TryHackMe Advent of Cyber 2025 Day 14 Challenge Report

Docker Containers & Container Escape Attacks

1. Executive Summary

This report documents the completion of Day 14 of the TryHackMe Advent of Cyber 2025 event. The challenge focused on Docker containerization fundamentals, container escape techniques, and privilege escalation through Docker socket exploitation. Successfully investigated Docker layers, performed container escape attack, escalated privileges to deployer container, and restored the defaced DoorDasher website service.

2. Challenge Overview

Objective: Investigate Docker layers, perform container escape, escalate privileges, and restore defaced Hopperoo website to original DoorDasher service.

Context: DoorDasher food delivery service defaced by attackers, renamed to 'Hopperoo'

3. Container Fundamentals

3.1 What Are Containers?

Containers solve modern application deployment problems by packing applications with their dependencies in isolated environments. Key problems solved:

- **Installation:** Eliminate configuration quirks across environments
- **Troubleshooting:** Separate application issues from environment issues
- **Conflicts:** Run multiple versions without dependency collisions

3.2 Containers vs Virtual Machines

Virtual Machines:

- Run on hypervisor software
- Include full guest OS
- Heavier but fully isolated
- Ideal for multiple OS or legacy apps

Containers:

- Share host OS kernel
- Isolate only applications and dependencies
- Lightweight and fast to start
- Excel at scalable, portable microservices

3.3 Microservices Architecture

Modern applications break down into parts based on business function (vs. monolithic single units). Benefits:

- Scale specific high-traffic parts independently
- Deploy updates to individual services
- Lightweight containers enable easy scaling

4. Docker Overview

4.1 What is Docker?

Docker is an open-source container engine that builds, runs, and manages containers by leveraging host OS kernel features (namespaces, cgroups). It uses:

- **Dockerfiles:** Text scripts defining app environments
- **Images:** Built from Dockerfiles, contain app layers
- **Containers:** Running instances of images

4.2 Container Engine Architecture

- **Client-Server Setup:** CLI tools (client) send requests to container daemon (server)
- **Runtime API:** Exposed via Unix sockets (runtime sockets)
- **Socket Communication:** Handles CLI and daemon traffic

5. Container Escape Attacks

5.1 Attack Definition

Container escape enables code inside a container to obtain rights or execute on the host kernel beyond its isolated environment. Attackers can:

- Create privileged containers
- Access resources beyond isolation
- Execute commands on host or other containers

5.2 Docker Socket Exploitation

Attack Vector: If attackers communicate with Docker socket (/var/run/docker.sock) from inside container, they can exploit the runtime.

Enhanced Container Isolation:

Docker's default setting blocks socket mounting to prevent malicious access. However, test containers sometimes need socket access, creating security risks.

6. Challenge Execution

6.1 Initial Reconnaissance

Command:

```
docker ps
```

Results: Identified running services including uptime-checker and deployer containers

Web Application: Defaced system 'Hopperoo' running at http://10.64.156.32:5001

6.2 Container Access & Socket Discovery

Step 1: Access uptime-checker container

```
docker exec -it uptime-checker sh
```

Step 2: Check socket access

```
ls -la /var/run/docker.sock
```

Finding: Docker socket accessible from container - vulnerability confirmed!

Step 3: Verify Docker API access

```
docker ps
```

Result: Successfully executed Docker commands from inside container - Docker Escape confirmed!

6.3 Privilege Escalation

Step 1: Access privileged deployer container

```
docker exec -it deployer bash
```

Step 2: Verify user privileges

```
whoami
```

Step 3: Navigate and locate flag

```
cd / ls cat flag.txt
```

Flag Located: Successfully accessed deployer container!

6.4 Service Recovery

Execute recovery script with sudo:

```
sudo /recovery_script.sh
```

Result: Website restored from Hopperoo to DoorDasher!

Verify restoration:

Refreshed http://10.64.156.32:5001 - DoorDasher service fully operational

7. Challenge Answers

7.1 Question 1: Docker Command

Question: What exact command lists running Docker containers?

Answer: docker ps

7.2 Question 2: Docker Image Definition

Question: What file is used to define the instructions for building a Docker image?

Answer: dockerfile

7.3 Question 3: Main Flag

Question: What's the flag?

Answer: THM{DOCKER_ESCAPE_SUCCESS}

7.4 Bonus Question: Secret Code

Question: There is a secret code on port 5002 (also the deployer password). Can you find it?

Method:

1. Access DoorDasher website on port 5002
2. Navigate to news page
3. Observe highlighted words in different color

Answer: DeployMaster2025!

8. Key Skills Developed

- Container fundamentals and architecture
- Docker commands and operations
- Container vs VM architecture understanding
- Docker socket exploitation techniques
- Container escape attack execution
- Privilege escalation through containers
- Docker security misconfigurations
- Container runtime API exploitation

9. Conclusion

Day 14 of the TryHackMe Advent of Cyber 2025 provided comprehensive training in Docker containerization and security. Successfully demonstrated container escape through Docker socket exploitation, privilege escalation to deployer container, and service recovery.

The challenge highlighted critical security risks when containers have access to Docker sockets. While Enhanced Container Isolation is Docker's default protection, test containers often require socket access for legitimate purposes. This creates potential attack vectors where compromised containers can escape isolation, create privileged containers, and execute commands on the host system. Understanding these risks is essential for securing containerized environments in production.

Challenge Status: **COMPLETED ✓**