Udajuicer:

Threat Report

Simone Canty

August 09, 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:

- Threat Assessment
  - Scoping out Asset Inventory
  - Architecture Audit
  - Threat Model Diagram
  - Threats to the Organization
  - Identifying Threat Actors
- Vulnerability Analysis
- Risk Analysis
- Mitigation Plan

Section 1

Threat Assessment

1.1: Asset Inventory

Components and Functions

- **Web Server*:***
  - to display the website content through storing, processing and delivering webpages to the client/user.
  - With Udajuicer website, it can have the user add products to a shopping cart, payment details, or create an account..
- **Application Server:**
  - when clients access the Ujuice application, it is being pulled from an application server and delivered via a web server.
  - It request the inventory database to return product availability or add details to a customer’s profile..
- **Database Server:**
  - is to store aggregations of data records or files.

1.1: Asset Inventory

Explanation of How A Request Goes from Client to Server

1. The user/client accesses the Juice Shop through a web browser by a request to the Web Server using the Hypertext Transfer Protocol (HTTP) or HTTP Request.

2. The Web Server will respond to the request by sending back the website's content or files through the HTTP Response to the web browser.

3. If Web Server needs help in processing the files or content, it will send a servlet or binary request to the Application server and the Application server will communicate with the Database for the requested data. The Application server will send a servlet or binary response to the Web Server.

4. The Web Server will send the complex content to the Web Browser via HTTP response.

1.2 Architecture Audit

Flaws

- No Content Delivery Network:
- No Load Balancer with more than one Web Server, Application Server, and Database Sever.
- No Data Backups

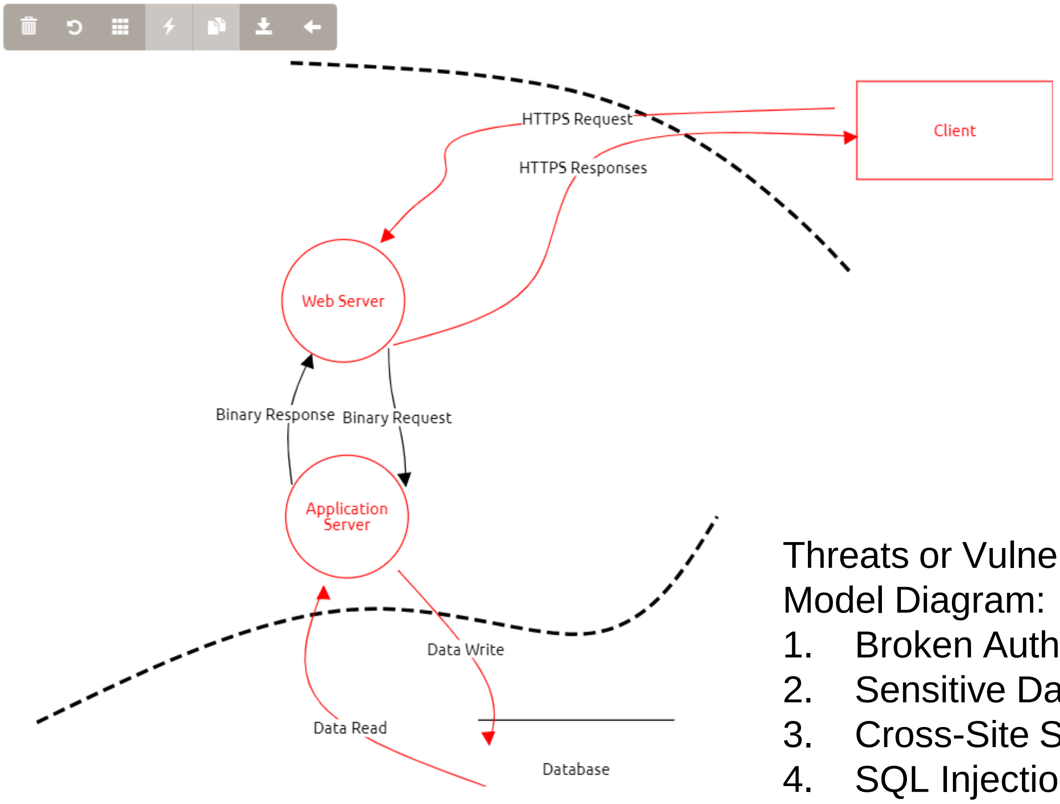
1.3 Threat Model Diagram

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:



Threats or Vulnerabilities given the Threat Model Diagram:

1. Broken Authentication
2. Sensitive Data Exposure
3. Cross-Site Scripting
4. SQL Injection

1.4 Threat Analysis

What Type of Attack Caused the Crash?

