Udajuicer:
Threat Report

JORGE VERGARA

12/20/2022

Purpose of this Report:

This is a threat model report for **Udajuicer**. The report will describe the threats facing Udajuicer. The model will cover the following:

- 1.0 Threat Assessment
    - 1.1 Scoping out Asset Inventory
    - 1.2 Architecture Audit
    - 1.3 Threat Model Diagram
    - 1.4 Threats to the Organization
    - 1.5 Identifying Threat Actors
- Vulnerability Analysis
- Risk Analysis
- Mitigation Plan

Section 1

Threat Assessment

1.1: Asset Inventory

Components and Functions

The first part of our threat model is being able to identify all the assets involved in the target of the assessment. What does Udajuicer have of value? Looking at the architecture diagram, I have identified all the components that make up Udajuicer and documented them in the assessment scope of this report.

Database *- an information storage system that allows access through a website. The database stores information, such as usernames and passwords, to create a better user experience. MySQL is a common choice for a website database, whereas PHP is a server-side language for accessing the database.*

Application Server - *a software framework that delivers content and assets for a client application. Clients include web-based applications, browsers, and mobile apps. Application servers provide clients with access to business logic. Through business logic, an app server transforms data into dynamic content and enables the functionality of the application. Examples of dynamic content are: A transaction result, Decision support and Real-time analytics.*

Web Server - *is a computer system that stores, processes, and delivers web pages to clients. The client is almost always a web browser or a mobile application. Depending on the setup, a web server can store one or more websites. This type of server only delivers static HTML content, such as: Documents, Images, Videos and Fonts.*

1.1: Asset Inventory

Explanation of How A Request Goes from Client to Server

a client sends a request to the server, which performs some action and sends a response back to the client, which could be a result or an acknowledgement (like an error message etc.)

The web server or HTTP server is a server software (or hardware dedicated to running a server software) that implements the request/response model using the World Wide Web and the HTTP (protocol.)

The HTTP server processes incoming network requests from the clients over the HTTP protocol and serves contents over the internet.

The HTTP or Hypertext Transfer Protocol is an application layer protocol that is used to virtually transmit files and other data on the World Wide Web, whether they're HTML files, image files, query results, or anything else.

The primary function of the HTTP server is to store, process and deliver web pages to clients using the Hypertext Transfer Protocol.

A client, or better said a user agent which is a software acting on behalf of a user (commonly represented by a web browser), initiates communication by making a request for a specific resource using HTTP to the server, an application running on a computer hosting a website.

The server, which provides resources such as HTML files and other content, or performs other functions on behalf of the client, returns a response message to the client.

The response served by the server contains completion status information about the request and may also contain requested content in its message body, or an error message if it is unable to serve the content.

1.2 Architecture Audit

Flaws

- *FLAW 1 – Needs external firewall*
- *FLAW 2 – Needs IPS intrusion detection system*
- *FLAW 3 – Needs Load Balanacer*
- *[Add extra flaws as necessary]*

