# Udajuicer Website Downtime - Mitigation Plan

January 4, 2024

## Introduction

So far, we have broken down all the risks and in what order Udajuicer should mitigate them. This file discusses mitigation plans to get Udajuicer website back up and running.

- First thing to do is to construct a secure architecture. We will use draw.io for this purpose.
- The next thing would be to identify ways to mitigate HTTP Flood (DDOS) attack that caused the website downtime.
- We will finally list ways to mitigate SQL injection and XXS.

## 1 Secure Architecture Diagram using draw.io

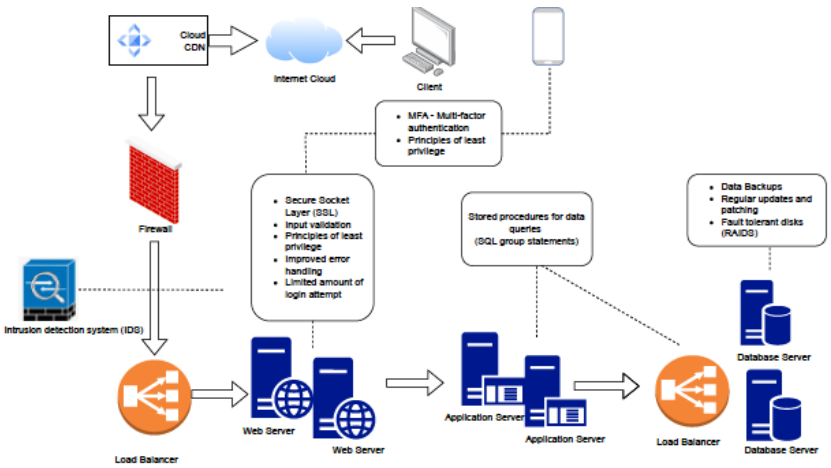


Figure 1: A secure architecture

## 2 HTTP Flood Attack / DDOS Attack Mitigation

Miitigating HTTP flood attacks involves implementing measures to protect web servers and applications from overwhelming traffic generated by a large number of requests. HTTP flood attacks aim to exhaust server resources, disrupt services, and potentially lead to service downtime. Here are key strategies to mitigate HTTP flood attacks:

- Implement rate limiting on your web server to restrict the number of requests from a single IP address or within a specific time frame. This helps prevent a single user or a group of users from overwhelming the server with excessive requests.

- Utilizing a Content Delivery Network to distribute incoming traffic across multiple servers located in different geographical regions to improve performance and security. CDNs can absorb and distribute the load, reducing the impact of a concentrated HTTP flood attack on a single server.
- Utilize load balancers to distribute incoming traffic across multiple application servers to ensure high availability and scalability. Load balancing helps evenly distribute the load, ensuring that no single server becomes a bottleneck during a flood attack.
- Deploy web application firewalls (WAF) to filter and monitor HTTP traffic between a web application and the internet to protect against common web application attacks. The WAF should be specifically designed to detect and mitigate various types of web attacks, including HTTP flood attacks. WAFs can analyze incoming traffic and block requests that exhibit patterns indicative of malicious activity.

# 3 SQL Injection Mitigation

A SQL injection attack is found when a SQL query is built up using user input. Mitigating SQL injection involves implementing security measures to prevent attackers from injecting malicious SQL code into input fields or other user-controllable parameters. Here are key strategies for mitigating SQL injection:

- Input Validation: Validate and sanitize all user input on the client and server sides. This ensures that only expected and valid data is accepted, preventing the injection of malicious SQL code.
- Use parameterized statements (prepared statements) or parameterized queries provided by the programming language or framework. Parameterization separates SQL code (queries) from user input (search term), preventing direct injection of malicious content.
- Stored Procedures: Use stored procedures whenever possible. Stored procedures encapsulate SQL logic on the database server, reducing the risk of injection attacks. Ensure that the stored procedures are well-designed and parameterized.
- Escape user input before incorporating it into SQL queries. This involves using proper escaping functions or libraries provided by the programming language. Escaping ensures that special characters are treated as literal values rather than executable SQL code.
- Least Privilege Principle: Implement the principle of least privilege when configuring database access permissions. This ensures that database accounts used by applications have the minimum necessary permissions to perform their tasks, reducing the impact of potential SQL injection attacks.

# 4 XSS - Cross Site Scripting Mitigation

As we have discussed, an XSS attack is a client-side attack. This means that, although the vulnerability is in the site itself, the targets are the users browsing the site. The attacker's goal is to run javaScript code in the browsers of the people visiting the site. Therefore mitigating XSS attacks would involve implementing security measures to prevent malicious scripts from being injected into web applications and executed by users' browsers. Here are key strategies for mitigating XSS attacks

- Sanitizing and validating all user input on both the client and server sides, checking for malicious scripts or coparing inputs against what is allowed. This includes data entered in forms, URL parameters, and any other user-controlled input. Reject or sanitize input that contains potentially malicious scripts. It is recommended to use a third party library and not try to implement your own filter.
- Conduct regular security audits and code reviews to identify and address potential security vulnerabilities, including XSS issues. Automated tools and manual reviews can help uncover vulnerabilities in the codebase.

- Implement session management best practices such as session timeout and token-based authentication.

## Reference

- Lecture note: SEC504: Hacker Tools, Techniques, and Incident Handling
- Lecture note: SEC401: Security Essentials - Network, Endpoint, and Cloud
- Lecture note: SEC275: Foundations: Computers, Technology, & Security