Mitigating Software Vulnerabilities

Presented by **Ahmed Mamdouh Bayoumi**

Ahmed Mamdouh Bayoumi

Agenda

01

Reflected XSS Attack

Understanding and executing a Cross-Site Scripting attack.

02

SQL Injection Attack

Demonstrating an SQL Injection attack on a login page.

03

Mitigation Strategies

Proposing effective steps to prevent XSS and SQLi vulnerabilities.

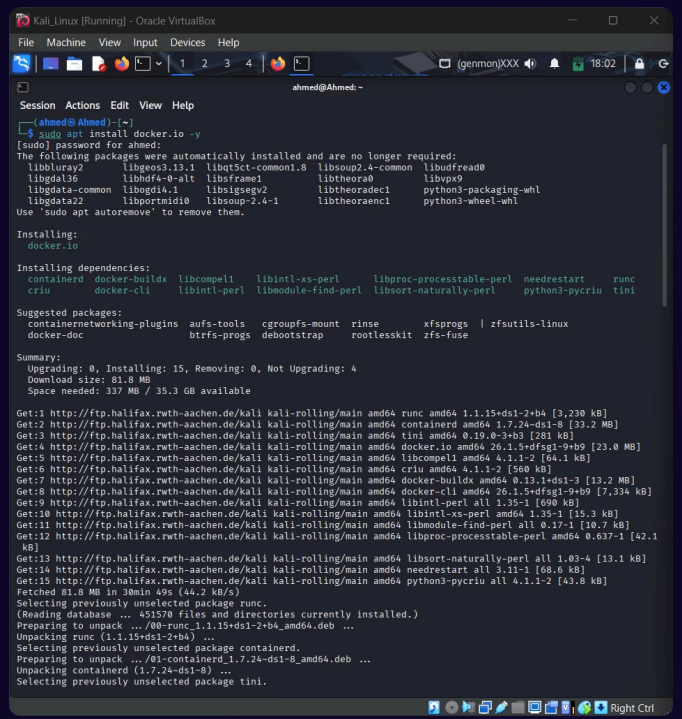
Ahmed Mamdouh Bayoumi

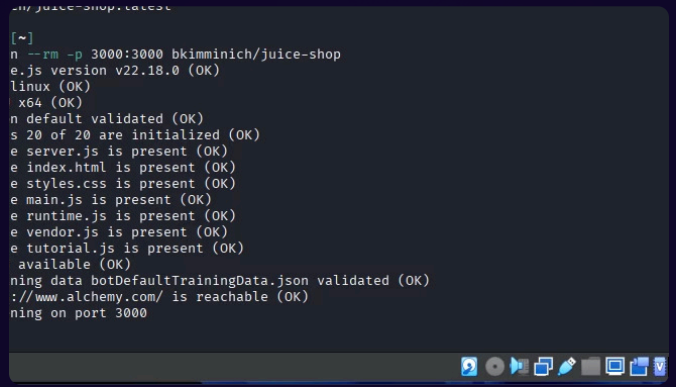
1. Perform a Reflected XSS Attack

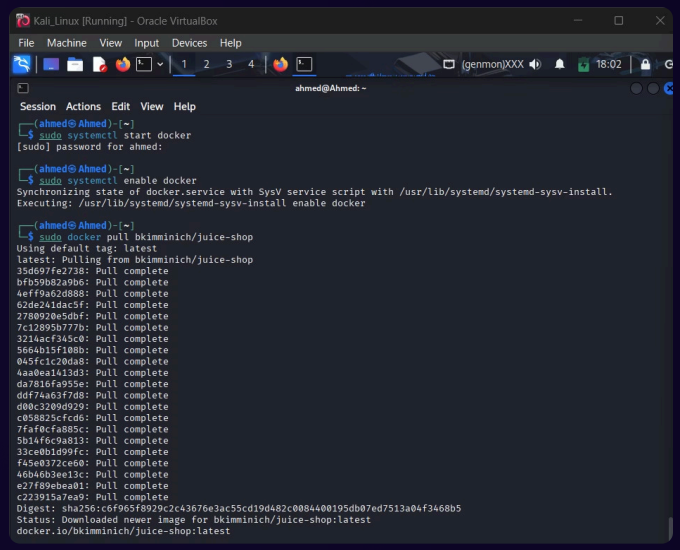
A. Setting up the Vulnerable Application

