Udajuicer: Threat Report

Simon Chen

6/6/2021

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]:*** [A Web Server is a program that uses HTTP(Hypertext Transfer Protocol) to serve files that form web pages to users, in response to their requests, which is forwarded by their computer’s HTTP clients.]
- ***2. [Application Server]:*** [Application server are system software upon which web applications or desktop run on.]
- ***3. [Database Server] : [*** *A Database server is an application that provides database services to other computer programs.* ***]***

1.1: Asset Inventory

Explanation of How A Request Goes from Client to Server

[Response Here]

The request starts when a clients wants to order a product, in this case the client wants to purchase Juice from the Juice Shop.

The first server is a Web server. The web server is a program that uses HTTP to server the static files to users.

The second server is Application Server. The application server is a server program in a computer in a distributed network that provides the business logistic for an application program.

The third server is the Database Server.

The Database server is a computer system that provides other computers with services related to accessing and retrieving data from a database.

1.2 Architecture Audit

Flaws

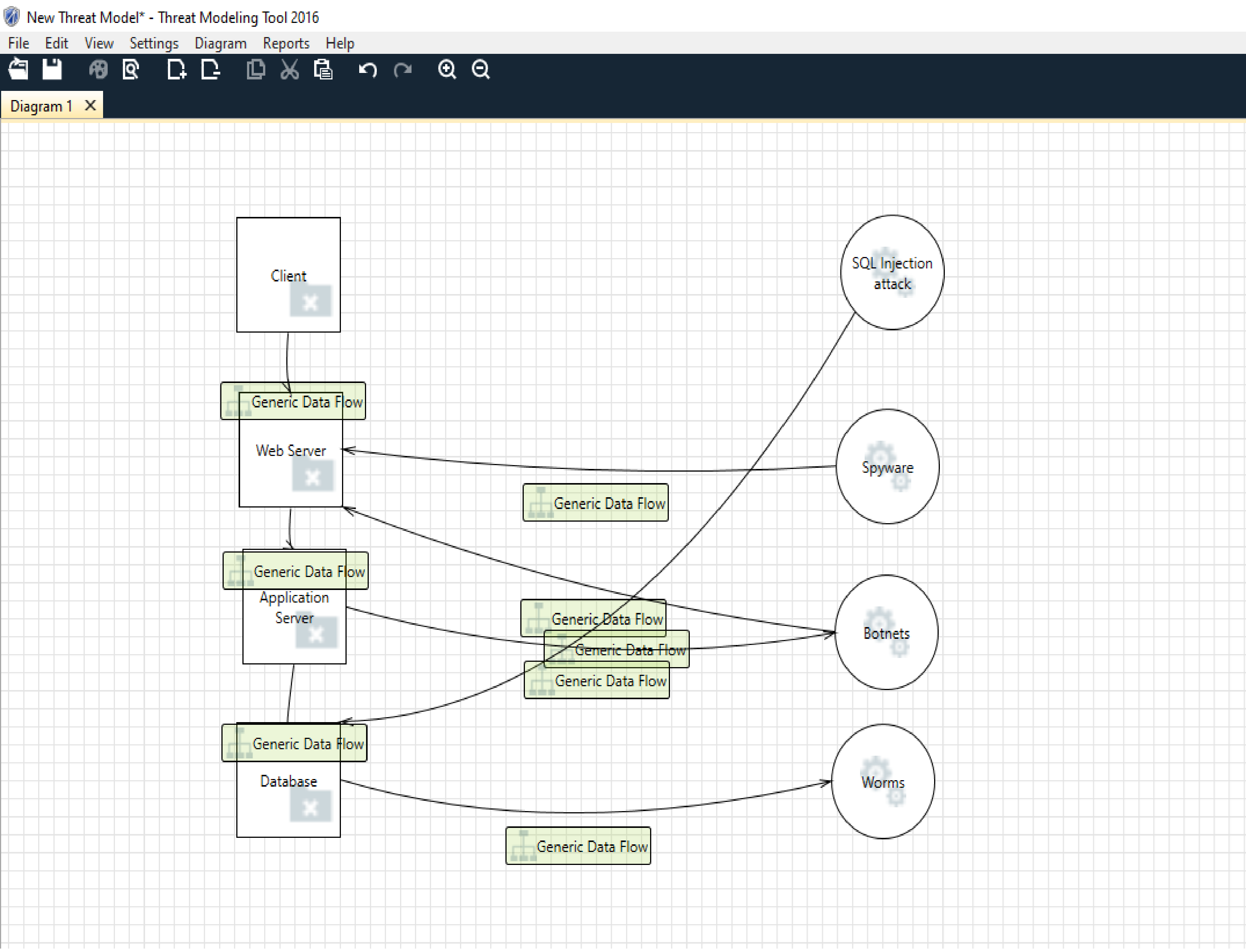
- *[1. Lack of Firewall] : There is currently no firewall in the security architecture. If there is no firewall, then it is susceptible to intrusions. Hackers can try to hack the system.*
- *They need to implement a firewall, so that it can monitor incoming/outgoing traffic.*
- *[2. Lack of Data Backups] : There is currently no data backups on their security architecture. If something were to occur for Example) a power outage. , then they would lose all of their data. They should implement Data Backups, so in case they lose their data that they can recover it.*
- *They should periodically Backup the data. If they do that, the they would make it a secure architecture.*
- *[3.Lack of Load Balancer.] : If they implement a load balancer, then it would make monitoring incoming traffic better.*

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



1.4 Threat Analysis

What Type of Attack Caused the Crash?

