action: The system captures the user's comment, associates it with the user's ID and the relevant post ID, then stores it in the database. The comment is then displayed under the post for other users to see. **Requirements**: Text input field for comments, user authentication to associate comments with user accounts. The question is not locked.
Pre-condition: User must be authenticated. The post on which the comment is to be made must exist. **Post-condition**: The comment added to the post and is visible to other users.
Side effects: Too many comments may make it difficult for users to read the entire comment thread.

5. Voting on Posts

Function: Post Voting
Description: Enables users to vote on posts to indicate their quality or relevance. This function allows for up-voting or down-voting of posts and handles the updating of vote counts associated with each post.
Inputs: Vote (up-vote or down-vote), user ID, post ID (the ID of the post being voted on).
Source: User input through the site's interface.
Outputs: UpdatedVoteCount – the new vote count for the post after the user's vote has been applied.
Destination: Database entry and webpage associated with the specific post.

Action: The system registers the user's vote, updates the post's total vote count in the database, and reflects this change on the user interface. The vote count influences the visibility and ranking of posts in search and view functions.
Requirements: Voting mechanism on the user interface, user authentication to ensure legitimate voting.
Pre-condition: User must be authenticated. The post on which the vote is to be made must exist.
Post-condition: The vote count of the post is updated, potentially affecting its visibility and ranking on the site.
Side effects: Malicious use may lead to an unfair ranking system.

6. Tagging Posts

Function: Post Tagging
Description: Allows users to add tags to their posts to categorize and facilitate easier searching and sorting of posts. This function enables the assignment of multiple tags per post, and the management of these tags, including adding new tags or selecting from existing ones.
Inputs: Selected tags, user ID, post ID (the ID of the post being tagged).
Source: User input through the site's interface.
Outputs: NewPost – the post with the newly assigned tags.
Destination: Database entry and webpage associated with the specific post.
Action: The system allows the user to select or create tags when creating a post. These tags are then associated with the post in the database, aiding in the organization and retrieval of posts via the search and view functions.
Requirements: Interface for selecting or creating tags, any requirement necessary for the 'Create New Post' functionality, since this is built into that.
Pre-condition: Currently in the post create workflow.
Post-condition: The post is created with the selected tags, enhancing searchability and categorization. **Side effects**: None.

7. User Profiles

Function: Profile Management
Description: Allows users to create, view, edit, and delete their own profiles on the site. This function manages user information, including usernames, account creation date, and profile pictures. It also displays user activity such as their posted content and voting history.
Inputs: Request with user information (username, profile picture, etc.).
Source: User input through the site's interface. Database.
Outputs: UserProfile page – the user's profile with their information and activity summary.
Destination: User's profile webpage and database for storing profile data.
Action: The system allows users to input and update their profile information, which is stored in the database. Users' activities, like posts and voting history, are aggregated and displayed on their profile page.
Security Enhancements:Implement enhanced authentication mechanisms to prevent user credential spoofing, which may include All defined cookies must set the httpOnly flag to true. All user input will be validated and sanitized to prevent injection attacks.
Requirements: Interface for entering and editing profile information, database storage for user data, mechanisms to display user activity, authentication mechanisms for logging in.
Pre-condition: User must be registered and authenticated to edit an account, or logged out to create a new account. Account must not already exist.
Post-condition: User's profile is created or updated with the provided information, which is then visible to the user and others.
Side effects: None.

8. Post Moderation

Function: Moderation System
Description: Provides a system for moderators (selected users) to review and manage posts to ensure they adhere to the site's guidelines. This includes editing, deleting, flagging, or

locking/unlocking posts that are inappropriate, off-topic, or violate community standards.
Inputs: Request with Post ID, moderator actions (edit, delete, flag, lock/unlock), moderator ID.
Source: Moderator input through the site's interface.
Outputs: Post – the post after undergoing the moderation action.
Destination: Database entry associated with the specific post and moderation logs.
Action: The system allows moderators to review posts and perform necessary actions such as editing content, removing posts, flagging or locking them for further review. Locked posts can also be unlocked.
Security Enhancements: Ensure that session IDs are randomly generated using a cryptographically secure method and are regenerated upon each login to prevent session hijacking. Apply the httpOnly flag to cookies to protect against XSS attacks and secure user sessions. Sanitize all user inputs to prevent code injection and maintain system integrity.
Requirements: Moderation tools interface, user roles with different permissions (e.g., regular user, moderator).
Pre-condition: Moderator must be authenticated and authorized to perform moderation tasks.
Post-condition: The post is edited, deleted, or flagged as per the moderator's action, influencing its visibility and status on the site.
Side effects: Some moderators may abuse their privileges and potentially harm the community.

9. DoD (**Threat: Tampering)**

This DoD applies to the following requirements: 'Create New Posts', 'Search for Existing Posts', 'Commenting on Posts', 'Tagging Posts', and 'Post Moderation'.