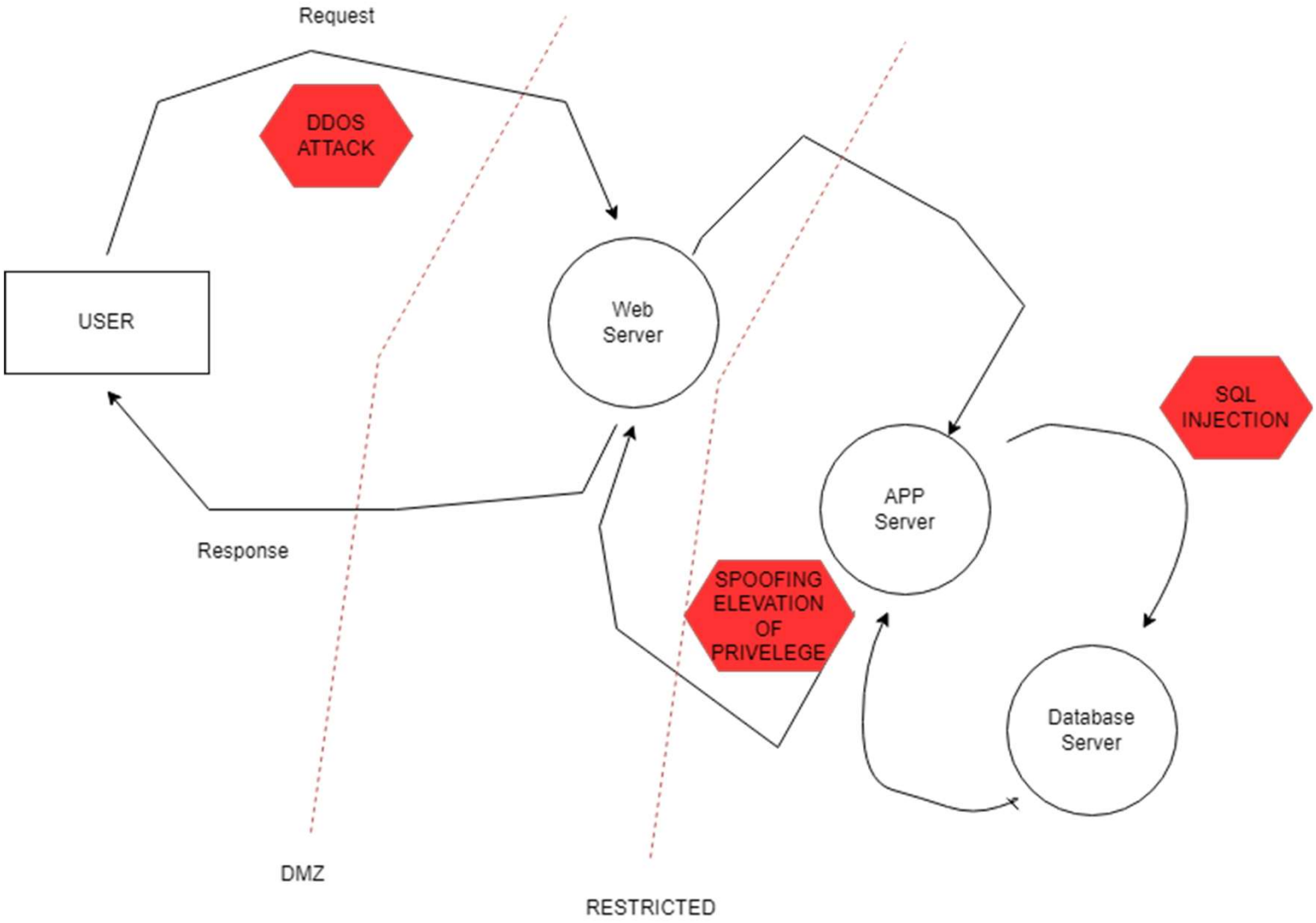
1.3 Threat Model Diagram

Using OWASP Threat Dragon, build a diagram showing the flow of data in the Juice Shop application and identify 3 possible threats to the Juice Shop. Make sure to include the following components:

- **Client**
- **Web Server**
- **Application Server**
- **Database**

1.3 Threat Model Diagram

Insert threat Model Diagram Here:



1.4 Threat Analysis

What Type of Attack Caused the Crash?

How it was originally set up, a simple DDOS attack definitely caused the crash.

What in the Logs Proves Your Theory?

By looking at the logs, I can clearly see that various if not many botnets were used to orchestrate a DDOS attack. Exactly the same date and time all these botnets decided to send traffic to the website.

1.5 Threat Actor Analysis

Who is the Most Likely Threat Actor?

Script Kiddies -Amateur cyber criminals with unclear motivations. They can be actual teenagers using scripts made by other individuals. using prebuilt tools and script that they most likely downloaded from the dark web.

What Proves Your Theory?