[Distributed Denial of Service(DDOS)] – I have analyzed the log file, and it seem that it was flooded with the same attack.

What in the Logs Proves Your Theory?

[I have analyzed the log file, and it seem that it was flooded with the same attack. All of the host is " www.udajuicer.com ", and all of the request states : "request: "Get/ login/ HTTP/1.1 ".

]

1.5 Threat Actor Analysis

Who is the Most Likely Threat Actor?

[Script Kiddie is the most likely threat actor]

What Proves Your Theory?

[Whoever took down the Udajuicer Network used a DDOS attack to take down the network. They implement the attack by trying to overwhelm the network. From the DDOS attack, it looks like they will not gain anything, and it happens that they only did it for fun.

I believe it was Script Kiddie is the most likely threat actor because they will not gain anything of value. They just want to use DDOS attack to learn and hack the network for fun. Script Kiddie is amateur that do hacking or network attacks because they either want to learn to hack or do it for fun.

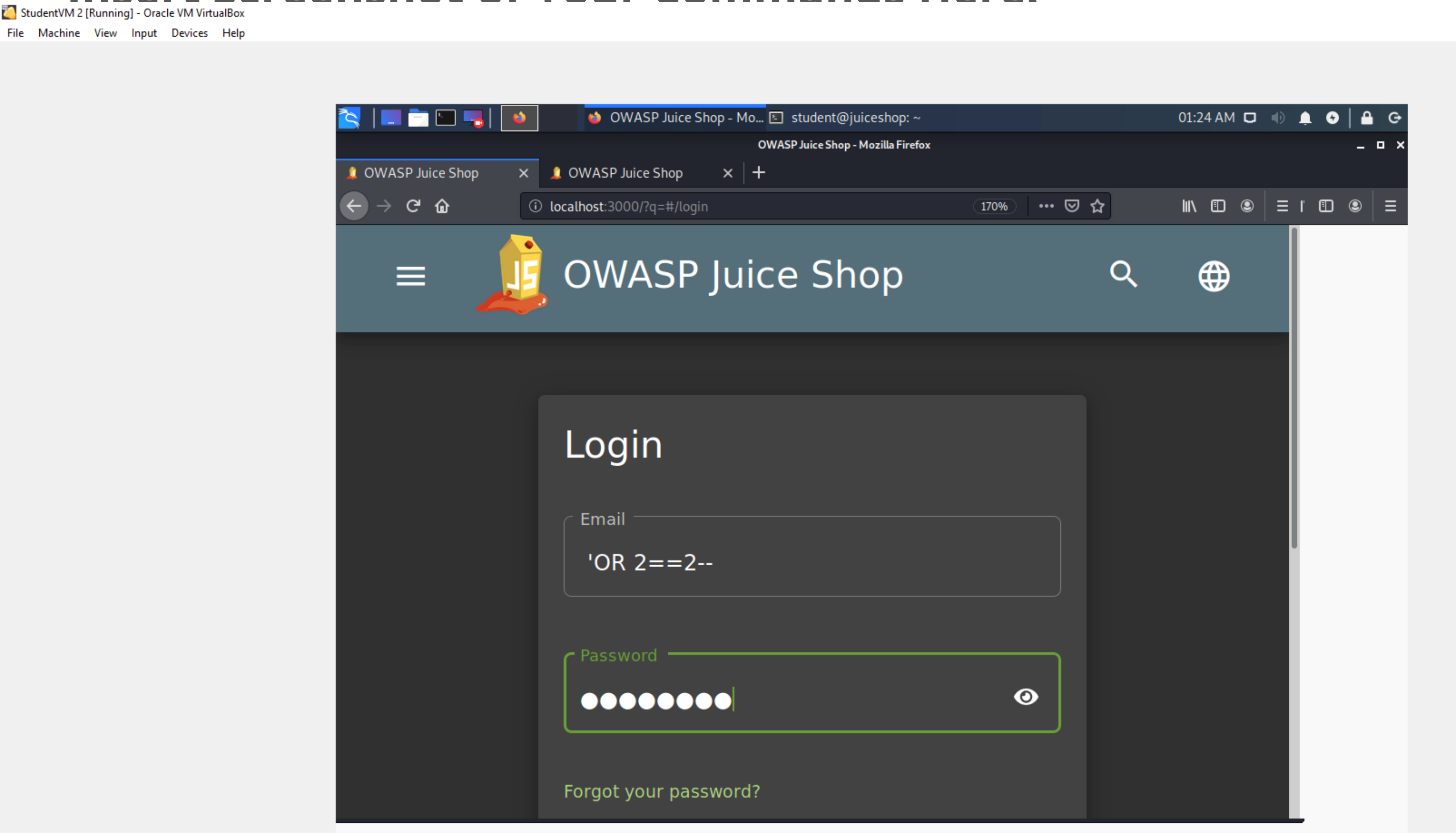
Script Kiddie's motive for their attack is they attack the network for fun or they are learning how to hack a network. Their skill level is beginner as they are only amateur hackers. I believe it is easy to exploit the vulnerability because Chad built an insecure application, and it is vulnerable, so the script kiddie takes an advantage of an exploit. There is no motive for attack. The Script kiddie is hacking for fun, and they are taking advantage of a vulnerability(Udajuicer's insecure application).]

