Lab 8 – Input Validation

Eshwar Kyatham

Northern Arizona University

ITC 505: Web Techonolgy

Troy L Adams

07/22/2024

Source Code

Website Link: <https://eshwarkyatham16.github.io/ITC505/lab-8/index.html>

Index.html

A screen shot of a computer

Description automatically generated

Styles.css

A screen shot of a computer screen

Description automatically generated

Scripts.js (With JavaScript validation, and XSS/SQLi prevention)

A screen shot of a computer program

Description automatically generated

A computer screen shot of a program code

Description automatically generated

**XSS and SQLi**

Cross-Site Scripting (XSS)

A common security flaw in web applications is called cross-site scripting (XSS), which enables attackers to insert malicious scripts onto web pages that other users are seeing. These scripts can carry out random operations while a user is logged in, which could have a variety of harmful effects.

Types of XSS Attacks

*Stored XSS:*

* Description: The malicious script used in stored XSS attacks is kept on the server indefinitely. This might be kept in a bulletin board, database, or other type of permanent storage. The malicious script is delivered as part of the page content when additional visitors access the compromised page.
* An example would be if a hacker left a malicious JavaScript remark on a blog. The software runs in other users' browsers when they visit the blog, possibly stealing session tokens or cookies.

*Reflected XSS:*

* Description: When user input is instantly mirrored off a web application without sufficient sanitization, it can lead to mirrored XSS. Usually, form submissions or URL parameters are used to do this. The victim's browser then launches the malicious script.
* A malicious payload is crafted into a URL and sent as a query parameter by an attacker. The script in the URL runs when a user hits this link since the web page's response reflects it.

*DOM-based XSS:*

* Description: DOM-based XSS targets flaws in JavaScript client-side code as opposed to server-side code. To run malicious scripts, the attack entails altering the page's Document Object Model (DOM).
* As an illustration, consider the following: An attacker can insert a script that modifies the user's browser's DOM and causes unwanted behaviors, such as data exfiltration, by manipulating a URL fragment (such as #hash).

Mitigation Measures

*Input Validation:*

* Justification: Prior to processing, make sure that every user input has been verified and cleaned. This entails verifying the accuracy of the incoming data and eliminating any possibly dangerous characters or scripts.
* Application: Make use of frameworks or libraries that manage input sanitization automatically. To escape user inputs, for instance, you can utilize JavaScript functions like encodeURIComponent.

*Output Encoding:*

* Explanation: Before displaying in the browser, make sure to encode the output data. This makes sure that any data that might be chtmlspecialcharse to run is instead seen as just regular text.
* Implementation involves using functions like htmlspecialchars in PHP or equivalent methods in different programming languages to prevent special characters like <, >, &, and " from being exploited in XSS attacks.

*Content Security Policy (CSP):*

* Clarification: Utilize a Content Security Policy to limit the varieties of content that can be loaded and run by a webpage. CSP helps reduce the impact of XSS attacks by stopping unauthorized scripts from running.
* Implementation: Configure your web server by setting up a Content Security Policy (CSP) header that dictates which sources are permitted for scripts, styles, and other content.

SQL Injection (SQLi)

SQL Injection (SQLi) is a type of attack where an intruder can manipulate the queries being sent to a database by an application. Attackers can achieve unauthorized access to or alter database data by inserting malicious SQL code.

*Types of SQL Injection In-band SQLi:*

* Definition: With this SQLi variation, the attacker utilizes the identical communication channel for both launching the attack and obtaining the results. This is the most widespread type and can be executed using error-based or union-based methods.
* An intruder enters a SQL payload into a login form in order to bypass authentication or retrieve data from the database.

*Inferential (Blind) SQLi:*

* Description: In Blind SQLi, the attacker is unable to view the query results firsthand. Rather, they deduce details by analyzing the app's reactions, like alterations in conduct or error alerts.
* An attacker could alter a query in order to verify whether specific conditions are accurate according to the application's feedback, deducing the database's structure and content.

*Out-of-band SQLi:*

* Out-of-band SQL injection refers to utilizing alternative communication channels for executing the attack and retrieving data. This technique is not frequently used but can be employed when other forms of SQL injection are not achievable.
* An attacker can transfer data to a server under their control through SQL queries and retrieve the data by making HTTP requests.

Mitigation Measures

*Prepared Statements:*

* Clarification: Prepared statements separate SQL code from data input by using parameterized queries. This blocks malicious SQL code injections from attackers in queries.
* Utilization: Utilize language-specific libramysqlir frameworks that provide support for prepared statements, like mysqli or PDO for PHP, or parameterized queries in SQL Server.

*Stored Procedures:*

* Definition: Stored procedures contain SQL queries within database routines, minimizing SQL injection risk by dividing data and code.
* Execution: Specify and execute stored procedures in your database, making sure they utilize parameterized inputs.

*Input Validation:*

* Explanation: Make sure to verify and clean all user inputs before utilizing them in SQL statements. Make sure that the inputs are in the correct formats and types as expected.
* Execution: Employ validation libraries or frameworks for verifying input data based on predetermined criteria and purify inputs to eliminate any potentially dangerous content.

Conclusion

Comprehending and reducing XSS and SQL Injection vulnerabilities are essential for creating secure web applications. Developers can safeguard their applications from common and dangerous attacks by utilizing appropriate input validation, output encoding, and employing prepared statements or stored procedures.

**TryHackMe SQLi**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Citation

Kyle, S. (2023, Jan 16.). What is cross-site scripting? XSS cheat sheet. Veracode. [URL.](https://www.veracode.com/security/xss)

Tim, T. (2023, March 20). What is SQL injection? tutorial & examples: Web security academy. What is SQL Injection? Tutorial & Examples | Web Security Academy. [URL.](https://portswigger.net/web-security/sql-injection)

Jain, S. (2024, March 18). Client-side form handling with JavaScript – explained with example code. freeCodeCamp.org. [URL.](https://www.freecodecamp.org/news/form-validation-in-javascript/)

**References**

XSS- <https://www.veracode.com/security/xss>

SQL Injection - <https://portswigger.net/web-security/sql-injection>

Form Validation - <https://www.freecodecamp.org/news/form-validation-in-javascript/>