To demonstrate a Reflected XSS attack, we first need to set up a deliberately insecure web application. We will use OWASP Juice Shop, installed and run via Docker.

Docker installation







Running the application

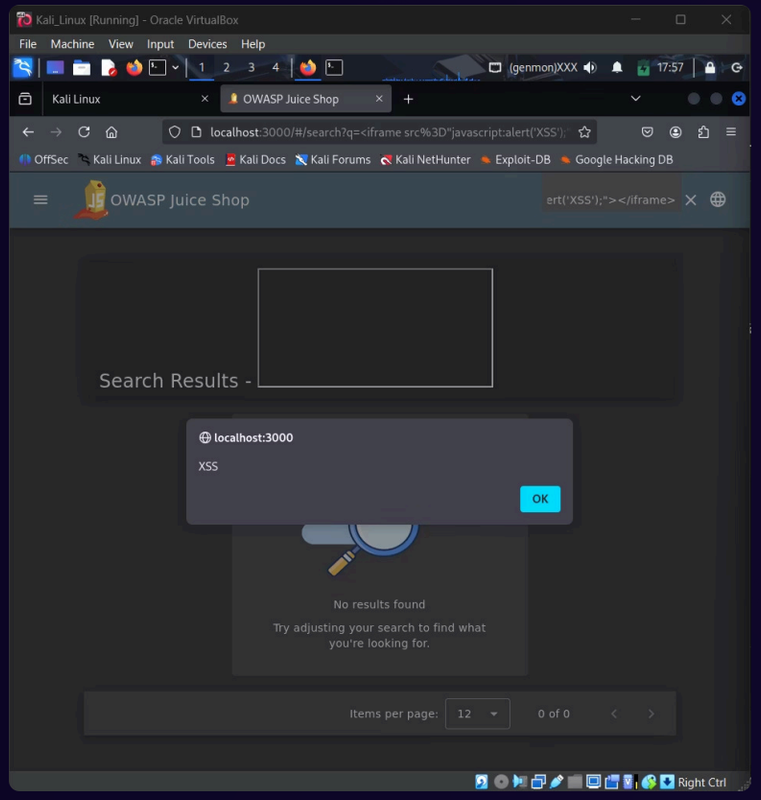
Ahmed Mamdouh Bayoumi

B. Identifying an Input Field and Executing the Attack

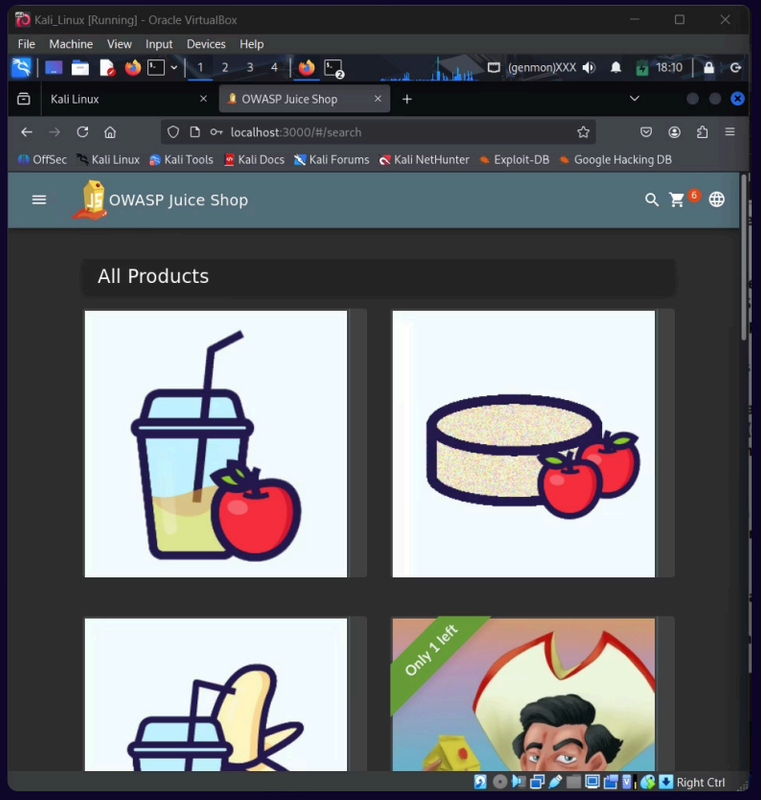
The next step involves identifying an input field that reflects user input back to the page. The search bar is a common and effective target for this type of attack.