Section 2

Vulnerability Analysis

2.1 SQL Injection

Insert Screenshot of Your Commands Here:

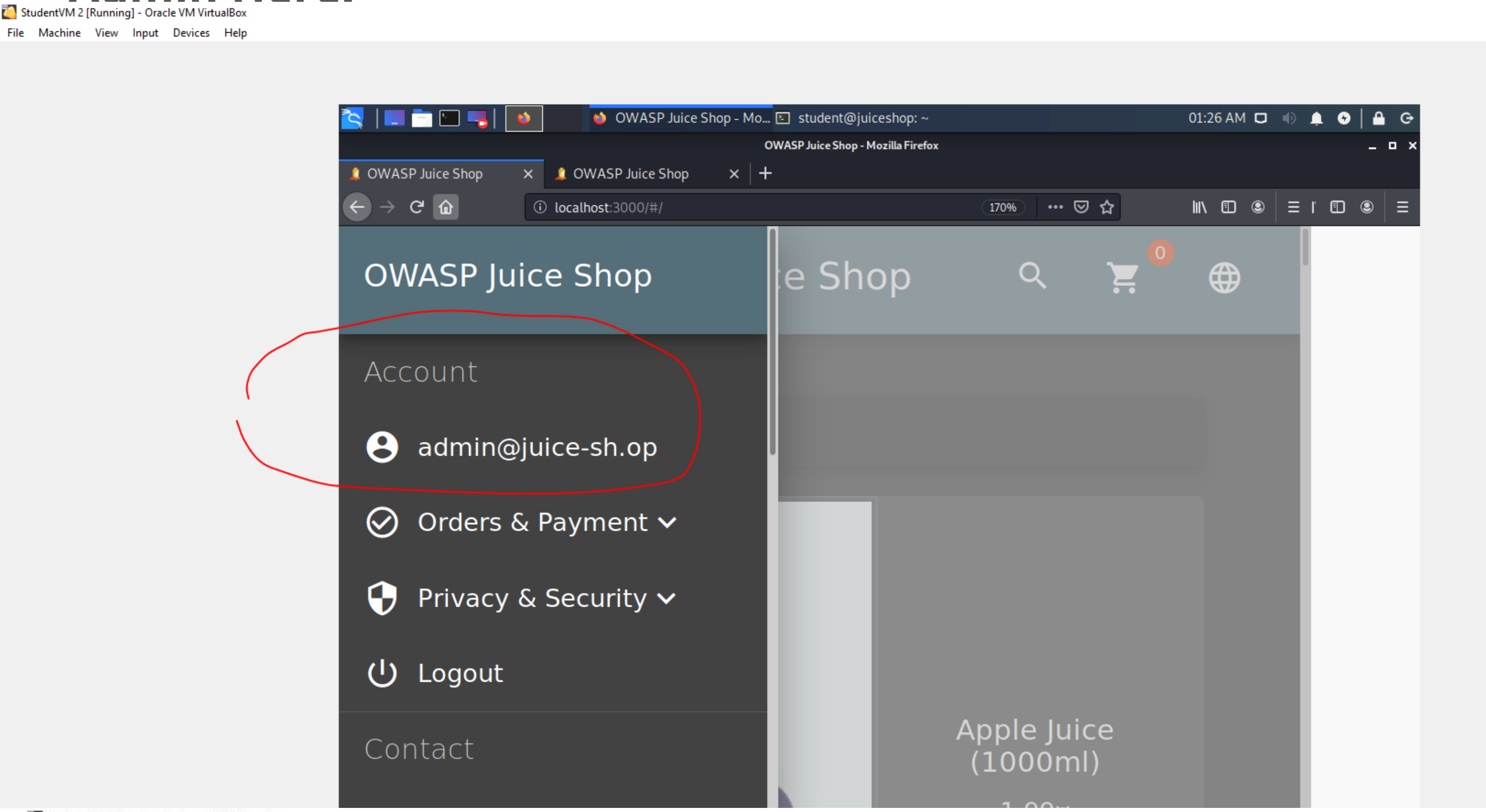

StudentVM 2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
OWASP Juice Shop - Mo... student@juiceshop: ~ 01:24 AM
OWASP Juice Shop - Mozilla Firefox
OWASP Juice Shop
OWASP Juice Shop
localhost:3000/?q=#/login 170%
OWASP Juice Shop
Login
Email
Password
Forgot your password?