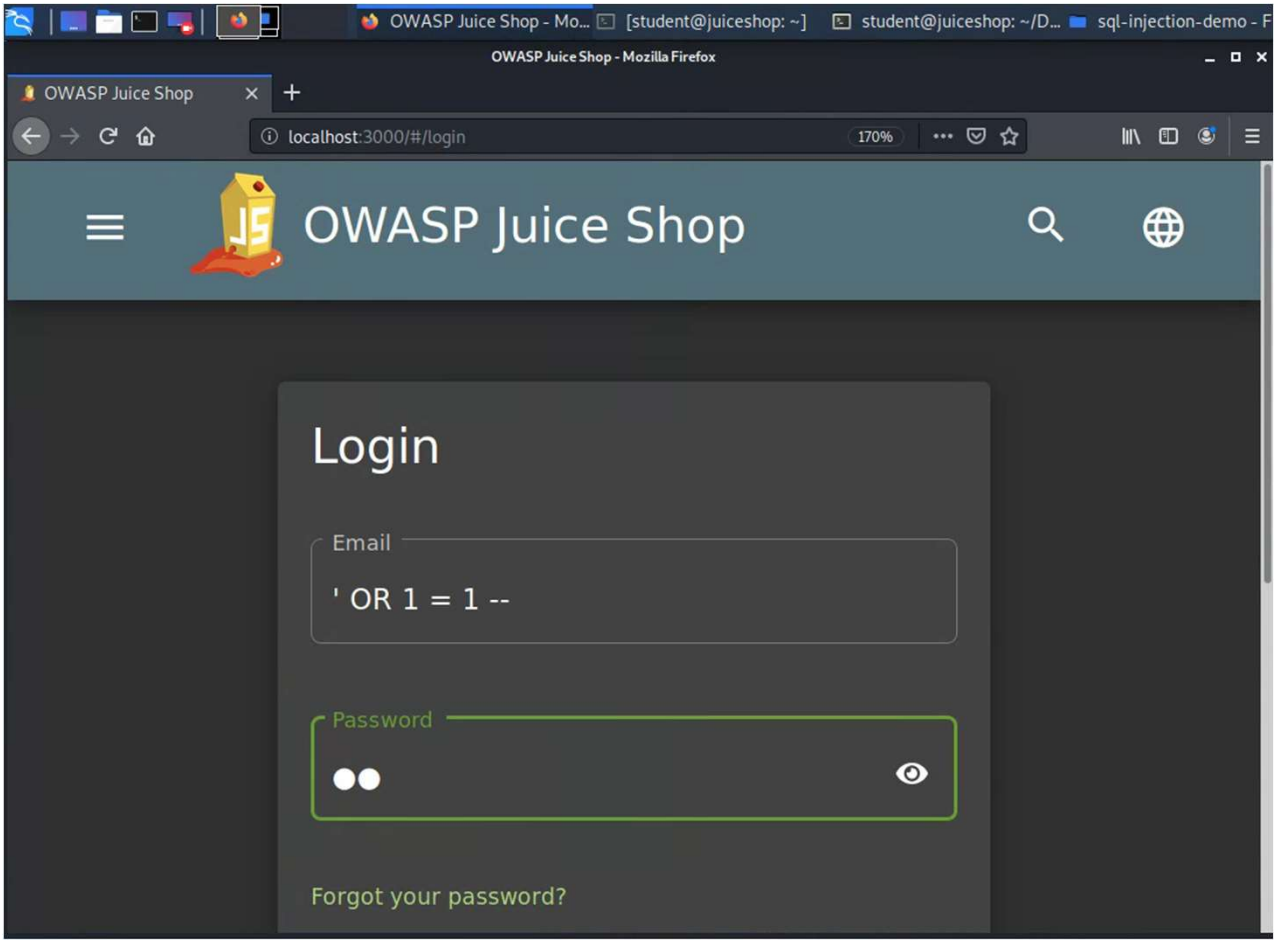
Script Kiddies – Most likely an amateur cyber criminal choosing this independent small business just to test their cyber attack skills, no ransom or sensitive info retrieval just basic malicious techniques and see what it does to the website. Definitely no motive or financial gains behind it, it is not that difficult to execute a DDoS attack. Very begginner skill at work here. No state sponsored or hacktivist would be behind this.