A Distributed Denial of Service (DDoS) attack.

What in the Logs Proves Your Theory?

There a multiple machines sending HTTP flood at the same time and the same day.

1.5 Threat Actor Analysis

Who is the Most Likely Threat Actor?

Script Kiddie

What Proves Your Theory?

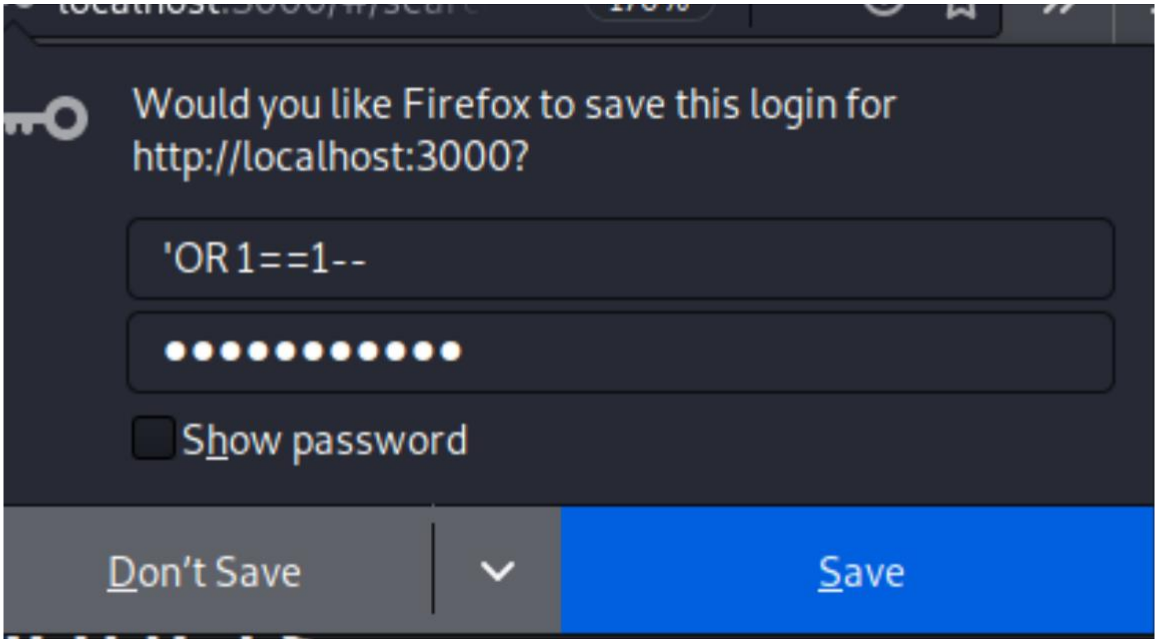
There were no clear motivate. It is most likely an amateur using an open source tool on how to conduct DDoS attacks.

Section 2

Vulnerability Analysis

2.1 SQL Injection

Insert Screenshot of Your Commands Here:

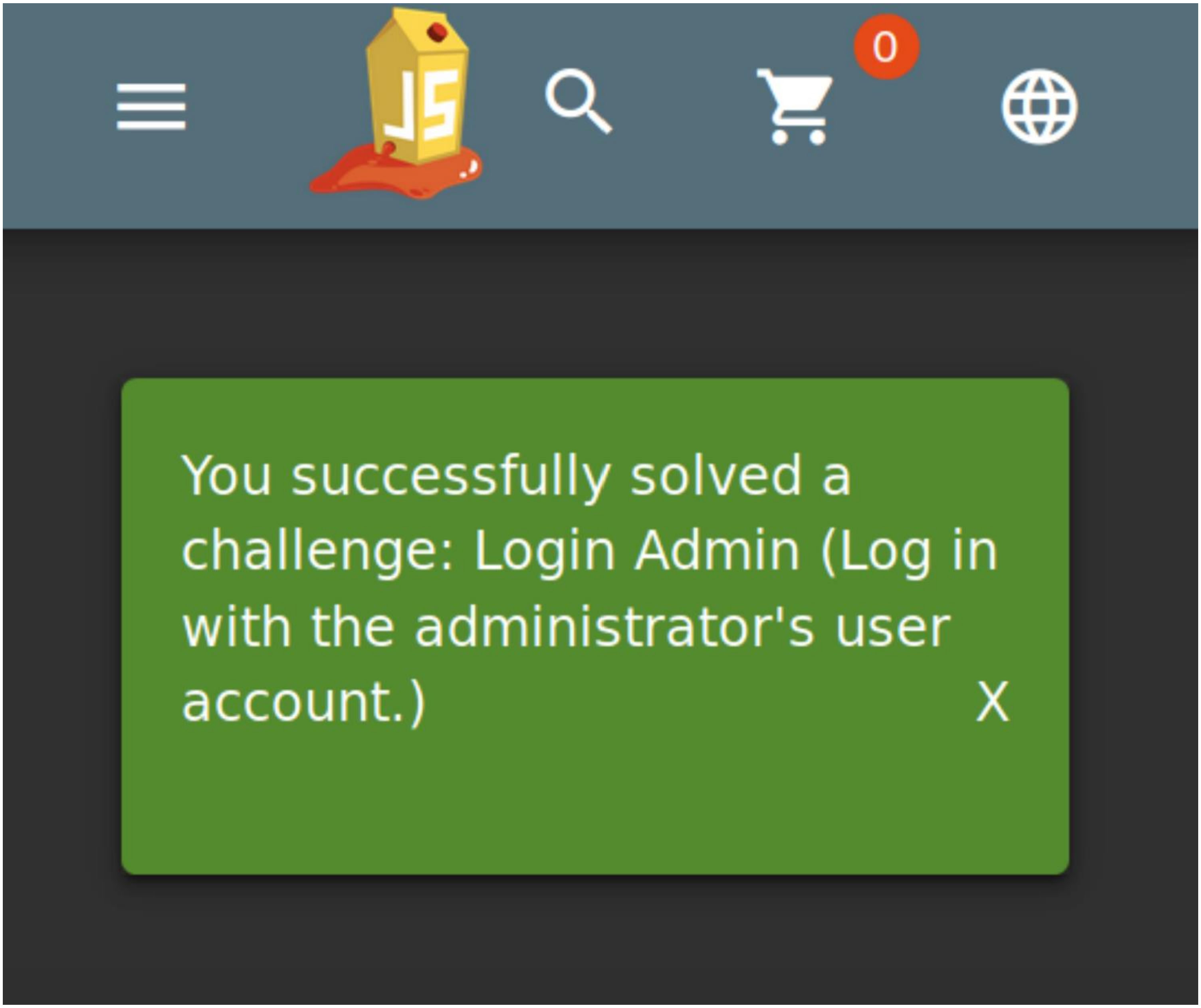


Would you like Firefox to save this login for http://localhost:3000?