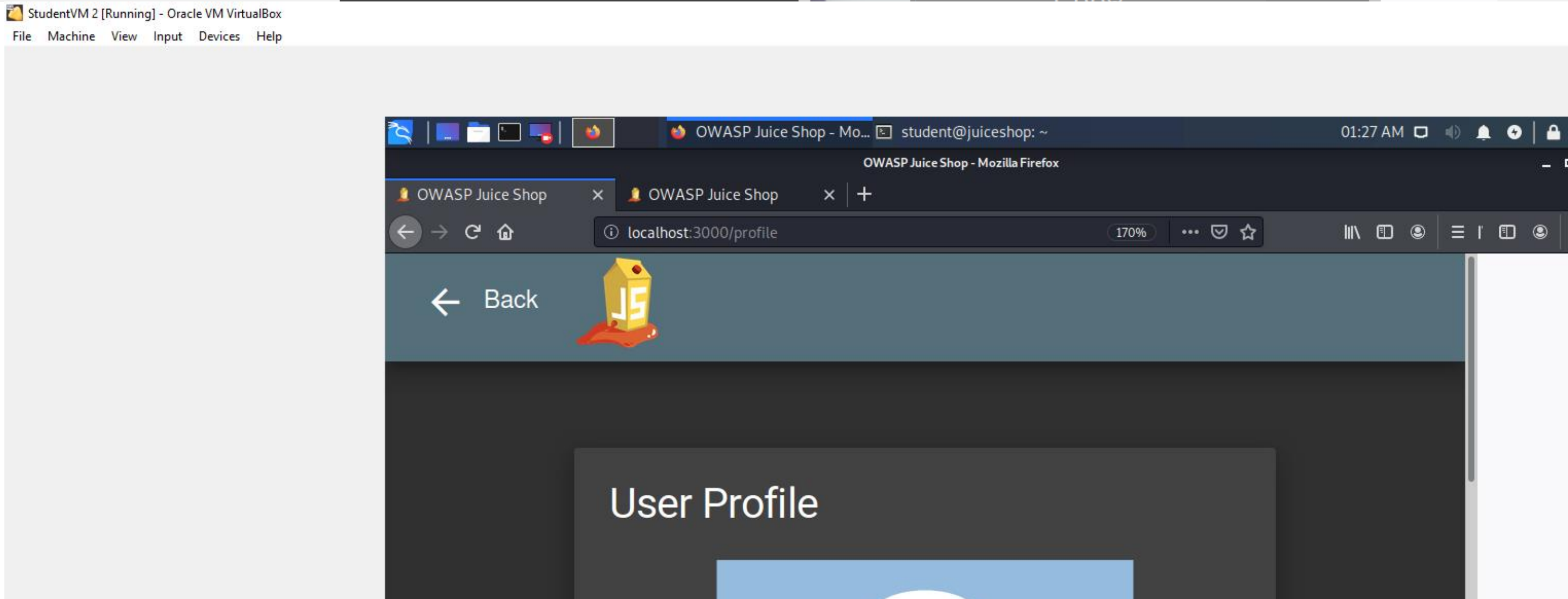
Email: 'OR 2==2--

Password: ●●●●●●●●

2.1 SQL Injection

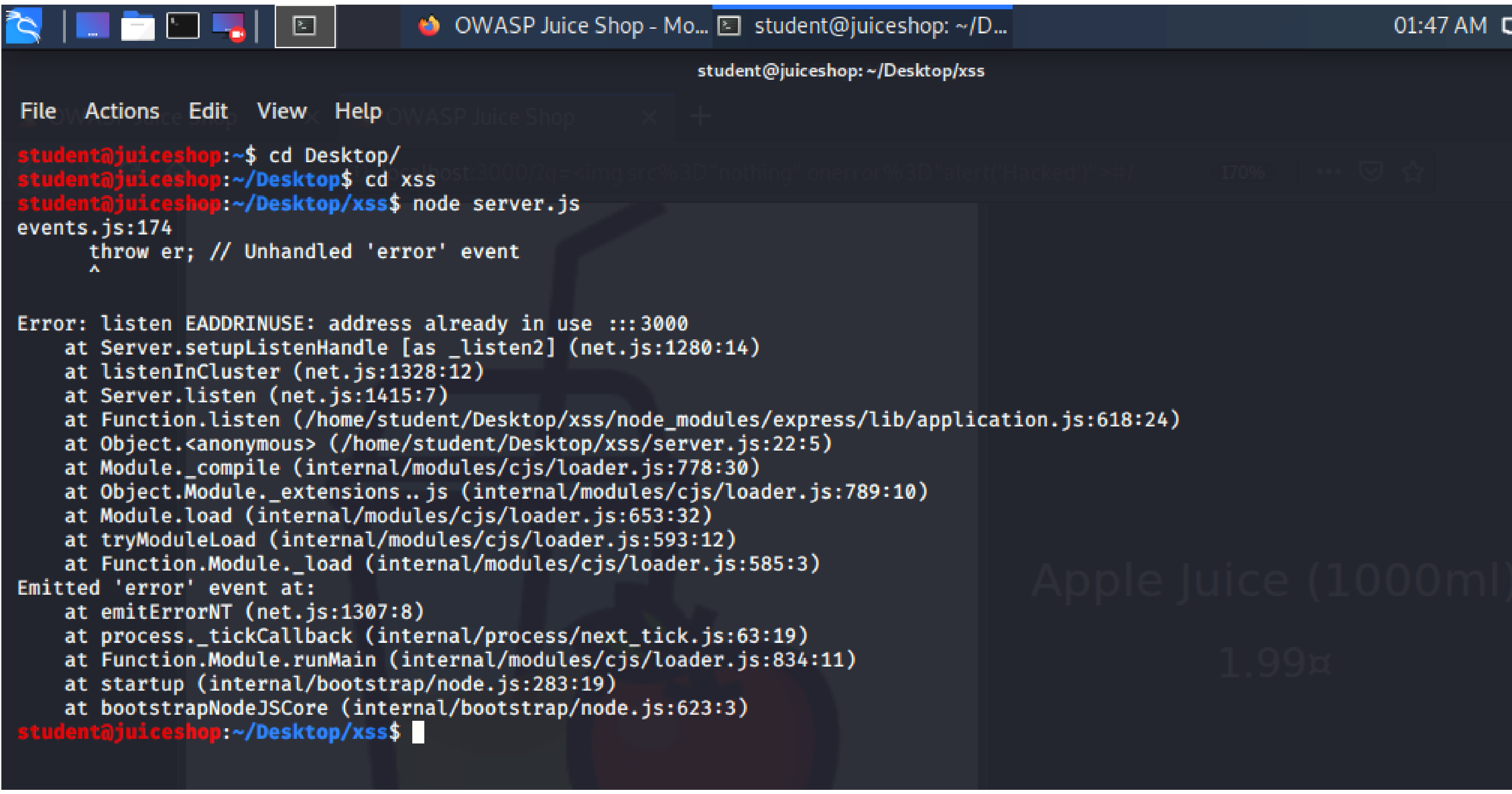
Insert Screenshot of Account Settings Showing You as Admin Here:



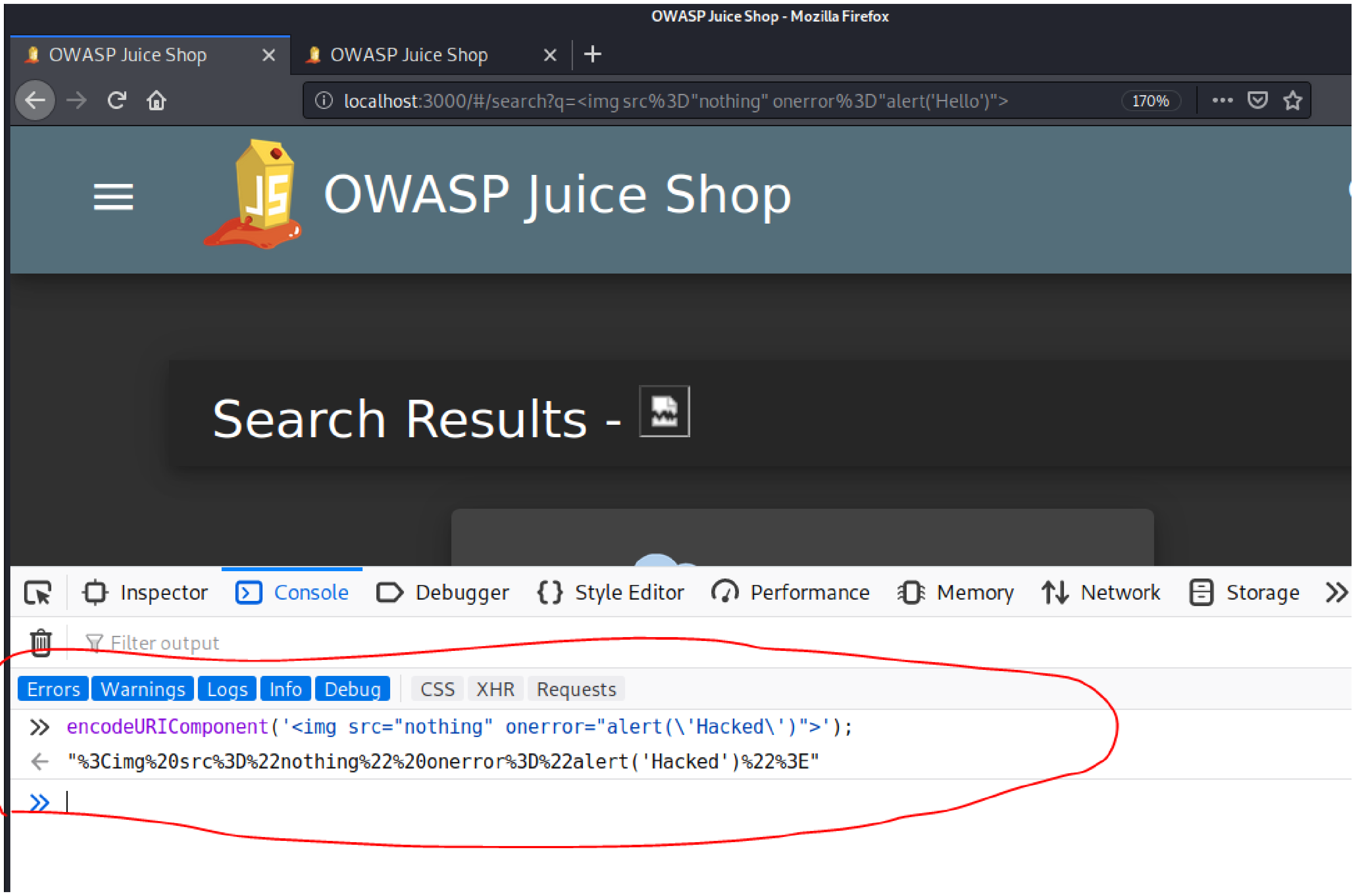


2.2 XSS

Insert Screenshot of Your Commands Here:


OWASP Juice Shop - Mo...
student@juiceshop: ~/D...
01:47 AM
student@juiceshop: ~/Desktop/xss
File Actions Edit View Help

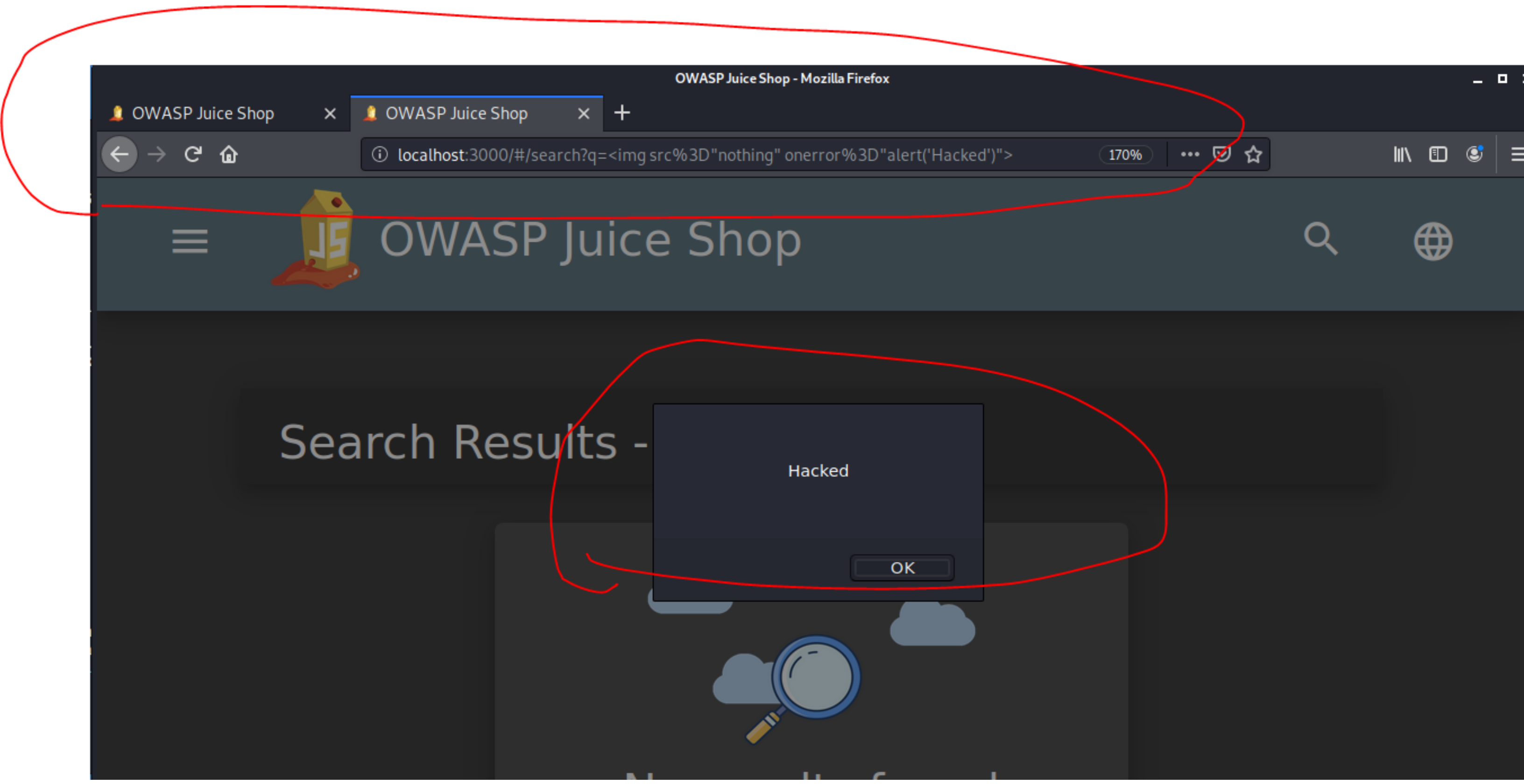
```
student@juiceshop:~$ cd Desktop/
student@juiceshop:~/Desktop$ cd xss
student@juiceshop:~/Desktop/xss$ node server.js
events.js:174
      throw er; // Unhandled 'error' event
      ^

Error: listen EADDRINUSE: address already in use :::3000
    at Server.setupListenHandle [as _listen2] (net.js:1280:14)
    at listenInCluster (net.js:1328:12)
    at Server.listen (net.js:1415:7)
    at Function.listen (/home/student/Desktop/xss/node_modules/express/lib/application.js:618:24)
    at Object.<anonymous> (/home/student/Desktop/xss/server.js:22:5)
    at Module._compile (internal/modules/cjs/loader.js:778:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:789:10)
    at Module.load (internal/modules/cjs/loader.js:653:32)
    at tryModuleLoad (internal/modules/cjs/loader.js:593:12)
    at Function.Module._load (internal/modules/cjs/loader.js:585:3)
Emitted 'error' event at:
    at emitErrorNT (net.js:1307:8)
    at process._tickCallback (internal/process/next_tick.js:63:19)
    at Function.Module.runMain (internal/modules/cjs/loader.js:834:11)
    at startup (internal/bootstrap/node.js:283:19)
    at bootstrapNodeJSCore (internal/bootstrap/node.js:623:3)
student@juiceshop:~/Desktop/xss$ 
```



