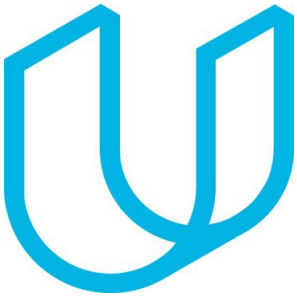
Udajuicer: Threat Report

Michael Wolff

10.03.2021



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

1. ***Web Server:***
   The web server is a computer that stores web server software and a website's component files, for example, HTML documents, CSS stylesheets, and JavaScript files. A web server connects to the Internet and supports physical data interchange with other devices connected to the web.

2. ***Application Server:***
   Sitting between the primary web-based server tier and the back-end tier of a database, an application server serves dynamic content and is a software framework which provides all the facilities required to create and run both web based and enterprise based applications. It is best suited for serving dynamic content and transferring applications from one device to another.

3. ***Database:***
   A database is an organized collection of structured information, or data, typically stored electronically in a computer system.

1.1: Asset Inventory

Explanation of How A Request Goes from Client to Server

1. The client's device is connected to the Internet and the browser requests an URL, in our case the URL of Udajuicer.

2. This request is sent to the public internet via ISP. The DNS lookup turns the URL into the corresponding IP address. This is the network address of the web server that is hosting Udajuicer.

3. The web server will usually fetch the requested file, and will send it back. If in our case the requested file is a script file, the web server knows to hand off the request to the application server that will process the script code.

4. The application server receives an HTTP Request from the web server, finds the file with the source code and executes the script program.

5. The program references a connection to a database and sends a query to request data stored and organized by the database server.

6. The database server fetches the files requested from the query.

7. The database server sends back the requested data to the application server.

8. The application server combines files if necessary and sends it back to the web server as an HTTP Response.

9. The web server sends back the HTML text to the requesting connection.

10. The data is transmitted over the Internet back to the client's device and the connection ends.

11. The device browser reads the generated HTML code and displays it on the screen.

1.2 Architecture Audit

Flaws

The following flaws have been discovered:

- The architecture has no **Firewall or IDS.** Firewalls serve as a critical security foundation. An Intrusion Detection System (IDS) is a system that monitors network traffic for suspicious activity and issues alerts when such activity is discovered.
- Improved **Error Handling** would prevent the attacker from gathering information from errors. Instead of showing the error, the website could present a **generic** error web page.
- Missing **input validation**. Data needs verification at input, before being passed on.
- Missing **stored parameterized procedures** (SQL group statements).
- Files on server (e.g ftp, administration) are accessible to everyone with no restrictions. **Restricting file and folder access** to specific groups or individual users prevent file server access exploitation.
- Introducing load balancing and sufficient bandwidth.
- Fault tolerance through database backups.

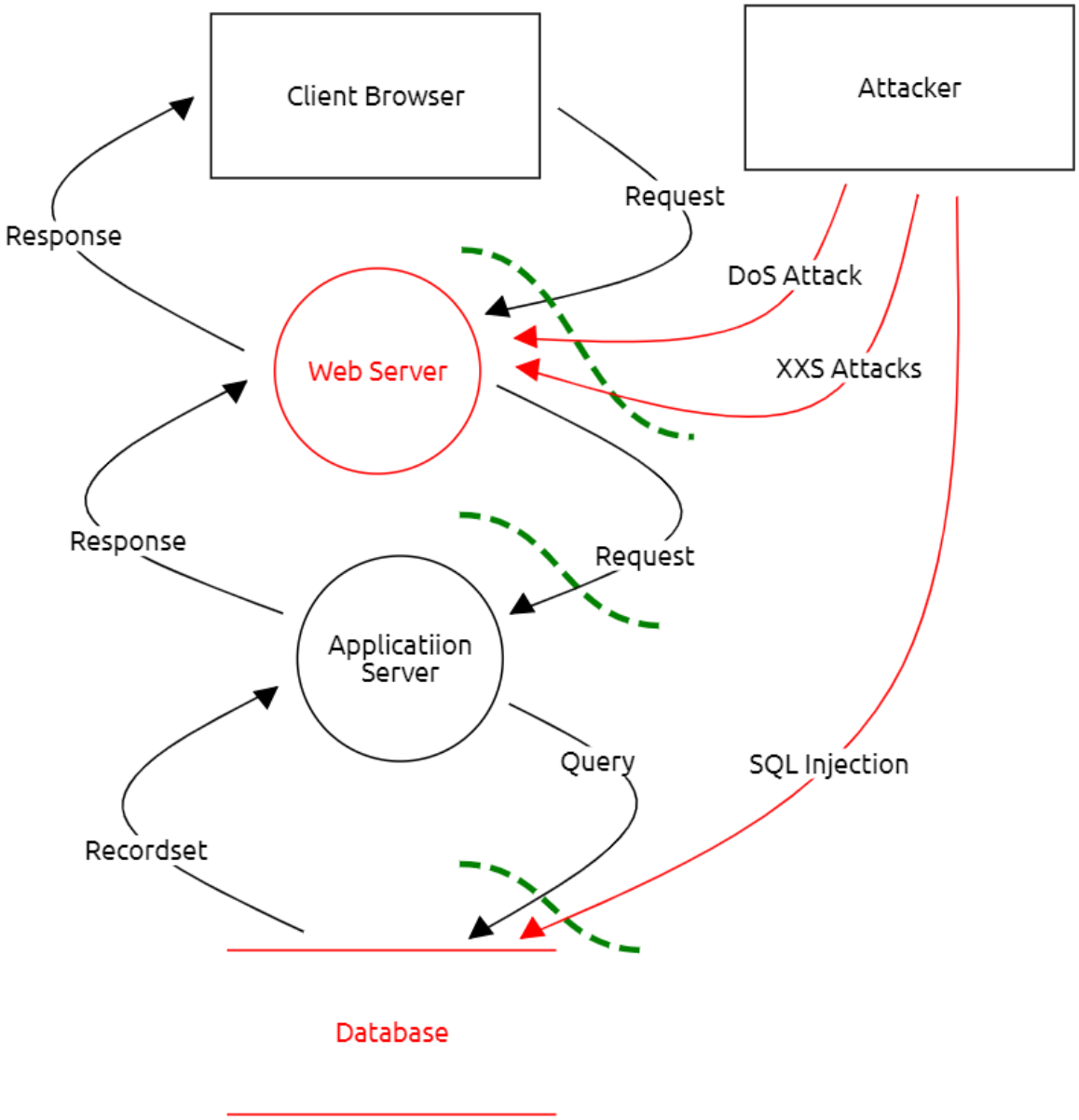
1.3 Threat Model Diagram

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 and Possible Threats Here:



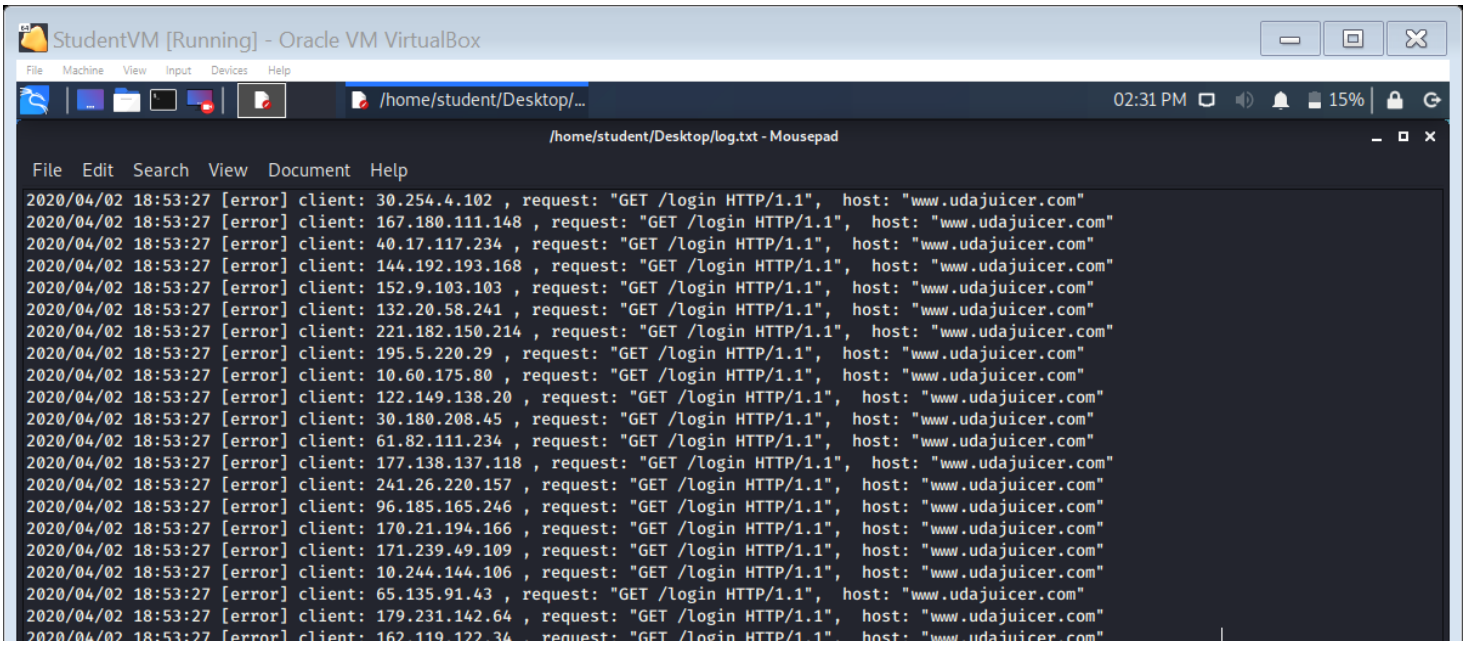
1.4 Threat Analysis

What Type of Attack Caused the Crash?

I believe it was a HTTP flood / DDoS attack which crashed the Udajuicer store. HTTP flood is a type of Distributed Denial of Service (DDoS) attack in which the attacker exploits seemingly-legitimate HTTP GET or POST requests to attack a web server or application. HTTP flood attacks are volumetric attacks, often using a botnet—a group of Internet-connected computers

What in the Logs Proves Your Theory?

The log file shows numerous requests at the exact same time from various IP addresses, from which I conclude that it was a DDoS attack. It also shows “GET” entries. HTTP clients like a web browser talk to applications or servers by sending HTTP requests – generally one of two types of requests: GET or POST. Perpetrator using HTTP Flood attacks aim to inundate the server or application with multiple requests that are each as processing-intensive as possible.



1.5 Threat Actor Analysis

Who is the Most Likely Threat Actor?

In my opinion the most likely threat actor would be a “Script Kiddie”. I believe the attacks came from a rather inexperienced attacker using the Distributed Denial of Service attack, while the application architecture would allow easier ways to gain access to sensible data.

What Proves Your Theory?