'OR 1==1--

●●●●●●●●●●●

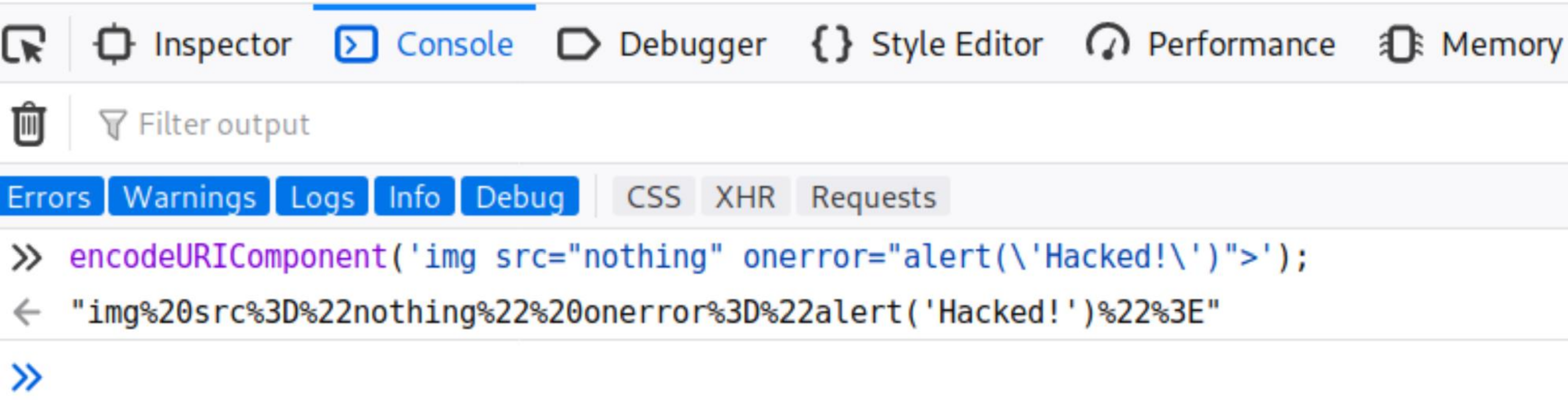
2.1 SQL Injection



You successfully solved a challenge: Login Admin (Log in with the administrator's user account.) X

2.2 XSS

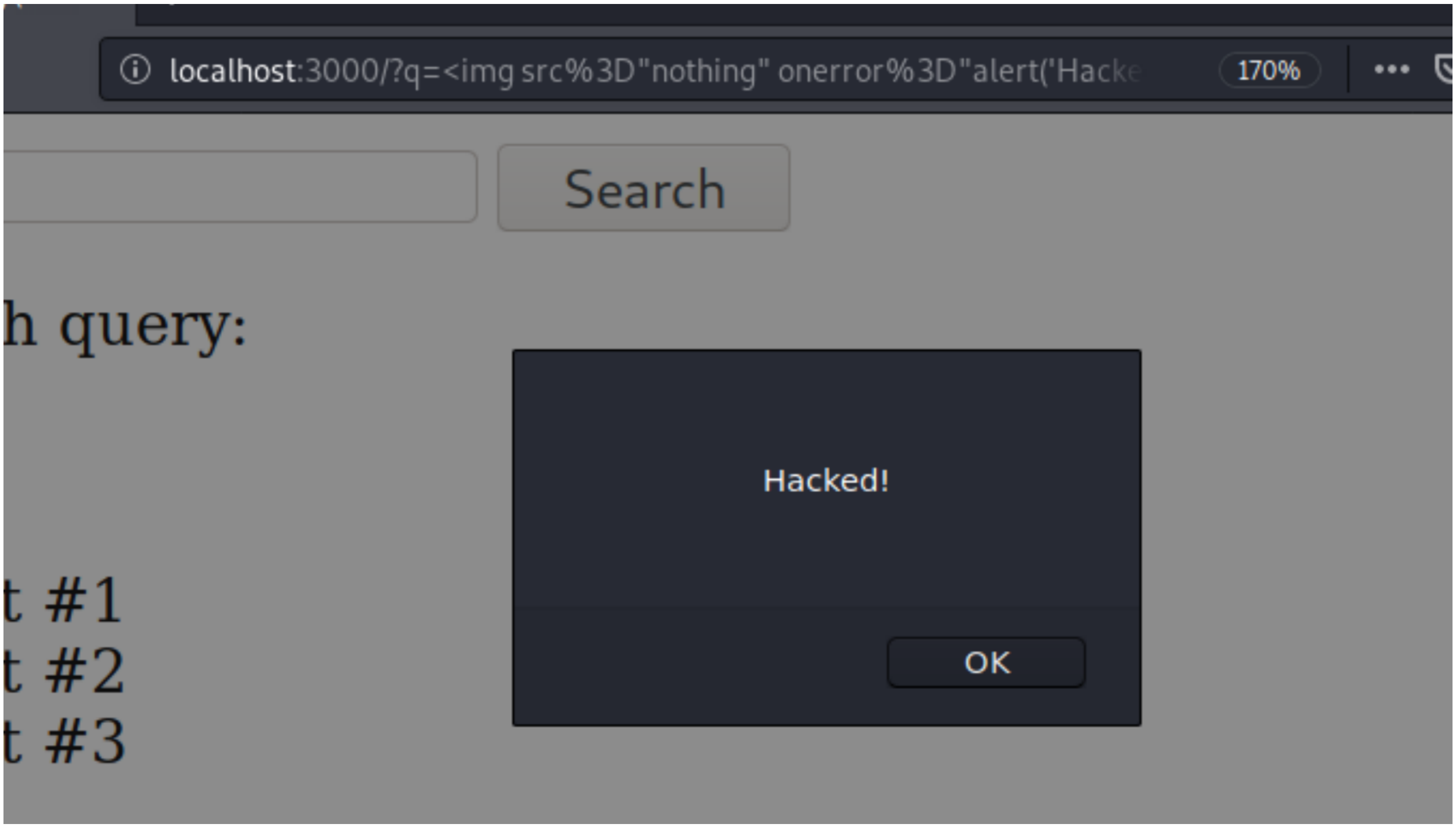
Insert Screenshot of Your Commands Here:



```
>> encodeURIComponent('img src="nothing" onerror="alert(\'Hacked!\')">');
← "img%20src%3D%22nothing%22%20onerror%3D%22alert('Hacked!')%22%3E"
>>
```

