Lab Manual for Hacking Containers Simulation

# Lab Manual for Hacking Containers Simulation Without Hardware

This lab manual provides a detailed guide for simulating container security testing and exploitation in a controlled environment without requiring physical hardware. The focus is on using container platforms like Docker in virtualized environments to mimic real-world scenarios.

---

## \*\*Lab Objectives\*\*  
1. Understand the fundamentals of containerization and its security implications.  
2. Simulate common vulnerabilities in containerized environments.  
3. Identify misconfigurations and exploit weaknesses in container setups.  
4. Demonstrate best practices for securing container environments.

---

## \*\*Lab Prerequisites\*\*  
1. A computer with the following tools installed:  
   - \*\*Docker\*\*: For running and managing containers.  
   - \*\*Kali Linux\*\*: Includes tools for penetration testing and container exploitation.  
   - \*\*Portainer\*\*: A web-based GUI for managing Docker containers.  
   - \*\*Visual Studio Code (VS Code)\*\*: With the Docker extension for inspecting containers.

2. Preconfigured vulnerable container images:  
   - [OWASP Juice Shop](<https://hub.docker.com/r/bkimminich/juice-shop):> A vulnerable web application running in a container.  
   - [DVWA](<https://hub.docker.com/r/vulnerables/web-dvwa):> Damn Vulnerable Web Application.  
   - Custom vulnerable images (e.g., misconfigured SSH or exposed sensitive files).

3. Knowledge of container concepts:  
   - Images, containers, volumes, and networking.  
   - Common container vulnerabilities (e.g., privilege escalation, weak configurations).

---

## \*\*Lab Exercises\*\*

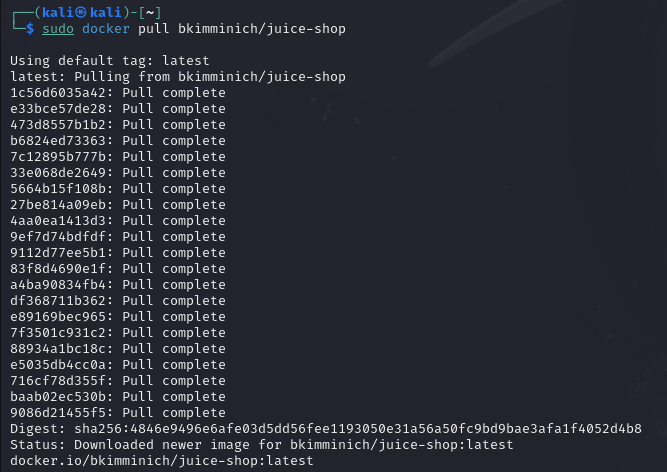
### \*\*Exercise 1: Setting Up a Vulnerable Container Environment\*\*

#### \*\*Objective\*\*:  
Learn how to set up and configure vulnerable containers for testing.

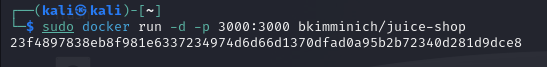
#### \*\*Steps\*\*:  
1. \*\*Install Docker\*\*:  
   - Follow the installation guide for your operating system: [[https://docs.docker.com/get-docker/](https://docs.docker.com/get-docker/).](https://docs.docker.com/get-docker/%5d(https:/docs.docker.com/get-docker/).)



2. \*\*Pull Vulnerable Images\*\*:  
   - Download vulnerable container images:  
     ```bash  
     docker pull bkimminich/juice-shop  
     docker pull vulnerables/web-dvwa  
     ```



3. \*\*Run Containers\*\*:  
   - Start the vulnerable containers:  
     ```bash  
     docker run -d -p 3000:3000 bkimminich/juice-shop  
     docker run -d -p 8080:80 vulnerables/web-dvwa  
     ```