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

2.2 XSS

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



```
localhost:3000/#/search?q=<img src%3D"nothing" onerror%3D"alert('Hacked')">
```

```
Hacked
```

Section 3

Risk Analysis

3.1 Scoring Risks

Risk	**Score** ***(1 is most dangerous, 4 is least dangerous)***
[Name of Attack Identified in 1.3 Here] *Denial of Service (DOS)*	1
Insecure Architecture	3
SQL Injection	2
XSS Vulnerability	4

3.2 Risk Rationale

Why Did You Choose That Ranking?

[I chose my ranking by first I read a statement “"Excitement was in the air as Udajuicer finally had its flagship product! Udajuicer proceeded to test the application out in beta for a week before production. Unfortunately, the site would constantly go down and they weren’t sure what the issue was“ . I

In this statement, it is Dos Attack so I have to rank Dos attack as the #1 as the most dangerous risk on the list because in the Dos Attack it can force the website to go down quick.

I chose to rank SQL Injection as the second most dangerous risk on the list because SQL Injection made it onto the OSWAP Top 10 and its on top of the list.

Next, I had to decide between XSS Vulnerability or Insecurity Architecture as which one should be third or fourth on this list.

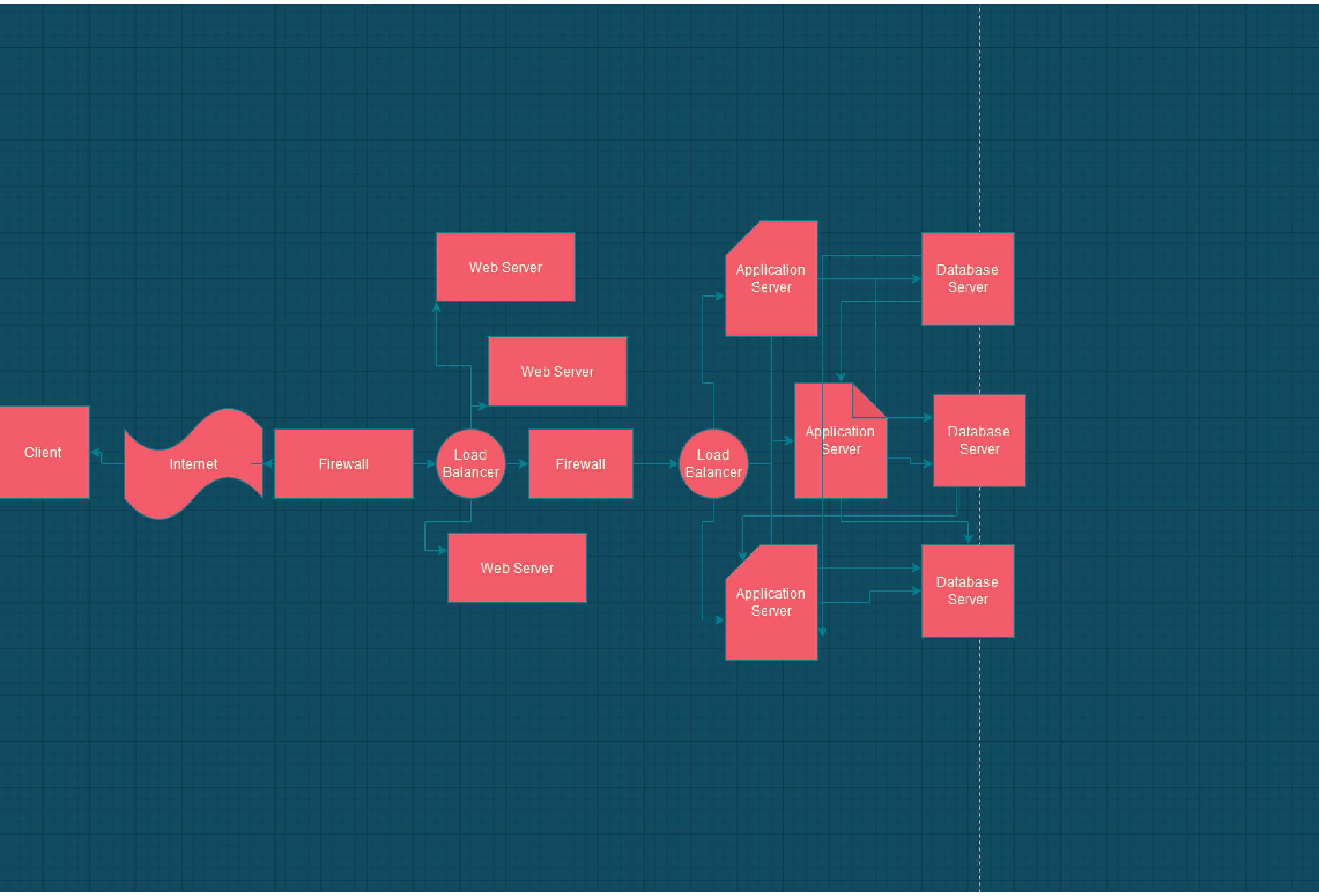
I ended up choosing Insecure Application as the third most dangerous risk on the list because Insecure Architecture is very critical to an organization. I believe that Insecure Application has higher risk and should be more priority over XSS Vulnerability.

I chose XSS Vulnerabilty as the 4th or last dangerous risk on the list because XSS vulnerability has less impact as compared to “ Insecure Architecture” as XSS are completely client side attack, it will not have a direct impact on Udajuicer assets.]

Section 4

Mitigation Plan

4.1 Secure Architecture



4.2 Mystery Attack Mitigation

What is Your Mitigation Plan?

[The Mitigation Plan for preventing Denial of Service(DOS) attack is that I would implement a firewall.]