C. Entering the Script Payload and Observing the Results

The attack is performed by entering a malicious JavaScript payload into the search field. This payload, typically a simple alert box script, triggers a pop-up, demonstrating the vulnerability.



Entering the malicious payload



Observing the alert box

D. How the Attack Works and Its Potential Impact

A Reflected XSS attack occurs when a malicious script from user input is reflected off a web server and executed in the victim's browser. The attacker sends a specially crafted URL containing the malicious code to the victim. When clicked, the vulnerable web application includes the script in the response, and the browser executes it.

Session Hijacking

Stealing the victim's session cookies to impersonate them.

Malware Distribution

Redirecting the user to a malicious website that downloads malware.

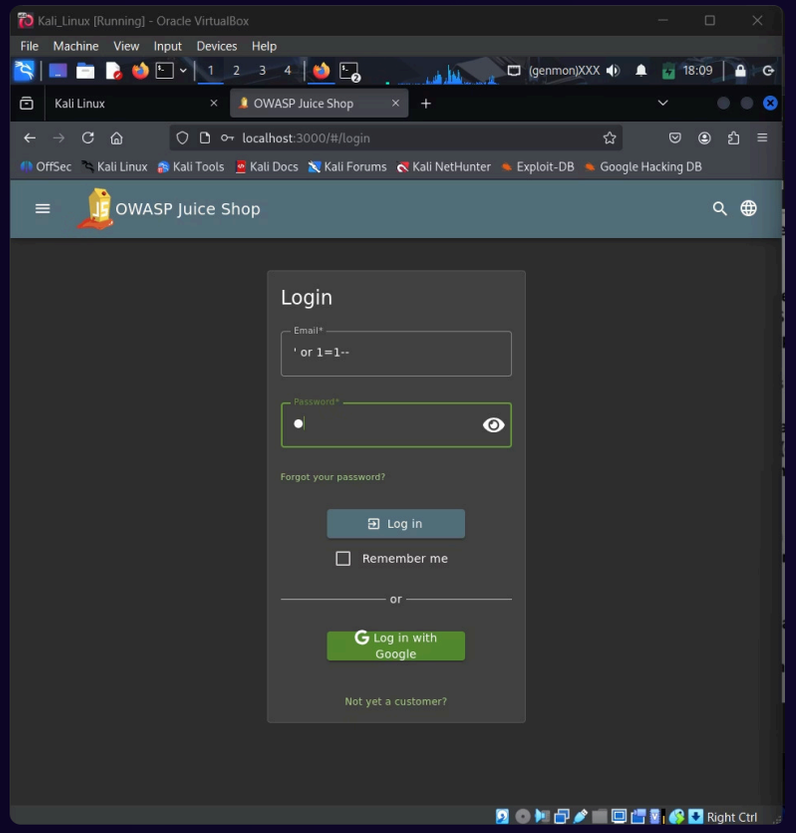
Data Theft

Stealing sensitive information displayed on the page, like personal data or credit card numbers.