#### \*\*Expected Outcome\*\*:  
Students will have two running vulnerable containers for testing.

---

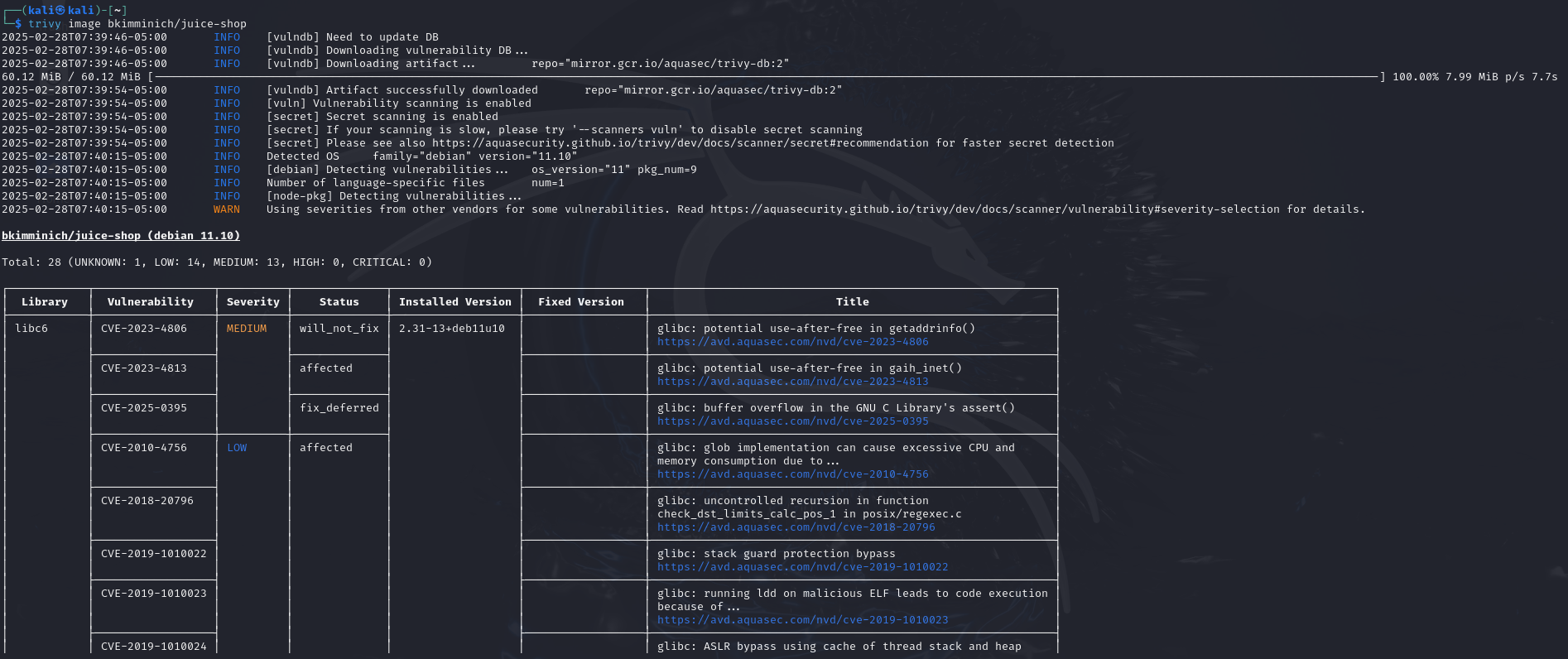
### \*\*Exercise 2: Analyzing Container Images\*\*

#### \*\*Objective\*\*:  
Perform static analysis of container images to identify potential security issues.

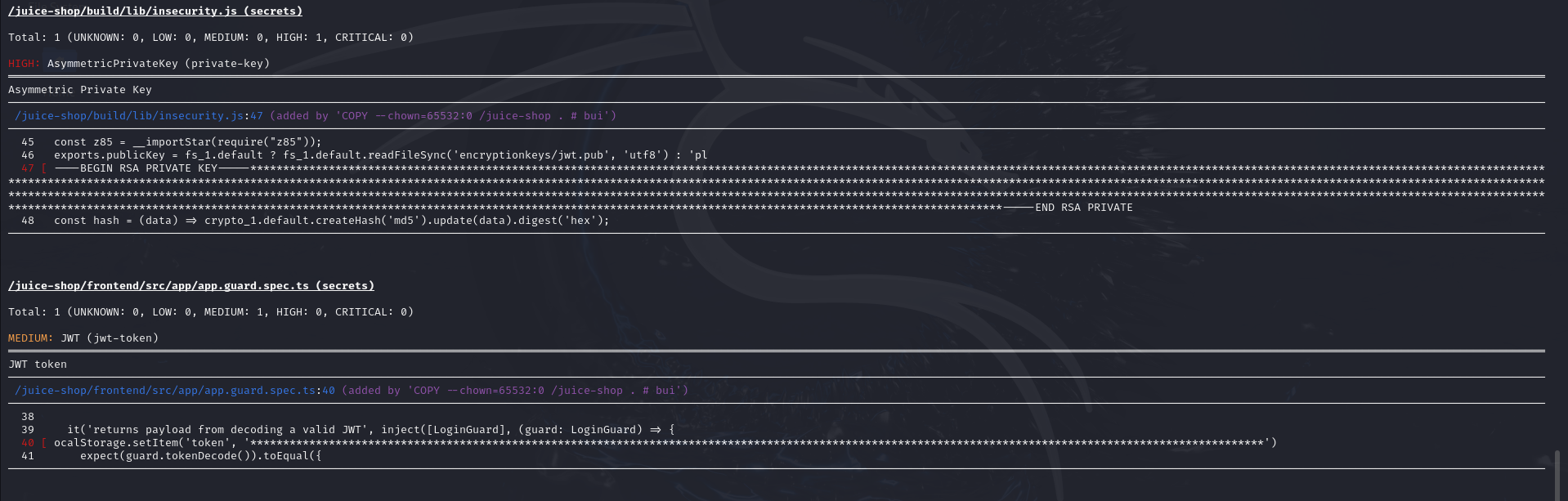
#### \*\*Steps\*\*:  
1. \*\*Inspect Docker Images\*\*:  
   - Use Docker commands to analyze image metadata:  
     ```bash  
     docker inspect bkimminich/juice-shop  
     ```



