

VULNERD Lab Environment Documentation

1. Project Overview

The VULNERD project is a controlled cybersecurity lab designed to support vulnerability assessment, exploitation, detection engineering, and threat analysis. The environment hosts intentionally vulnerable services while maintaining strict access controls to prevent public exposure.

The lab is deployed on a Raspberry Pi 5 and accessed remotely through secure channels, allowing team members to perform testing from off-site locations while maintaining a responsible security posture.

2. Hardware Platform

- **Device:** Raspberry Pi 5
- **Operating System:** Raspberry Pi OS Lite (64-bit)
- **Storage:** USB-based solid-state storage
- **Network:** Wired Ethernet
- **Access Mode:** Fully headless (no monitor, keyboard, or mouse)
- **Purpose:** Vulnerable web application testing lab

Justification

The Raspberry Pi 5 was selected to demonstrate that effective cybersecurity lab environments can be deployed using compact, low-power hardware while still supporting modern security tooling. USB-based solid-state storage was chosen to provide consistent performance, broad compatibility, and ease of deployment.

This design supports portability, rapid reinstallation, and reproducibility, which are important considerations for academic lab environments and collaborative testing scenarios.

3. Operating System Selection

- **OS:** Raspberry Pi OS Lite (64-bit)

Justification

Raspberry Pi OS Lite was selected due to its lightweight design, official hardware support, and suitability for headless server deployments. By omitting a graphical desktop environment,

unnecessary services were eliminated, reducing system overhead and minimizing the attack surface.

4. System Configuration

- **Hostname:** vulnerd
- **Remote Management:** Secure Shell (SSH)
- **Networking:** DHCP-assigned private IP address

The system was configured using Raspberry Pi Imager with SSH enabled prior to first boot, allowing immediate remote access without requiring physical interaction.

5. Initial System Hardening

Before deploying any vulnerable services, the operating system was fully updated to establish a secure baseline.

```
sudo apt update && sudo apt upgrade -y
```

Purpose

Updating the system ensures that baseline packages include the latest security patches, reducing unintended vulnerabilities outside of the intentionally vulnerable applications.

6. Containerization Strategy

- **Platform:** Docker
- **Orchestration:** Docker Compose

Justification

Docker was used to containerize vulnerable applications in order to:

- Isolate services from the host operating system
- Ensure repeatable and consistent deployments
- Simplify teardown and reinstallation
- Prevent persistent system-level compromise

This approach aligns with modern DevSecOps and security lab deployment practices.

6A. Docker Installation and Configuration

Docker was installed using the official Docker installation script to ensure compatibility with the Raspberry Pi hardware platform.

```
curl -fsSL https://get.docker.com | sh
```

To allow Docker usage without root privileges, the default user was added to the Docker group.

```
sudo usermod -aG docker $USER  
newgrp docker
```

Docker installation was verified using:

```
docker version
```

Docker Compose was installed to support declarative multi-container deployments.

```
sudo apt install docker-compose-plugin -y
```

Verification was performed using:

```
docker compose version
```

7. Vulnerable Application Deployment (DVWA)

- **Application:** Damn Vulnerable Web Application (DVWA)
- **Deployment Method:** Docker containers
- **Container Architecture:** ARM64-compatible
- **Exposed Port:** 8081 (non-standard HTTP port)

Design Consideration

Because the Raspberry Pi 5 uses an ARM64 architecture, DVWA was deployed using an ARM-compatible container image to ensure native execution and avoid emulation overhead.

8. Database Backend Deployment

DVWA requires a backend database service. A dedicated MariaDB container was deployed to support this requirement.

- **Database Engine:** MariaDB
- **Database Name:** dvwa
- **Database Username:** vulnerd
- **Database Password:** vulnerd

The database service is isolated within Docker's internal network and is not directly exposed to the host or external networks.

9. Docker Compose Configuration

A Docker Compose configuration file was used to define both the DVWA web application and the MariaDB backend.

```
services:
  db:
    image: mariadb:10.6
    container_name: dvwa-db
    environment:
      MYSQL_DATABASE: dvwa
      MYSQL_USER: vulnerd
      MYSQL_PASSWORD: vulnerd
      MYSQL_ROOT_PASSWORD: vulnerd
    volumes:
      - dvwa_db:/var/lib/mysql
    restart: unless-stopped

  dvwa:
    image: ghcr.io/digininja/dvwa:latest
    container_name: dvwa
    depends_on:
      - db
    ports:
      - "8081:80"
    environment:
      DB_SERVER: db
      DB_DATABASE: dvwa
      DB_USER: vulnerd
      DB_PASSWORD: vulnerd
    restart: unless-stopped

volumes:
  dvwa_db:
```

The services were deployed using:

```
docker compose up -d
```

10. Deployment Verification and Application Initialization

Container status was verified using:

```
docker ps
```

```
vulnerd@vulnerd:~/dvwa $ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES
0d3e2544c824   ghcr.io/digininja/dvwa:latest       "docker-php-entrypoi..." 7 minutes ago  Up 7 minutes  0.0.0.0:8081->80/tcp, [::]:8081->80/tcp  dvwa
0d4b6a30bac1   mariadb:10.6                        "docker-entrypoint.s..." 8 minutes ago  Up 7 minutes  3306/tcp                            dvwa-db
```

Successful output confirmed that both the DVWA and database containers were running.

DVWA was accessed using the host's private IP address:

```
http://192.168.0.239:8081
```

The application database was initialized through the built-in setup interface by selecting **Create / Reset Database**. Default DVWA credentials were used exclusively within the isolated lab environment.

11. Secure Remote Access

- **Solution:** Tailscale (WireGuard-based VPN)
- **Access Model:** Private VPN overlay

Justification

Because DVWA is intentionally insecure, exposing it to the public internet would introduce unacceptable risk. Instead, access to the lab was restricted using a private VPN, ensuring that only authorized team members could reach the environment.

No services in the VULNERD lab are publicly accessible.

11A. Tailscale Installation and Configuration

Tailscale was installed to provide encrypted, authenticated remote access to the lab environment.

```
curl -fsSL https://tailscale.com/install.sh | sh
sudo tailscale up
```

Once connected, the system was assigned a private VPN IP address. Remote team members access DVWA using this VPN-assigned IP address and port 8081.

12. Access Control Model

- **Local Network Access:** Restricted to private IP range
- **Remote Access:** Restricted to authenticated VPN clients
- **Public Exposure:** None

This access model balances usability for team members with responsible security practices.

13. Role Alignment (VULNERD Team)

The lab environment supports multiple cybersecurity roles:

- **Cloud Security Architect:** Environment design and isolation
- **Red Team Operator:** Exploitation and attack simulation
- **Detection Engineer:** Analysis of attack indicators and behavior
- **Cyber Threat Intelligence Analyst:** Vulnerability context and attribution
- **AI & Automation Lead:** Reporting, synthesis, and automation

14. Summary

The VULNERD lab environment demonstrates that realistic cybersecurity testing platforms can be deployed securely, efficiently, and cost-effectively. By combining a headless operating system, containerized vulnerable services, a dedicated database backend, and VPN-restricted access, the lab enables hands-on security experimentation while maintaining responsible risk controls.