2.2 XSS

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



localhost:3000/?q=<img src%3D"nothing" onerror%3D"alert('Hacke

Hacked!

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)***
Distributed Denial of Service (DDoS)	1
Insecure Architecture	2
SQL Injection	3
XSS Vulnerability	4

3.2 Risk Rationale

Why Did You Choose That Ranking?

Distributed Denial of Service(DDoS):

- The site would constantly go down, affecting the availability of the site. In addition, this affects the business' finances and reputation.

Insecure Architecture:

- The site not having a secure architecture can affect the confidentiality, availability, and integrity of the Udajuicer website. As per written, "

SQL Injection:

- This injection can affect the integrity and confidentiality of the Udajuicer website. The attacker can gain access to sensitive data such as credit card information, personal information, etc.

Section 4

Mitigation Plan

4.1 Secure Architecture

Insert Image of Your Secure Architecture Here:

Please refer to the PDF

4.2 Mystery Attack Mitigation

What is Your Mitigation Plan?

To prevent further Distributed Denial of Service (DDoS) attack:

- To implement an internal and external firewall. A firewall will help to monitor network traffic.
- Set up a firewall rule, to allow and block the traffic based on the ports and protocols.
- Implement a Content Delivery Network (CDN). This will help with:
    - Improving website security and reducing DDoS attacks
    - Improving website load times:
    - Increasing content availability and redundancy

4.3 SQL Injection Mitigation

What is Your Mitigation Plan?

What is Your Mitigation Plan?

- [**Securing User Input**

  - **Input Sanitization**: involves taking user input and checking, cleaning, and filtering data from users for unwanted characters and strings. To work with the application/software developer to disallow content to show an error if the user is entering bad content.

  - **Input Validation:** process of taking user input and matching it against an allow list or expected input. Again, work with the application/software developer to assure there is code written in the software for this**.**

This will prevent untrusted data when it is sent to an interpreter will not trick the interpreter into executing unintended commands or accessing data without proper authorization.

- **Escaping**

  - This will prevent the attacker from executing scripts in the user's browser which can hijack user sessions, redirect users to a potentially malicious web sites, or vandalize the web site**.**

4.4 XSS Mitigation

What is Your Mitigation Plan?

- [**Securing User Input**

  - **Input Sanitization**: involves taking user input and checking, cleaning, and filtering data from users for unwanted characters and strings. To work with the application/software developer to disallow content to show an error if the user is entering bad content.

  - **Input Validation:** process of taking user input and matching it against an allow list or expected input. Again, work with the application/software developer to assure there is code written in the software for this**.**

This will prevent untrusted data when it is sent to an interpreter will not trick the interpreter into executing unintended commands or accessing data without proper authorization.

- **Escaping**

  - This will prevent the attacker from executing scripts in the user's browser which can hijack user sessions, redirect users to a potentially malicious web sites, or vandalize the web site**.**