2. \*\*Scan for Vulnerabilities\*\*:  
   - Use `trivy` to scan images for vulnerabilities:  
     ```bash  
     trivy image bkimminich/juice-shop  
     ```







3. \*\*Analyze Findings\*\*:  
   - Review the output for outdated packages, misconfigurations, and known vulnerabilities.

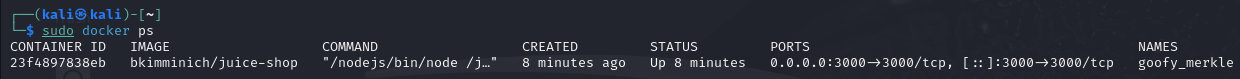
#### \*\*Expected Outcome\*\*:  
Students will understand how to analyze Docker images for security risks.

---

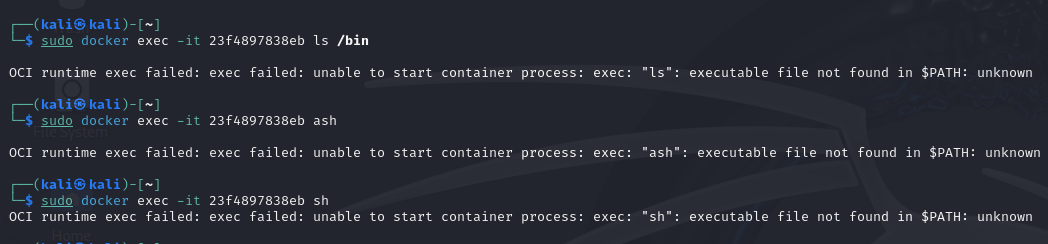
### \*\*Exercise 3: Exploiting Misconfigured Containers\*\*

#### \*\*Objective\*\*:  
Simulate attacks on containers with weak configurations.

#### \*\*Steps\*\*:  
1. \*\*Enumerate Running Containers\*\*:  
   - List active containers:  
     ```bash  
     docker ps  
     ```



2. \*\*Access Container Shell\*\*:  
   - Use Docker to open a shell in a running container:  
     ```bash  
     docker exec -it <container\_id> /bin/bash  
     ```



3. \*\*Explore Privileges\*\*:  
   - Check for root access and sensitive files (e.g., `/etc/passwd`).

4. \*\*Privilege Escalation\*\*:  
   - Simulate privilege escalation by leveraging writable volumes or insecure bindings.

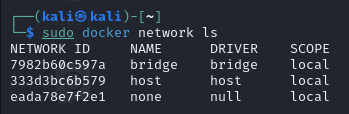
#### \*\*Expected Outcome\*\*:  
Students will learn how insecure container setups can lead to exploitation.