Section 2

Vulnerability Analysis

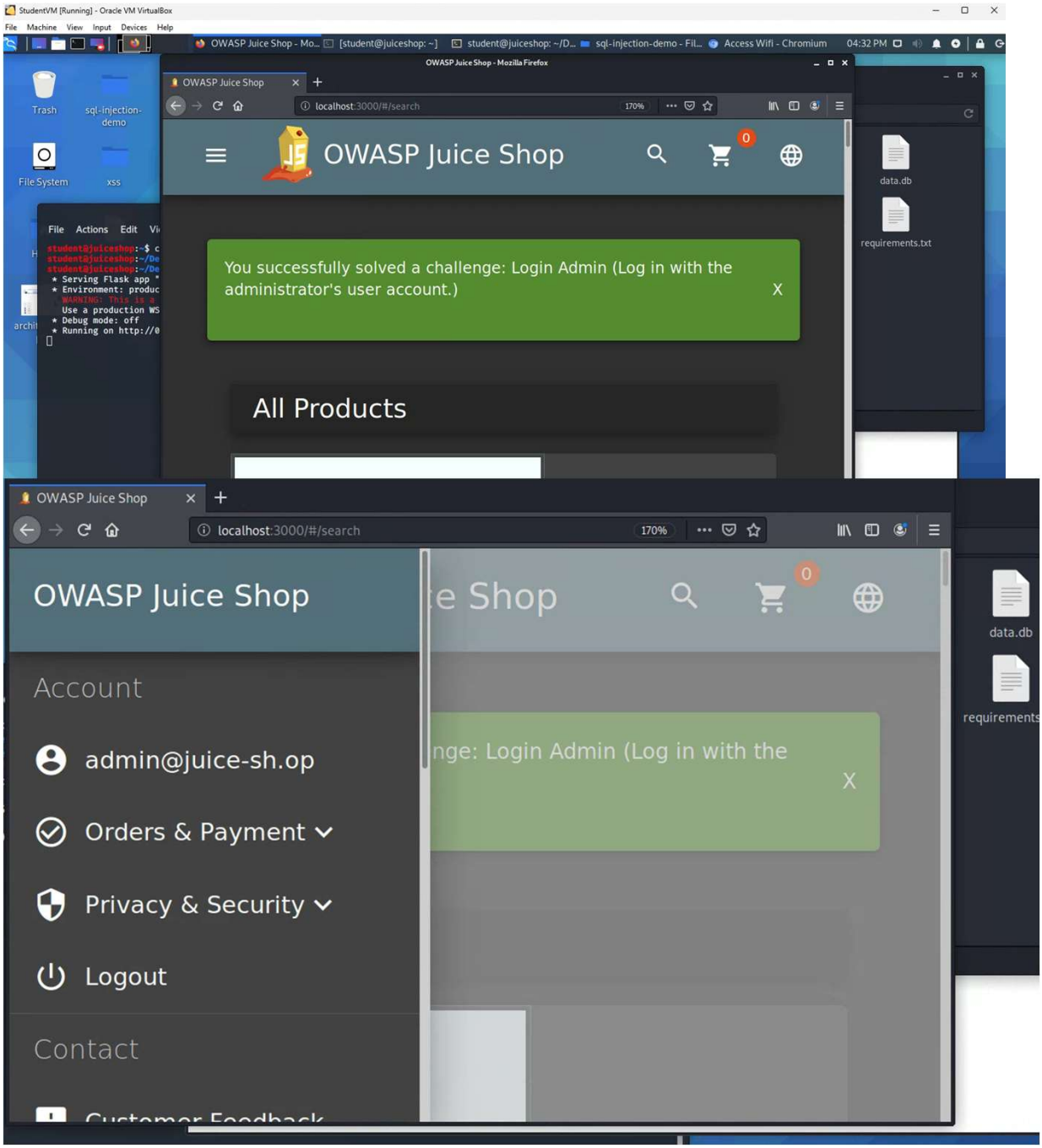
2.1 SQL Injection

Insert Screenshot of Your Commands Here:



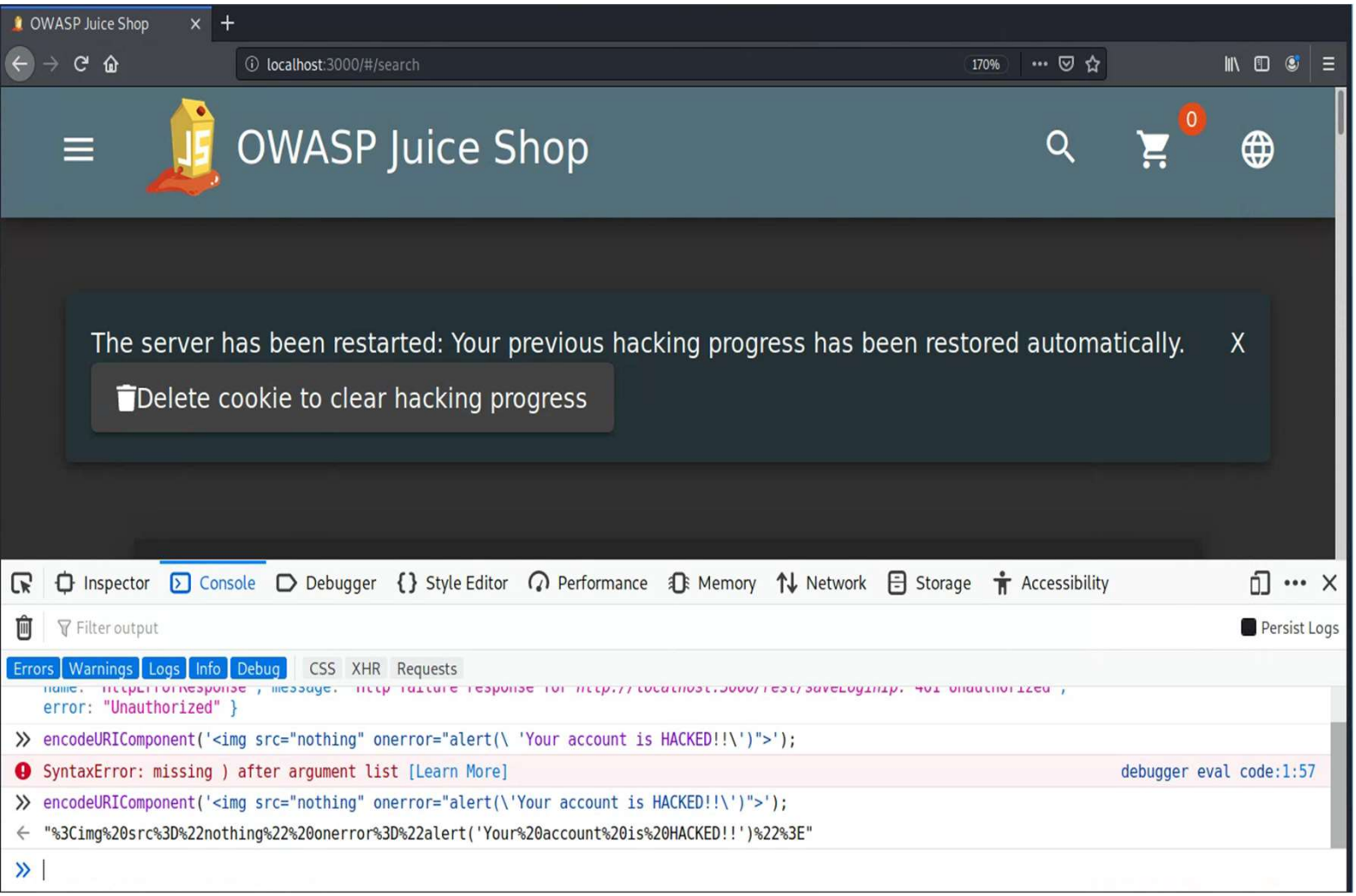
2.1 SQL Injection

Insert Screenshot of Account Settings Showing You as Admin Here:



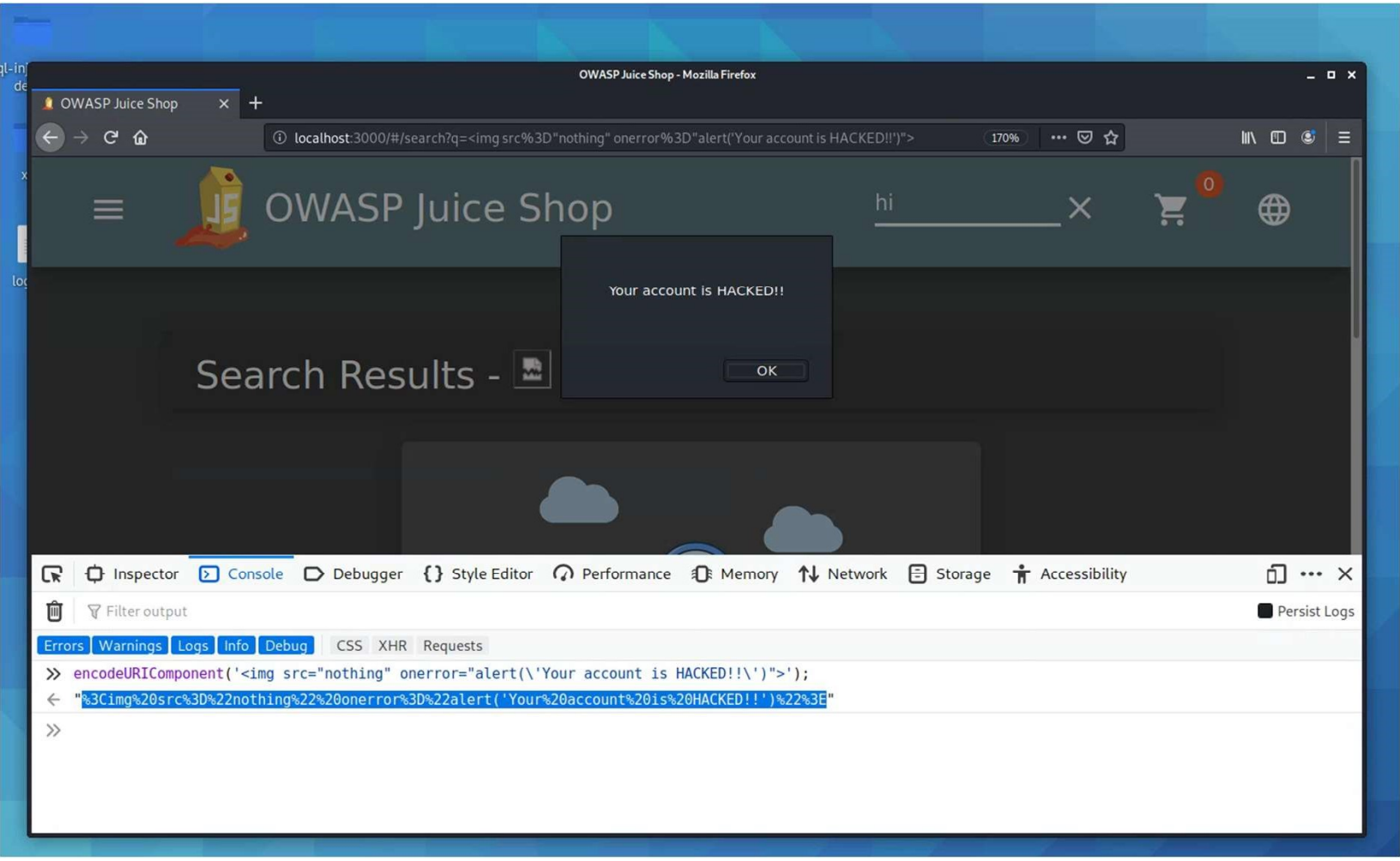
2.2 XSS

Insert Screenshot of Your Commands Here:



2.2 XSS

Insert Screenshot of `alert()` popup saying "Hacked!" Here:



Optional Task:

Extra Vulnerabilities

- *[Vulnerability 1 Here]*
- *[Vulnerability 2 Here]*
- *[Add more vulnerabilities as necessary]*

Section 3

Risk Analysis

3.1 Scoring Risks

Risk	**Score** ***(1 is most dangerous, 4 is least dangerous)***
DOS ATTACK	1
Insecure Architecture	4
SQL Injection	2
XSS Vulnerability	3

3.2 Risk Rationale

Why Did You Choose That Ranking?