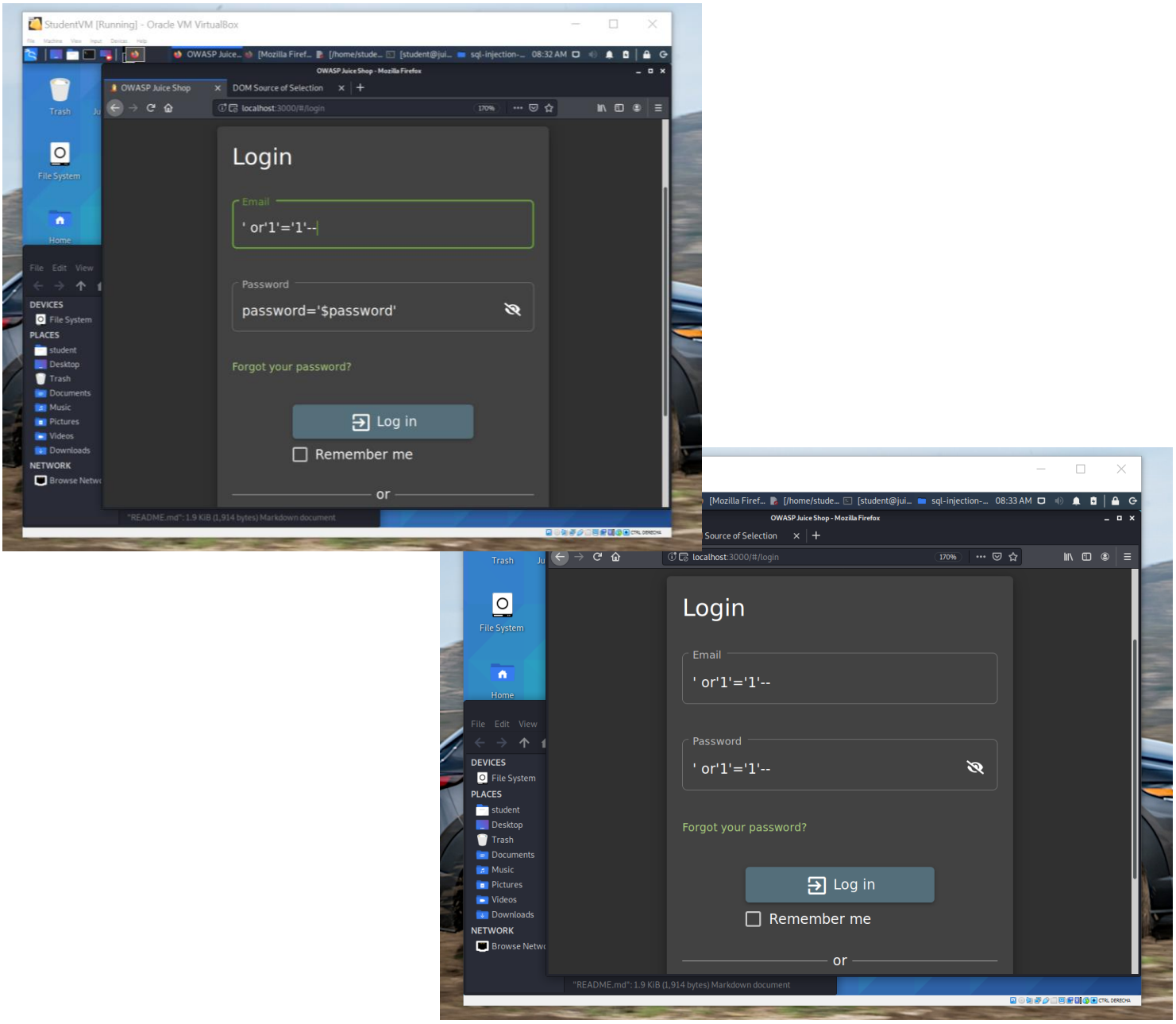
An experienced hacker would probably try another way of gaining access that leaves less evidence behind and is faster e.g. SQL injections. Experienced hacker might delete any evidence after the attack. We can exclude organized crime, APT groups or State-Sponsored attacks in my opinion, as the shop does not contain enough interesting information for these kind of organisations. Insider would probably have access to passwords and might not know how to access the system otherwise nor how to use any hacker tools.

Section 2

Vulnerability Analysis

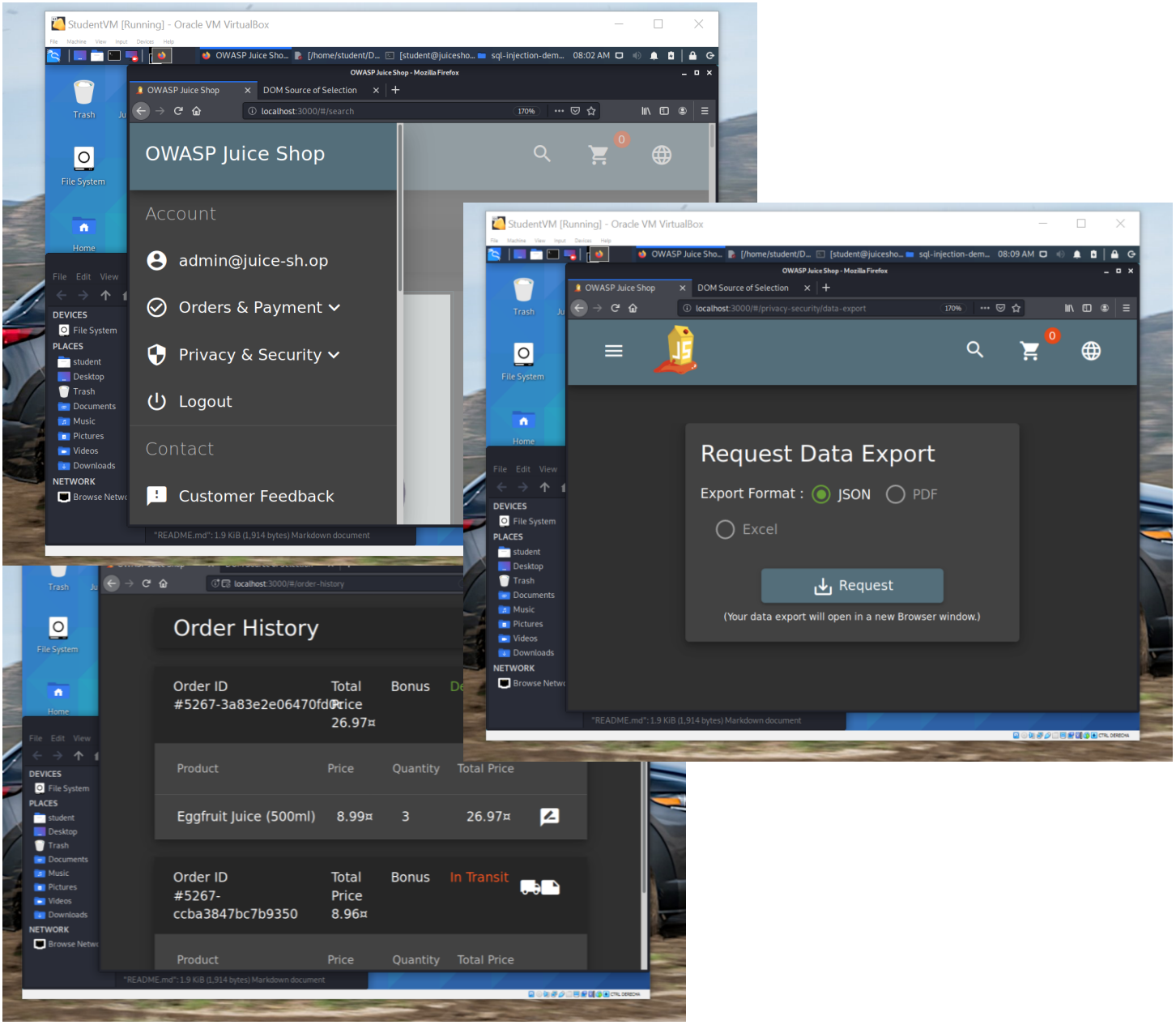
2.1 SQL Injection

Insert Screenshot of Your Commands Here:



2.1 SQL Injection

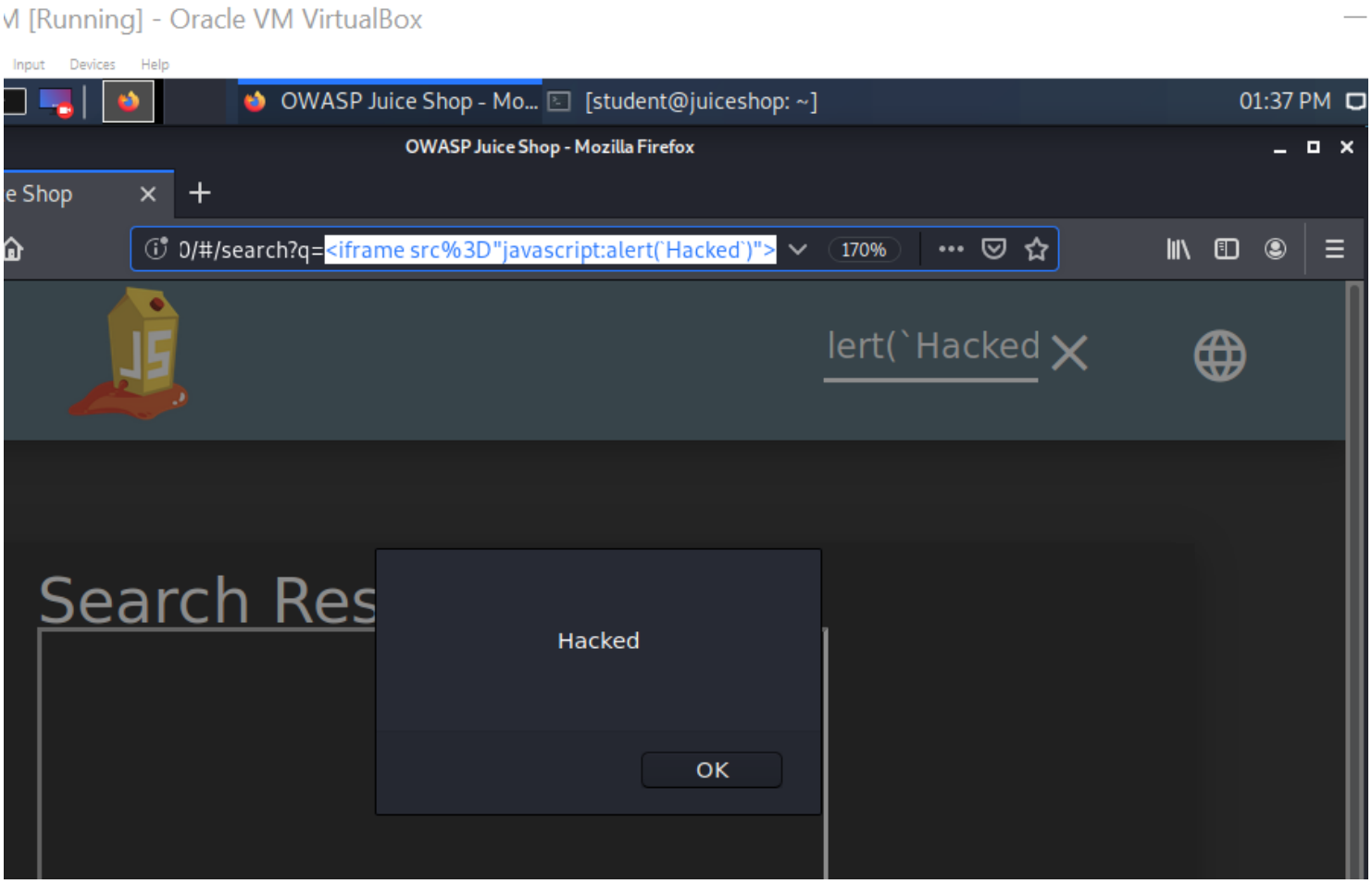
Insert Screenshot of Account Settings Showing You as Admin Here:



2.2 XSS

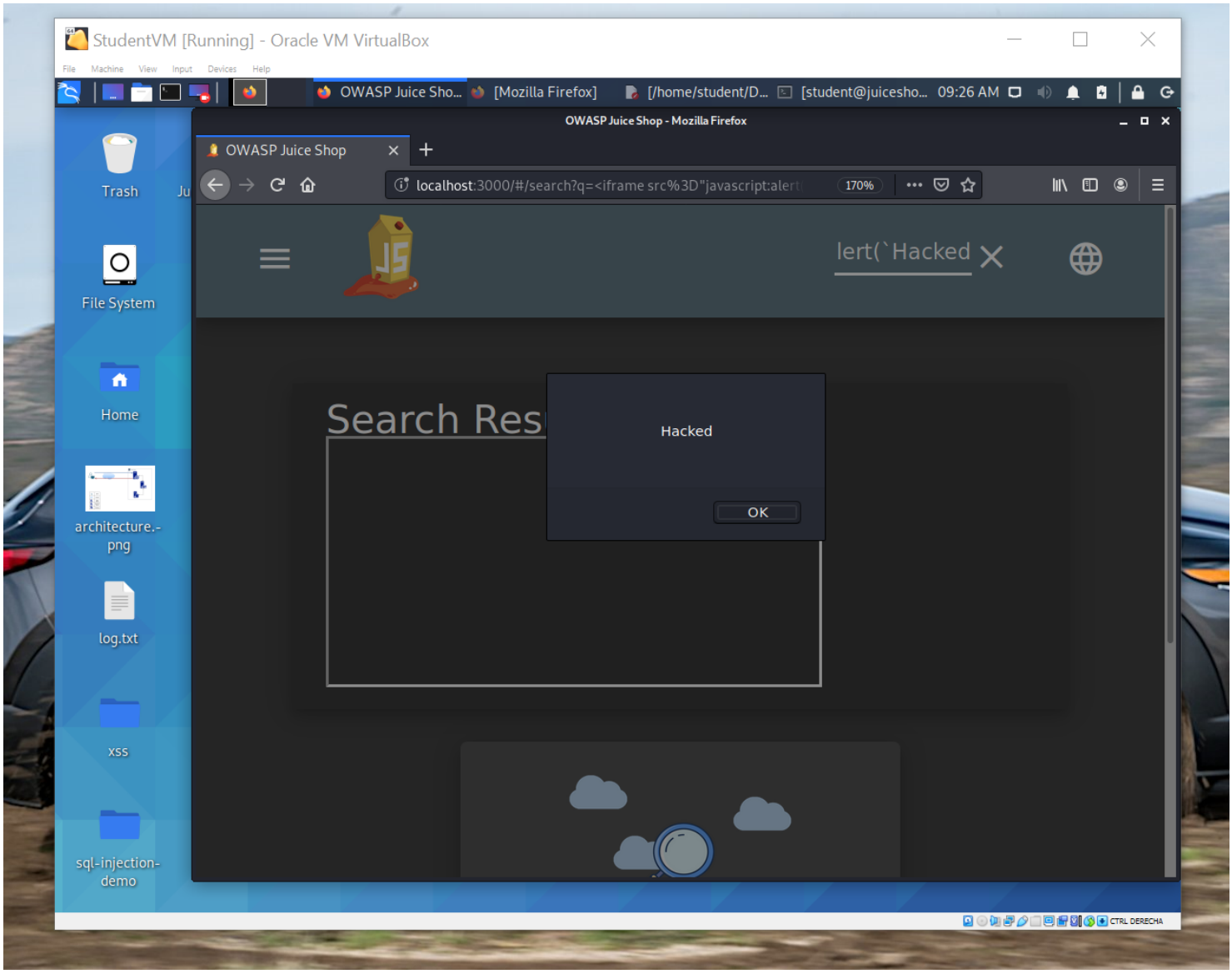
Insert Screenshot of Your Commands Here:

<iframe src="javascript:alert(`Hacked`)">



2.2 XSS

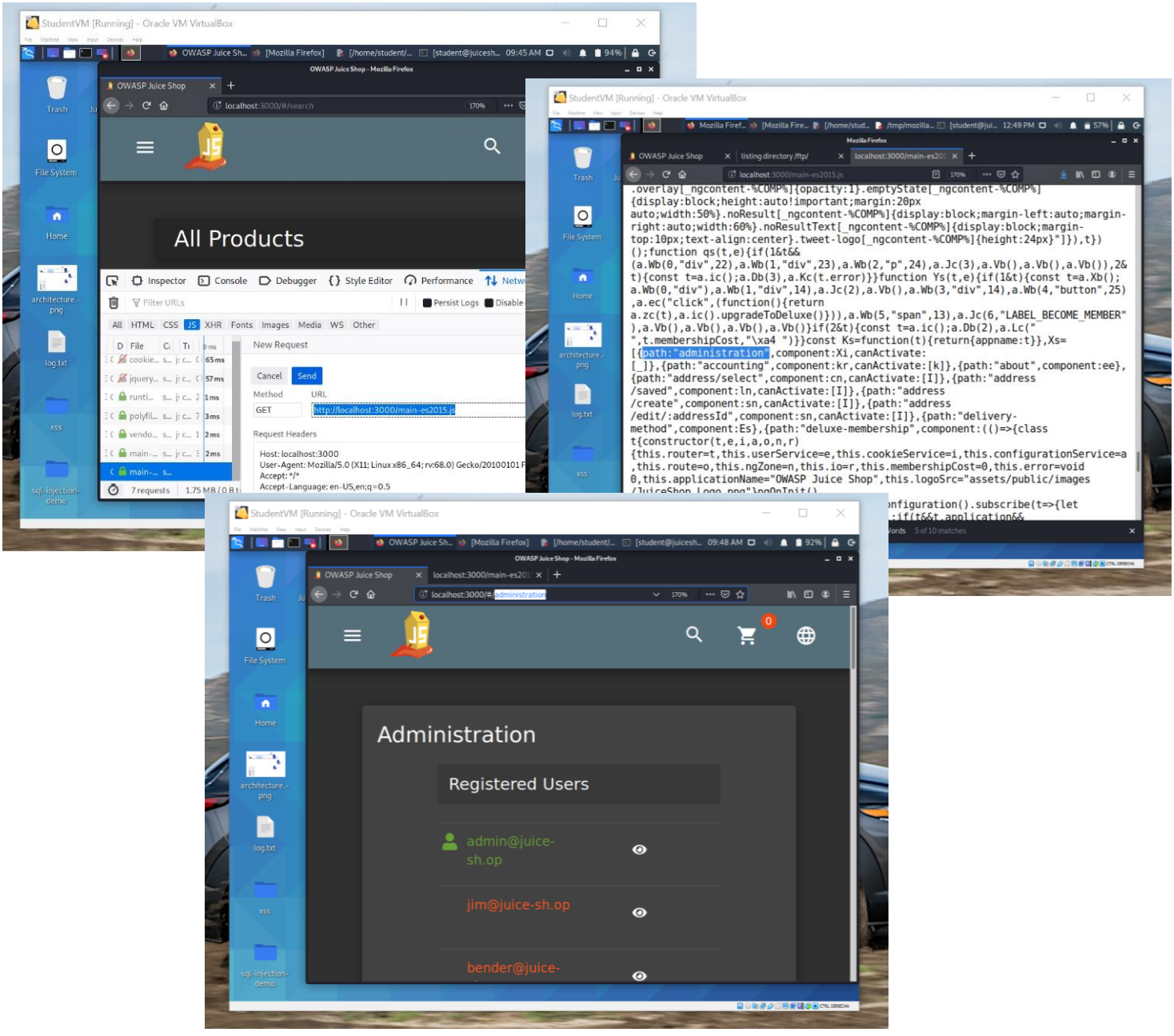
Insert Screenshot of `alert()` popup saying "Hacked!" Here:



Optional Task:

Extra Vulnerabilities

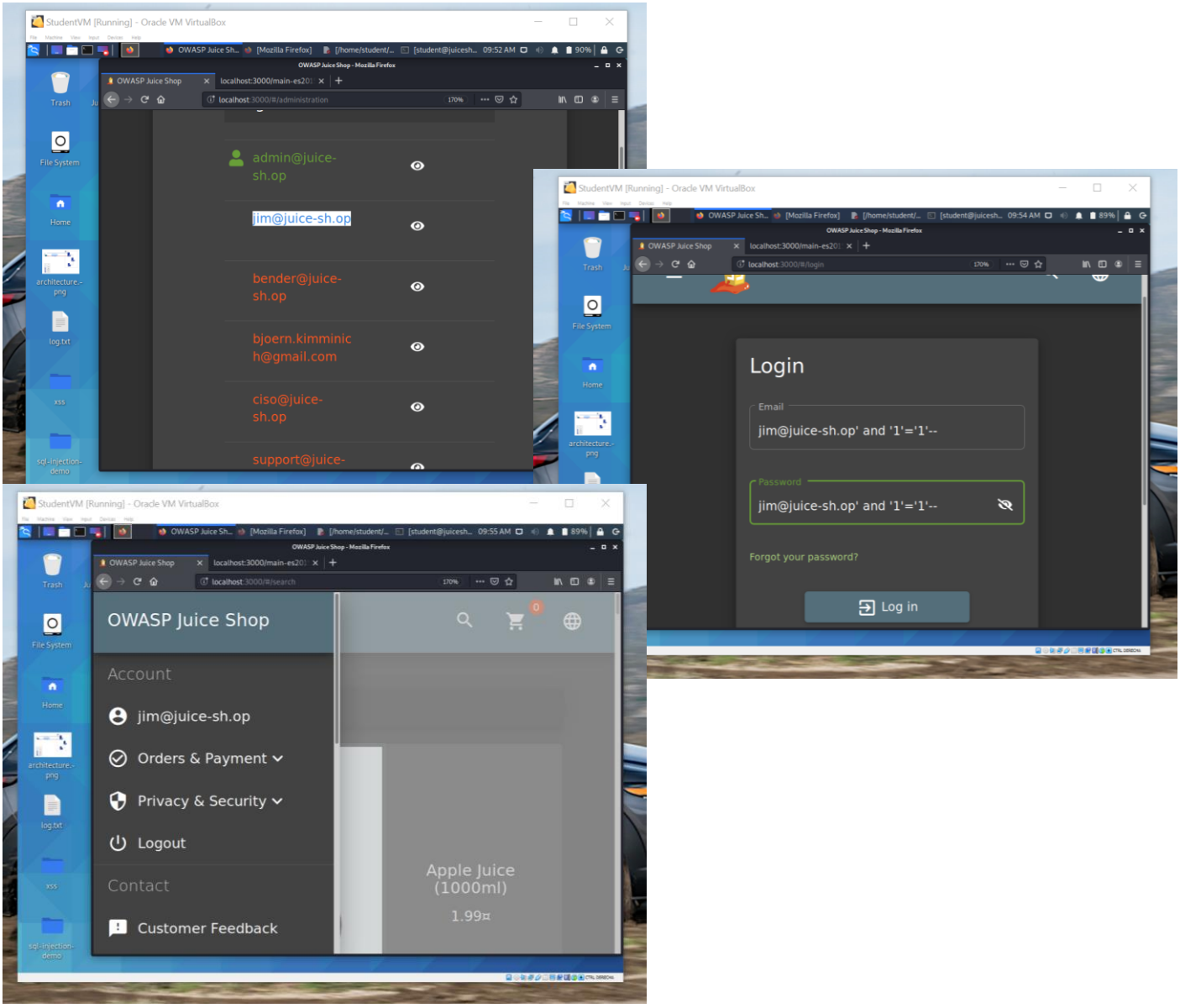
- *Access to administration panel*



Optional Task:

Extra Vulnerabilities

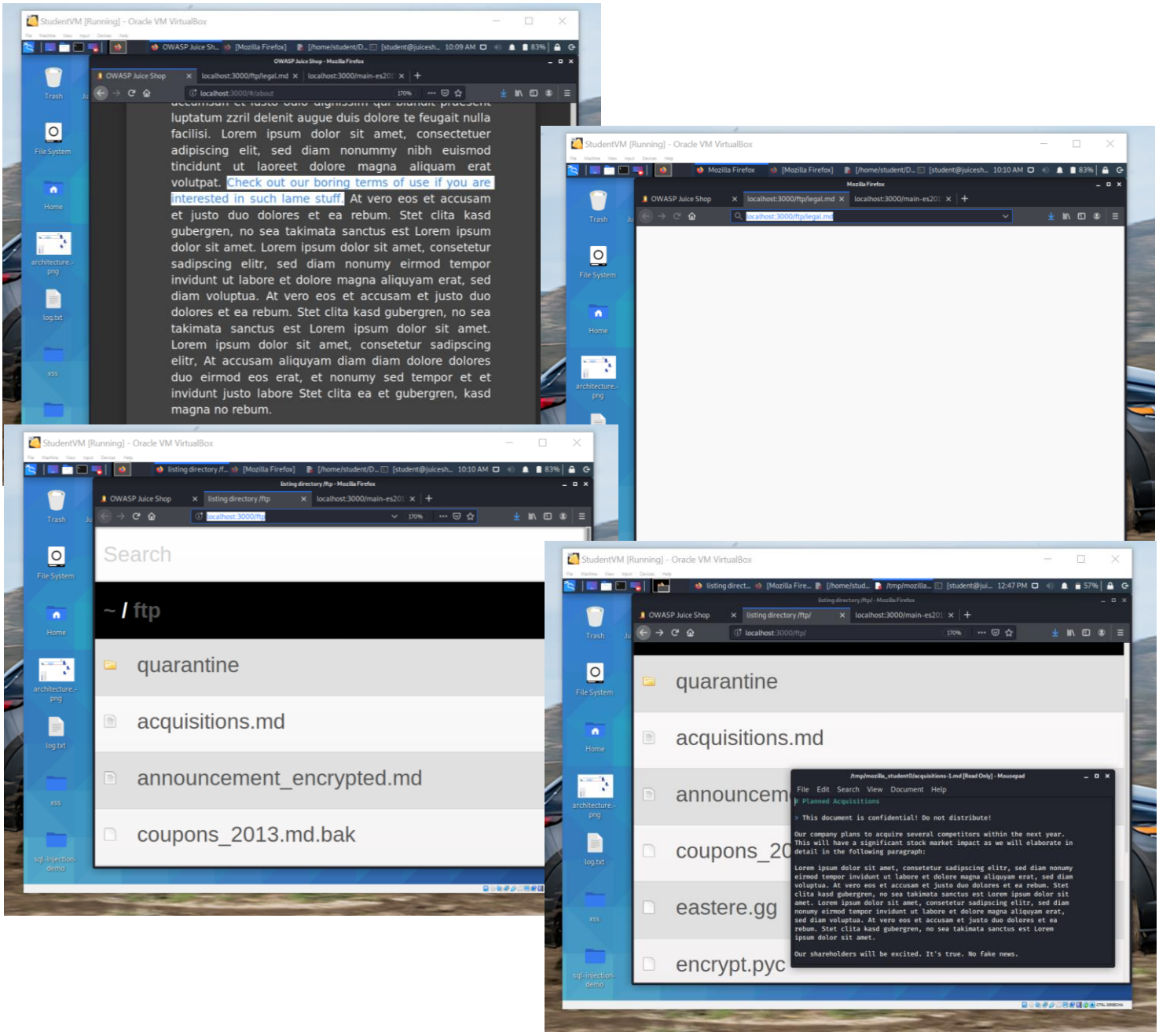
- *Login with client details*



Optional Task:

Extra Vulnerabilities

- *Access to sensitive data on server*



Section 3

Risk Analysis

3.1 Scoring Risks

Risk	**Score** ***(1 is most dangerous, 4 is least dangerous)***
HTTP flood / DDoS	1
Insecure Architecture	3
SQL Injection	2
XSS Vulnerability	4

3.2 Risk Rationale

Why Did You Choose That Ranking?

The HTTP flood / Distributed Denial of Service (DDoS) attack ranks highest, because the current active threat is always the most dangerous threat. Injections are on second place. Injections are leading OWASP's top 10 and are among the most common application vulnerabilities followed by security misconfigurations and Cross Site Scripting (XSS).

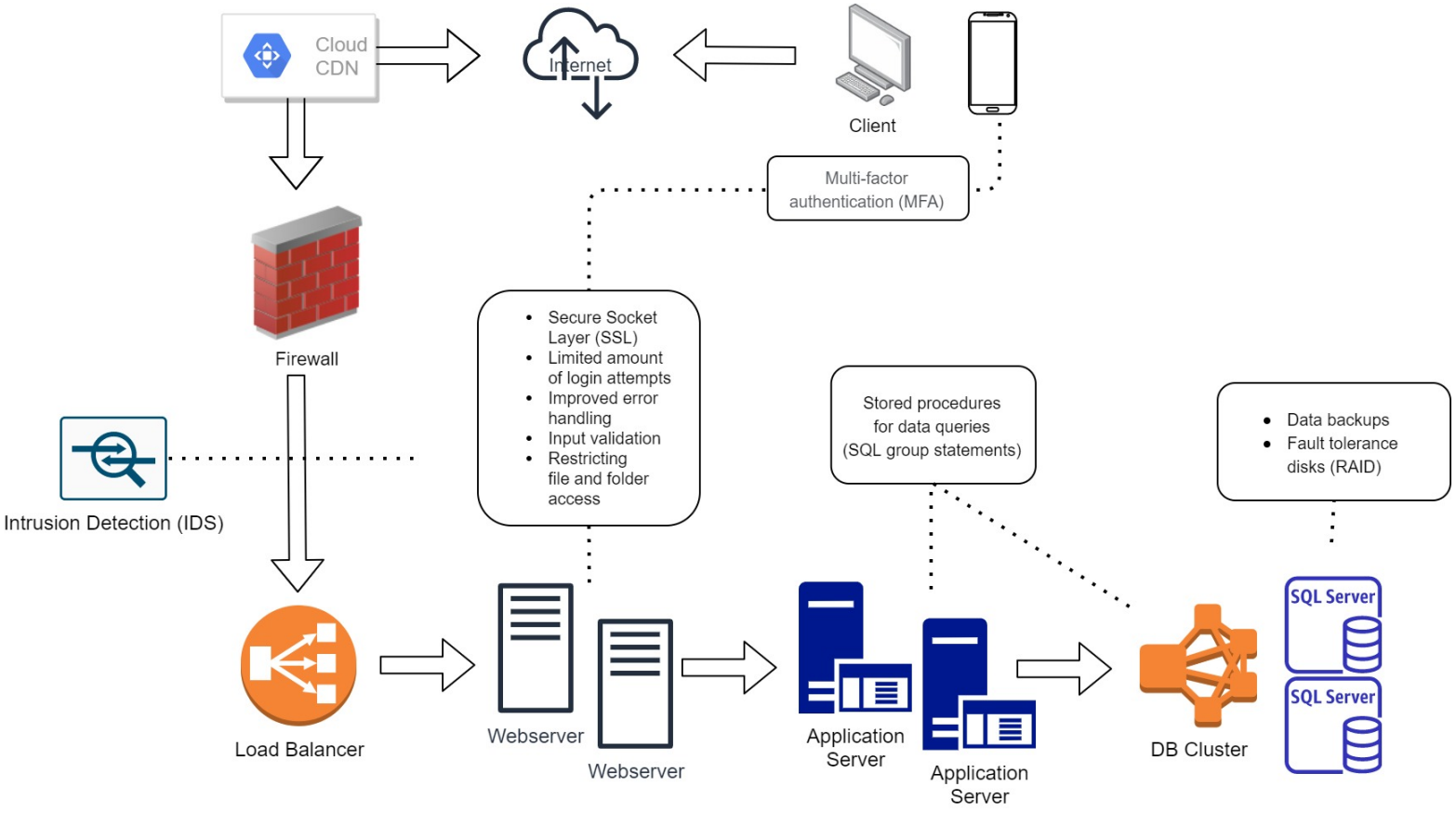
Section 4

Mitigation Plan

4.1 Secure Architecture

What is Your Mitigation Plan?