---

### \*\*Exercise 4: Attacking Container Networks\*\*

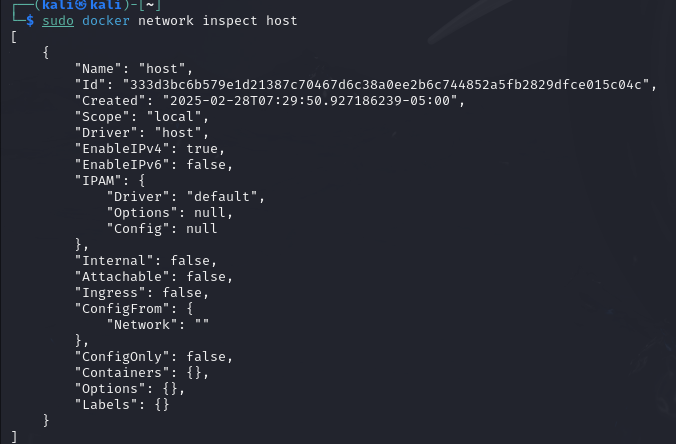
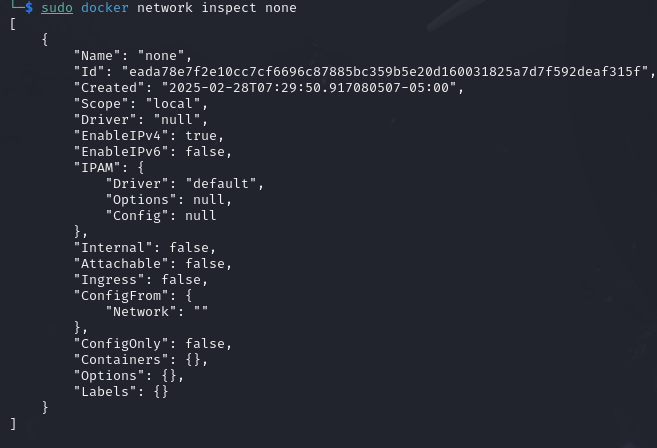
#### \*\*Objective\*\*:  
Simulate attacks on containerized networks to intercept or disrupt traffic.

#### \*\*Steps\*\*:  
1. \*\*Enumerate Networks\*\*:  
   - List Docker networks:  
     ```bash  
     docker network ls  
     ```



2. \*\*Inspect Network Configuration\*\*:  
   - Inspect a specific network:  
     ```bash  
     docker network inspect <network\_name>  
     ```



3. \*\*Perform a MITM Attack\*\*:  
   - Use tools like `ettercap` to intercept traffic between containers.

4. \*\*Exploit Weak Configurations\*\*:  
   - Identify containers with open ports or exposed APIs.

#### \*\*Expected Outcome\*\*:  
Students will understand how container network misconfigurations can expose systems to attacks.

---

### \*\*Exercise 5: Hardening Containers\*\*

#### \*\*Objective\*\*:  
Implement security best practices to secure container environments.

#### \*\*Steps\*\*:  
1. \*\*Restrict Container Privileges\*\*:  
   - Run containers with limited privileges:  
     ```bash  
     docker run --cap-drop=ALL --security-opt=no-new-privileges -d <image>  
     ```

2. \*\*Use Read-Only Filesystems\*\*:  
   - Start containers with read-only root filesystems:  
     ```bash  
     docker run --read-only -d <image>  
     ```

3. \*\*Scan for Security Issues\*\*:  
   - Re-scan containers with `trivy` to verify fixes.

#### \*\*Expected Outcome\*\*:  
Students will learn how to mitigate common vulnerabilities in container environments.

---

## \*\*Lab Summary\*\*  
1. \*\*Skills Acquired\*\*:  
   - Setting up and interacting with containerized environments.  
   - Identifying and exploiting vulnerabilities in containers.  
   - Implementing security measures to harden containers.

2. \*\*Key Takeaways\*\*:  
   - Containers must be configured securely to prevent exploitation.  
   - Regular scanning and monitoring are essential for maintaining container security.

---

## \*\*Additional Resources\*\*  
1. \*\*Container Security Tools\*\*:  
   - Trivy: [[https://github.com/aquasecurity/trivy](https://github.com/aquasecurity/trivy).](https://github.com/aquasecurity/trivy%5d(https:/github.com/aquasecurity/trivy).)  
   - Docker Bench for Security: [[https://github.com/docker/docker-bench-security](https://github.com/docker/docker-bench-security).](https://github.com/docker/docker-bench-security%5d(https:/github.com/docker/docker-bench-security).)

2. \*\*Container Security Guides\*\*:  
   - Docker Security Best Practices: [[https://docs.docker.com/security/](https://docs.docker.com/security/).](https://docs.docker.com/security/%5d(https:/docs.docker.com/security/).)  
   - OWASP Container Security Project: [[https://owasp.org/www-project-container-security/](https://owasp.org/www-project-container-security/).](https://owasp.org/www-project-container-security/%5d(https:/owasp.org/www-project-container-security/).)

By following this manual, students can simulate real-world container penetration testing scenarios and understand the importance of securing containerized environments. Let me know if you need further clarification or additional exercises!