2. SQL injection is a specific type of attack that occurs when an attacker is able to inject malicious SQL code into a database through a vulnerability in the application. The attacker can then extract sensitive information or make unauthorized changes to the data. SQL injection attacks can have serious consequences for the data and the integrity of the system, but they are typically limited to the specific application or database that is vulnerable to the injection.

4. Insecure architecture refers to poor design decisions in the architecture of a system or application that can lead to security vulnerabilities. Examples include using weak encryption methods, not properly securing sensitive data, or not properly controlling access to resources. These types of vulnerabilities can be difficult to detect and can have wide-reaching consequences, potentially compromising the entire system.

3. XSS (Cross-Site Scripting) vulnerability allows an attacker to inject malicious code into a web page viewed by other users. This vulnerability can be used to steal sensitive information or to take control of a user's browser. XSS vulnerability can be used to steal sensitive information from the users who visit the infected website.

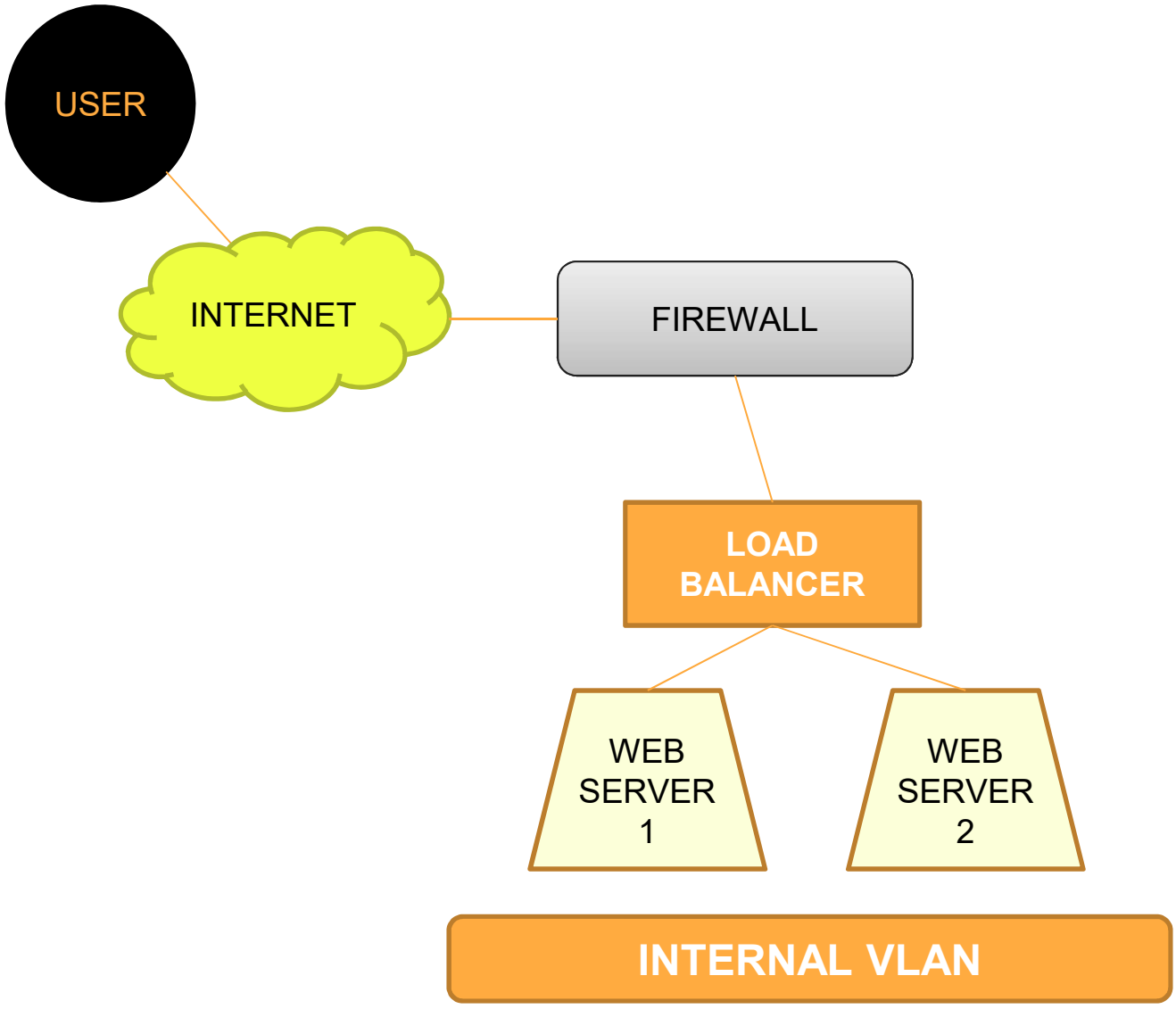
1. DOS Attack - an attack meant to shut down a machine or network, making it inaccessible to its intended users. DoS attacks accomplish this by flooding the target with traffic, or sending it information that triggers a crash. In both instances, the DoS attack deprives legitimate users (i.e. employees, members, or account holders) of the service or resource they expected. But from the content on udacity, a DDoS attack will affect availibility if impacted hard. Control and containment attacks will be difficult, which is the risk that will cause udajuicer system to crash completely.