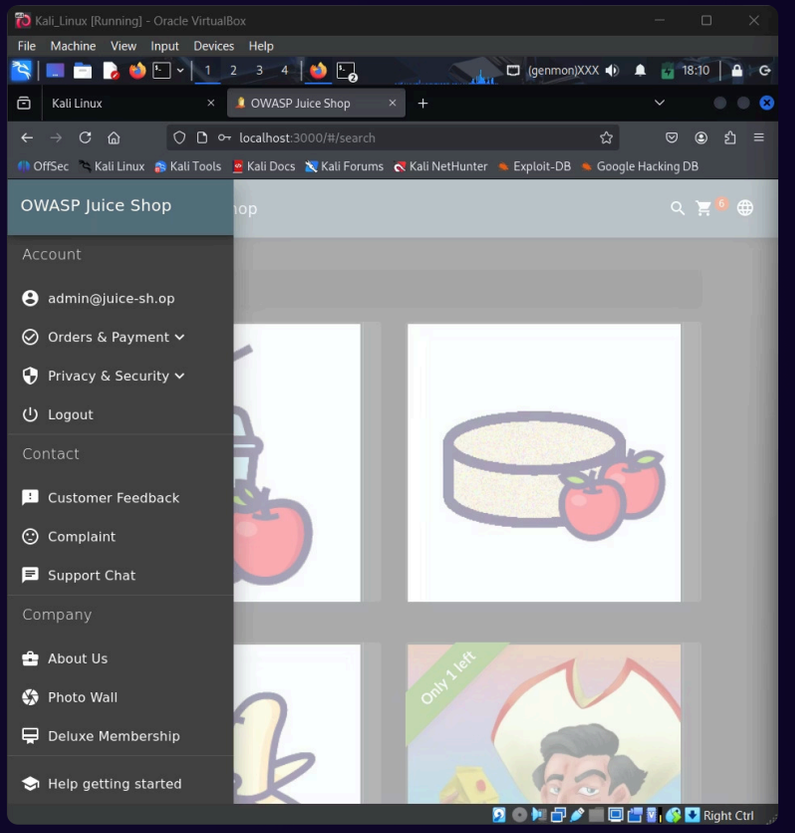
2. Conduct an SQL Injection Attack

A. Using a Testing Environment and Injecting Malicious Commands

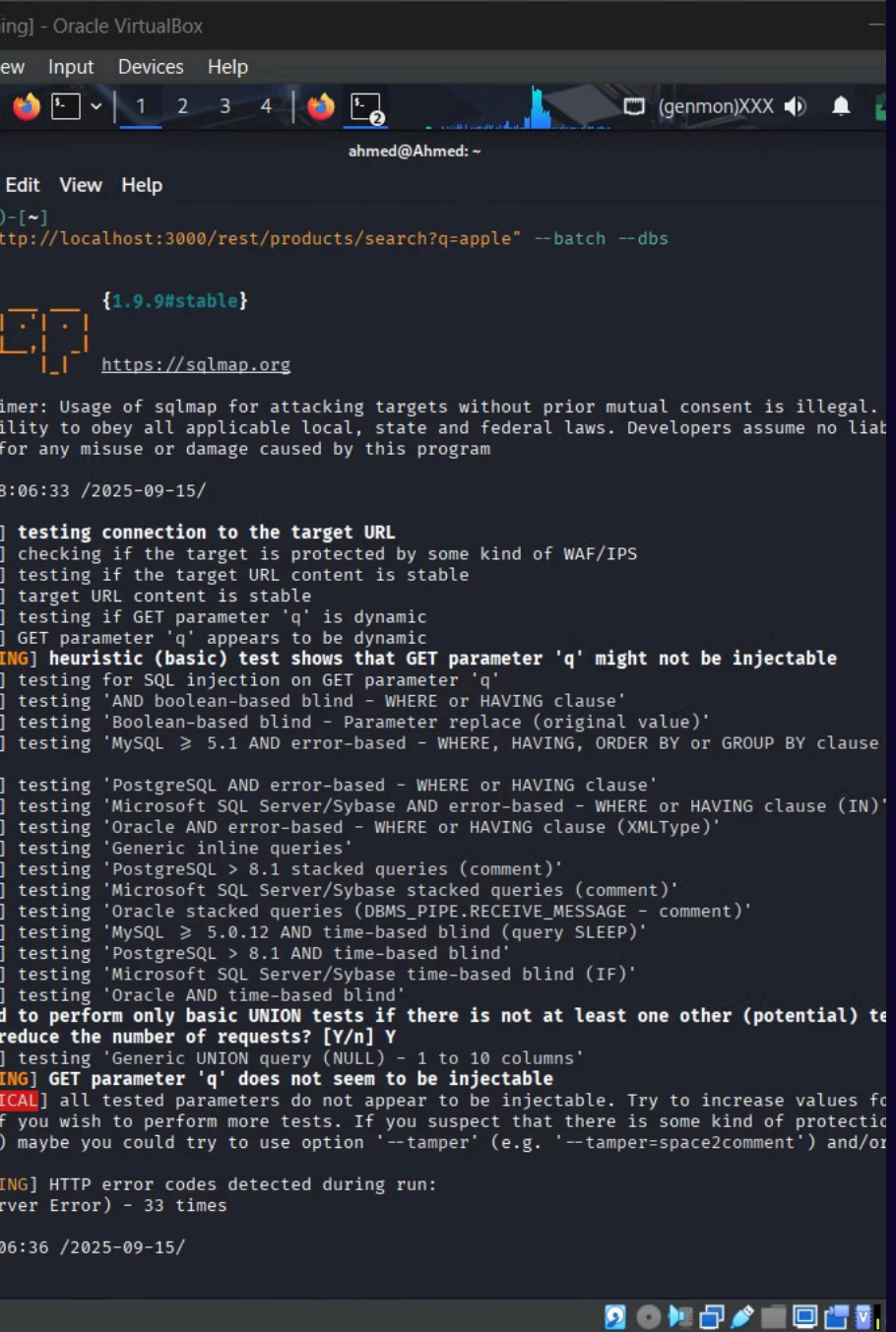
An SQLi attack exploits vulnerabilities in an application's database queries. This demonstration uses a manual attack on the login form of OWASP Juice Shop.



SQLi login bypass attempt



Successful login bypass



B. Recording the Results and Explaining Consequences

The manual SQLi attack successfully bypassed the login authentication, granting unauthorized access to the admin account. The consequences of SQLi vulnerabilities can be catastrophic.

Data Breach

Stealing, modifying, or deleting sensitive data, including customer information, financial records, and intellectual property.

Full System Control

Attackers can gain administrative control over the database and the underlying server.

Application Downtime

Malicious queries can cause the database to crash, leading to a denial of service.

Ahmed Mamdouh Bayoumi

3. Propose Mitigation Steps

A. XSS Mitigation

The most effective way to mitigate XSS attacks is through robust input validation and sanitization.

1

Input Validation

Ensure user input matches the expected format (e.g., only alphabetic characters for a name field).

2

Output Encoding/Sanitization

Sanitize or escape special characters before displaying user-generated content to prevent browser interpretation as executable code.

B. SQLi Mitigation

The best way to prevent SQL Injection is by using parameterized queries or prepared statements, along with the principle of least privilege.

1

Parameterized Queries

Use placeholders instead of embedding user input directly into SQL strings. This separates the command from the data, treating input as a literal value.

2

Principle of Least Privilege

Ensure database user accounts have only the minimum necessary permissions to perform their functions, avoiding administrative privileges for applications.