- The secure architecture includes a Firewall which provides protection against outside cyber attackers by shielding the network from malicious or unnecessary network traffic and an Intrusion Detection System (IDS) to monitor network traffic for suspicious activity and issue alerts when such activity is discovered
- We introduce Multifactor Authentication (MFA) to provide two or more verification factors to gain access to each user account
- We will use load balancing to distribute network traffic across multiple servers.
- Regular database backups will ensure fault tolerance. Fault-tolerant systems use backup components that automatically take the place of failed components, ensuring no loss of service
- Content Delivery Network (CDN) caches content on edge server

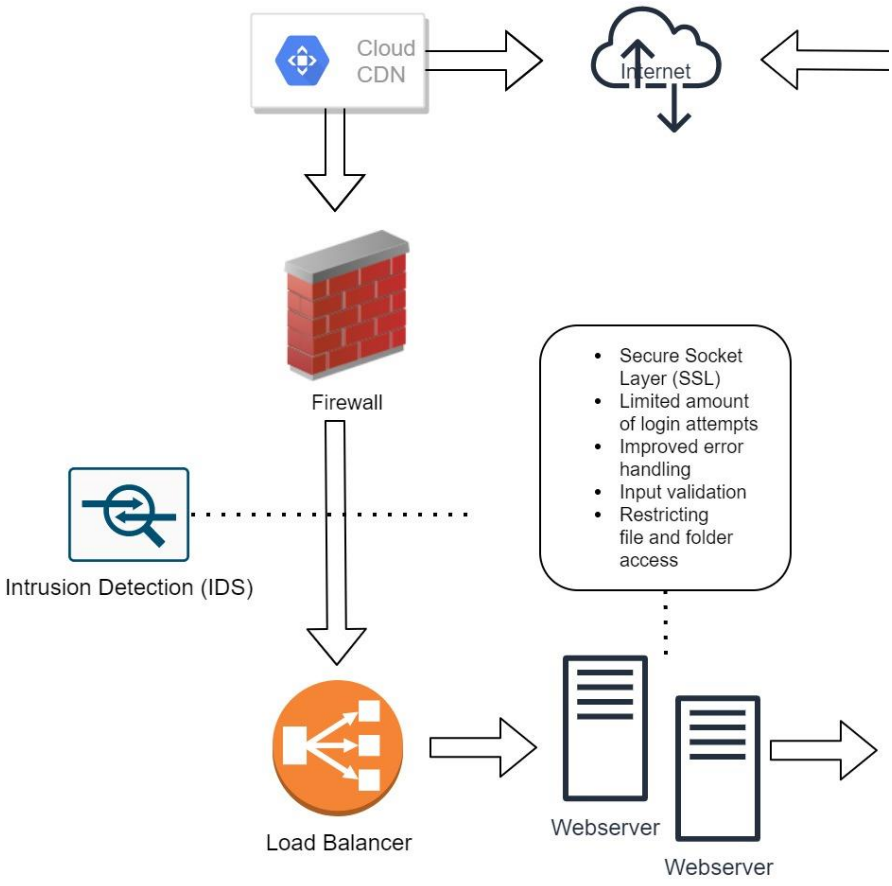


4.2 Mystery Attack Mitigation

What is Your Mitigation Plan?

The following methods can be implement to protect web server from **HTTP floods / DDoS attacks.**

- Load balancer distribute network traffic across multiple servers and avoid overload
- A Firewall with implemented rate limits. DDoS optimized firewalls can identify incomplete connections and flush them from the system when they reach a certain threshold
- Implementing Content Delivery Network (CDN) sets up cached content on edge server handling requests during data attack
- IDS monitoring traffic behaviour

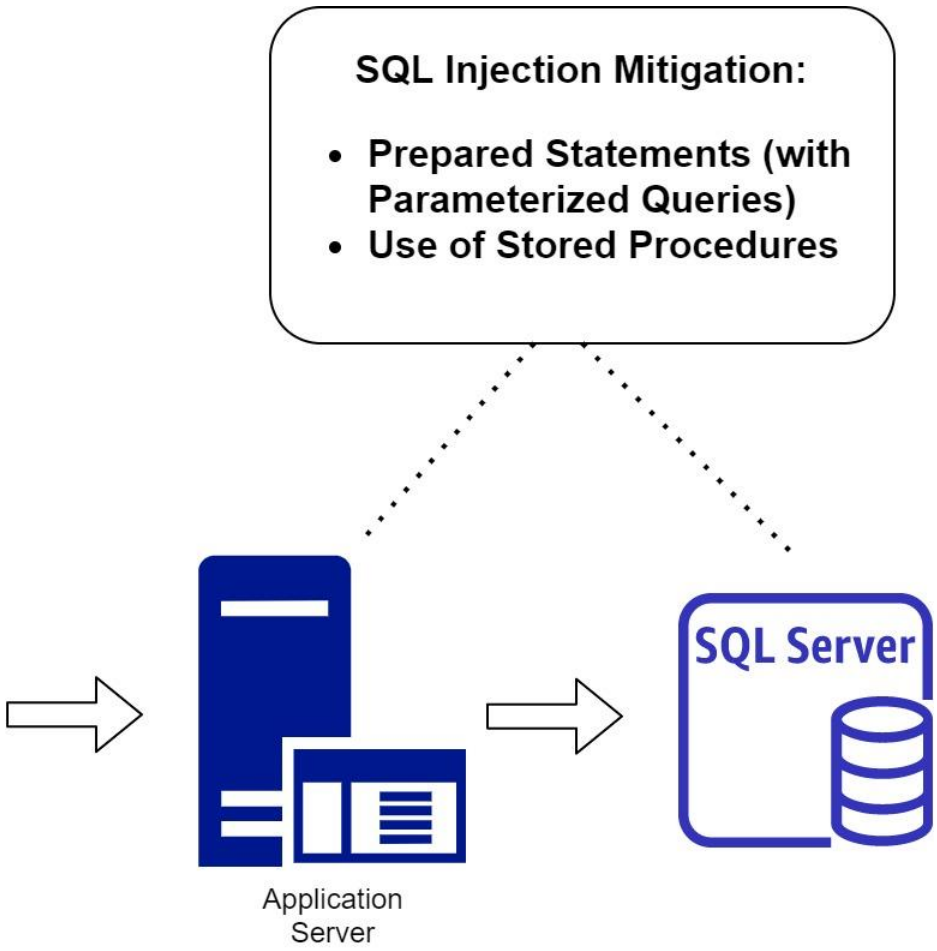


4.3 SQL Injection Mitigation

What is Your Mitigation Plan?

The following methods can be implement to protect web server from SQL Injections:

- **The use of Prepared Statements (with Parameterized Queries)**
- **Or the use of Stored Procedures**



4.4 XSS Mitigation

What is Your Mitigation Plan?

The following methods can be implemented to mitigate XSS attacks:

- **Input Sanitization:** Taking the user input and cleaning it
- **Input validation.** User inputs filtered from the malicious chain of commands. **Encoding.** All the inputs including special characters should be ciphered in respective HTML or URL codes