Section 4

Mitigation Plan

4.1 Secure Architecture

Insert Image of Your Secure Architecture Here:



4.2 Mystery Attack Mitigation

What is Your Mitigation Plan?

Since DDOS comes from a series of botnets. The mitigation strategy for this issue can first go to the firewalls. By setting special type of rules and monitor the patterns of the traffic on the firewall or on an Access Control Lists (ACLs) to control what traffic reaches you're application.

Server capacity - Most DDoS attacks are volumetric attacks that use up a lot of resources; it is, therefore, important that you can quickly scale up or down on your computation resources. You can either do this by running on larger computation resources or those with features like more extensive network interfaces or enhanced networking that support larger volumes. Additionally, it is also common to use load balancers to continually monitor and shift loads between resources to prevent overloading any one resource.

Whenever you detect elevated levels of traffic hitting a host, the very baseline is to be able only to accept as much traffic as our host can handle without affecting availability. This concept is called **rate limiting**. More advanced protection techniques can go one step further and intelligently only accept traffic that is legitimate by analyzing the individual packets themselves. To do this, you need to understand the characteristics of good traffic that the target usually receives and be able to compare each packet against this baseline.

4.3 SQL Injection Mitigation

What is Your Mitigation Plan?

We cant trust any user input. We have to treat all user input as untrusted. Any user input that is used in an SQL query introduces a risk of an SQL Injection. Treat input from authenticated and/or internal users the same way that you treat public input.

We should use whitelists, not blacklist. We cannot filter user input based on blacklists. A clever attacker will almost always find a way to circumvent your blacklist. If possible, verify and filter user input using strict whitelists only.

We must adapt and adopt the latest technology updates. Older web development technologies don't have SQLi protection. If we use the latest version of the development environment and language and the latest technologies associated with that environment/language. For example, in PHP use PDO instead of MySQLi.

Enforce Prepared Statements And Parameterization. Prepared statements provide a fundamental and critical defense against SQL injection vulnerabilities. Where possible, developers should attempt to implement prepared statements so that a database will treat malicious SQL statements as data and not as a potential command.

Don't Divulge More Than Necessary In Error Messages. SQL injection attackers can learn a great deal about database architecture from error messages. To block exploration of this type, ensure that error messages display minimal information.

4.4 XSS Mitigation

What is Your Mitigation Plan?

Input validation: Ensure that all user input is properly sanitized and validated to prevent the injection of malicious code.

Escaping: Use proper escaping mechanisms for user input when displaying it on a web page.

Content Security Policy (CSP): Use a Content Security Policy (CSP) to specify which sources of content are allowed to be loaded by the browser.

Use a framework that automatically escapes output: Many web frameworks have built-in mechanisms to automatically escape output and help prevent XSS attacks.

Use a security scanner to identify and fix vulnerabilities: Use a security scanner to identify and fix vulnerabilities in your web application.

Keep software up-to-date: Keep all software, including web browsers, frameworks, and libraries, up-to-date to ensure that any known vulnerabilities are patched.

Educate your employees: Educate your employees about the risks of XSS and the importance of keeping software up-to-date and following best practices for secure coding.