The Firewall would help prevent further attacks because the firewall is built to monitor incoming/outgoing traffic. You can set and establish set of rules to prevent malicious attacks like : Denial of Service(DOS) Attacks. By building a firewall, I am sure that it help make your organization more secure because it makes it have more security.]

4.3 SQL Injection Mitigation

What is Your Mitigation Plan?

[The mitigation plan for SQL Injection is that I would mitigate SQL Injection by using 1) Input Sanitization   and 2) Input Validation.]

1.The first mitigation strategy is Input Sanitization.

 In the first step, Input Sanitization is defined as the process of taking user input and cleaning it.

This process make sure that there is no unexpected behavior when compiling.

2. The second mitigation strategy is Input Validation.

In the second step, Input Validation is defined as the process of taking user input and matching it against expected input or an allow list.

Input Validation can prevent further SQL Injection attacks because it can make secure. Input Validation can check for length,format when entering a username.

If the input is validated, then the user can access the site while if

The input doesn’t matches up, then the user is denied access to the website.

4.4 XSS Mitigation

What is Your Mitigation Plan?

[My Mitigation Plan for XSS(Cross Site Scripting) is that I would mitigate it by : 1) Escaping   2) Sanitizing User Input   3) Validating User Input

1) Escaping is defined as the process of converting characters in code, so they don't get interpreted.

I would implement Escaping as the mitigation strategy for

Cross Site Scripting(XSS) because it can prevent further attacks because escaping is like encrypting characters, so they don't get interpreted.

2) The second plan to mitigate XSS is by Sanitizing User Input.It would help prevent further attacks because it helps remove malicious characters which will then remove malicious input.

3) The third plan to mitigate XSS is Validating User Input.

For Validating User Input, it is the process of comparing input to what is allowed. It helps prevent further attacks because you will compare input to what it is allowed to be inputed .