

Containers - DoorDasher's Demise

Day 14

Continue your Advent of Cyber journey and learn about container security.

- [Container Security](#)

Container Security

Overview

Room URL: <https://tryhackme.com/room/container-security-aoc2025-z0x3v6n9m2>

Difficulty: Medium

Category: Containers

Date Completed: 12/14/2025

Objectives

- Learn how containers and Docker work, including images, layers, and the container engine
 - Explore Docker runtime concepts (sockets, daemon API) and common container escape/privilege-escalation vectors
 - Apply these skills to investigate image layers, escape a container, escalate privileges, and restore the DoorDasher service
 - DO NOT order “Santa's Beard Pasta”
-

Table of Contents

[Introduction](#)

[Walk Through](#)

[Lessons Learned](#)

[Resources](#)

Introduction

DoorDasher's food delivery service has fallen victim to a sophisticated attack—the Hopperoo website now displays defaced content instead of the legitimate service. Your mission is to investigate the Docker infrastructure, identify the security vulnerability, and restore the original

application. This challenge demonstrates a critical real-world security risk: **improper Docker socket exposure** that enables container escape attacks. By exploiting overly permissive container configurations and weak credentials, you'll navigate through multiple containers, escalate privileges, and execute a recovery script to save the day. This hands-on exercise reveals how a single misconfiguration in container isolation settings can cascade into complete infrastructure compromise.

Walk Through

1. Spin up target machine
2. Connect to VPN
3. What exact command lists running Docker containers?
 1. `docker ps`
4. What file is used to define the instructions for building a Docker image?
 1. `dockerfile`
5. What's the flag?
 1. `docker exec -it uptimechecker sh`
 2. `docker exec -it deployer bash`
 3. `whoami`
 4. `pwd`
 5. `ls`
 6. `cd ../`
 7. `ls`
 8. `cat flag.txt`
 9. `sudo ./recovery_script.sh`

```

deployer@e58624a27d1c:/app$ ls
deployer@e58624a27d1c:/app$ pwd
/app
deployer@e58624a27d1c:/app$ cd ../
deployer@e58624a27d1c:/# ls
app  boot  etc      home  lib32  libx32  mnt  proc      root  sbin  sys  usr
bin  dev   flag.txt lib  lib64  media  opt  recovery_script.sh  run  srv  tmp  var
deployer@e58624a27d1c:/# cat flag.txt
THM
deployer@e58624a27d1c:/# ^C
deployer@e58624a27d1c:/# sudo ./recovery_script.sh
=== DoorDasher Recovery Script ===
Restoring original DoorDasher configuration...
Switching to DoorDasher skin (version 1)...
Copying version file to dasherapp container...
Successfully copied 2.05kB to dasherapp:/app/version.txt
Restarting dasherapp container...
dasherapp
Recovery complete! DoorDasher has been restored.
The hackers have been defeated!
deployer@e58624a27d1c:/# █

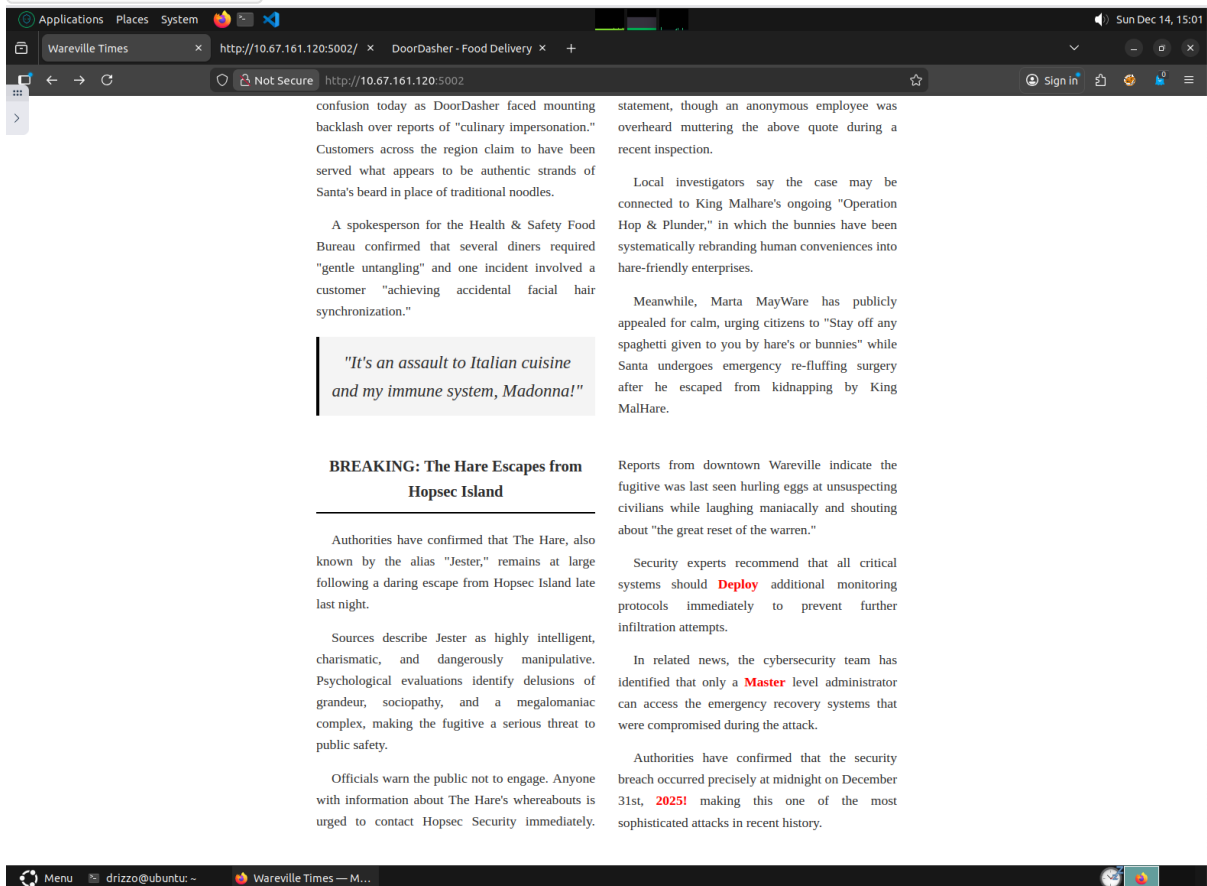
```

10.

6. Bonus Question: There is a secret code contained within the news site running on port 5002; this code also happens to be the password for the deployer user! They should definitely change their password. Can you find it?

1. open <website>:5002

2. DeployMaster2025!



3.

Lessons Learned

- **Docker Socket Exposure is Critical:** Exposing `/var/run/docker.sock` to containers without Enhanced Container Isolation enabled creates a direct pathway for attackers to escape container boundaries and execute arbitrary commands on the host system. Always use Enhanced Container Isolation and apply the principle of least privilege when granting container permissions.
 - **Container Security Requires Multi-Layered Defense:** This challenge demonstrates that a single misconfiguration (socket access) combined with weak credentials (`DeployMaster2025!`) can lead to complete infrastructure compromise. Effective container security demands careful attention to isolation settings, privilege escalation prevention, secrets management, and regular security audits of running services.
-

Resources

[TryHackMe](#)

[Docker Documentation](#)

[Docker Security Best Practices](#)

[Container Escape Techniques](#)