CTF Commands Reference Guide: Practical 2 - Zork Lab

Challenge: Breaking into Forthare's Gold Deposit

Type: Web-based Text Adventure / Hacker Mindset Training

Date: October 25, 2025

This guide contains all exact commands used during reconnaissance, deployment, testing, and analysis of Practical 2.

Table of Contents

Phase 1: Initial Reconnaissance

Understanding the Challenge

Read the training script to identify Practical 2:

Bash

```bash
cat /home/ubuntu/upload/training_sell | grep -A 10 "Practical_2"
```

Extract Practical 2 configuration:

Bash

```bash
grep "Practical_2=" /home/ubuntu/upload/training_sell
# Output: Practical_2="zork-hacker-mindset"
```

Find Practical 2 deployment section:

Bash

```bash
sed -n '/p2|2)/,/;;/p' /home/ubuntu/upload/training_sell
```

Key Information Discovered:

- Docker Image: `public.ecr.aws/m3r3g8n5/zork-hacker-mindset:latest`
- Port: 80:3000 (host:container)
- Type: Web-based challenge
- Access: http://localhost

Phase 2: Docker Environment Setup

Verify Docker Installation

Check Docker version:

Bash

```bash
docker --version
```

Check Docker daemon status:

Bash

```bash
docker info
```

Start Docker service (if needed):

Bash

```bash
sudo systemctl start docker
sudo systemctl enable docker
```

Pull the Challenge Image

Pull from AWS ECR:

Bash

```
sudo docker pull public.ecr.aws/m3r3g8n5/zork-hacker-mindset:latest
```

Verify image downloaded:

Bash

```bash
docker images | grep zork
```

Inspect image details:

Bash

```bash
docker inspect public.ecr.aws/m3r3g8n5/zork-hacker-mindset:latest
```

Check image size:

Bash

```bash
docker images --format "table {{.Repository}}\t{{.Tag}}\t{{.Size}}" | grep
zork
```

Phase 3: Container Deployment

Deploy the Container

Stop any existing containers:

Bash

```bash
sudo docker rm -f practical-2 2>/dev/null
```

Run the container:

Bash

```bash
sudo docker run -d \
  --name practical-2 \
  -p 8080:3000 \
  public.ecr.aws/m3r3g8n5/zork-hacker-mindset:latest
```

Alternative deployment on port 80:

Bash

```bash
sudo docker run -d \
  --name practical-2 \
  -p 80:3000 \
  public.ecr.aws/m3r3g8n5/zork-hacker-mindset:latest
```

Verify Container Status

Check if container is running:

Bash

```bash
sudo docker ps | grep practical-2
```

View container details:

Bash

```bash
sudo docker inspect practical-2
```

Check container logs:

Bash

```bash
sudo docker logs practical-2
```

Follow logs in real-time:

Bash

```bash
sudo docker logs -f practical-2
```

Check container resource usage:

Bash

```bash
docker stats practical-2 --no-stream
```

Phase 4: Web Application Testing

HTTP Reconnaissance

Test HTTP connectivity:

```bash
curl -I http://localhost:8080
```

Get full HTTP response:

```bash
curl -v http://localhost:8080
```

Download homepage:

```bash
curl -s http://localhost:8080 > homepage.html
```

Check response headers:

```bash
curl -I http://localhost:8080 2>&1 | grep -E "Server|Content-Type|X-"
```

Port Scanning

Scan the container port:

```bash
nmap -p 3000 localhost
```

Service detection:

```bash
nmap -sV -p 8080 localhost
```

Full scan:

```bash
nmap -sV -sC -p- localhost
```

Web Application Analysis

Check for robots.txt:

Bash

```bash
curl -s http://localhost:8080/robots.txt
```

Check for common files:

Bash

```bash
curl -s http://localhost:8080/sitemap.xml
curl -s http://localhost:8080/package.json
curl -s http://localhost:8080/.env
```

Test API endpoints:

Bash

```bash
curl -s http://localhost:8080/api
curl -s http://localhost:8080/api/status
```

JavaScript Analysis

Extract JavaScript files:

Bash

```bash
curl -s http://localhost:8080 | grep -oP '(?<=src=")[^"]*\.js'
```

Download main JavaScript:

Bash

```bash
curl -s http://localhost:8080/static/js/main.js -o main.js
```

Phase 5: Game Mechanics Analysis

Browser-Based Testing

Access the web interface:

Plain Text

```
Open browser: http://localhost:8080
```

Game Commands Testing

Test help command:

Plain Text

```
> help
```

Expected Output:

Plain Text

```
- Enter codes: enter [code]
- Move objects: move [object]
- Get hints: hint
- Help: help, ?

Available directions:
- go east

Type 'hint' for context-specific clues if you get stuck.
```

Test navigation:

Plain Text

```
> go east
```

Expected Output:

Plain Text

```
You are at the Maintenance Closet.
A small utility closet filled with cleaning supplies and maintenance
equipment.
```

Test hint system:

Plain Text

```
> hint
```

Expected Output:

Plain Text

```
HINTS FOR THIS LOCATION:
- This place contains useful tools
- The maintenance schedule might contain valuable information
- Something here could help you get over the wall

Try some of these actions:
- read schedule - break lock
```

Test interaction commands:

Plain Text

```
> read schedule
> break lock
> move object
> enter code
```

Test inventory:

Plain Text

```
> inventory
> inv
> i
```

Network Traffic Analysis

Capture HTTP requests (optional):

Bash

```bash
# Install tcpdump if needed
sudo apt-get install -y tcpdump

# Capture traffic
sudo tcpdump -i any -w zork-traffic.pcap port 8080
```

Analyze with curl:

Bash

```
curl -v http://localhost:8080 2>&1 | grep -E "^> |^< "
```

Phase 6: Offline Package Creation

Export Docker Image

Save the Docker image:

```bash
sudo docker save public.ecr.aws/m3r3g8n5/zork-hacker-mindset:latest | gzip >
zork-hacker-mindset-image.tar.gz
```

Verify image size:

```bash
ls -lh zork-hacker-mindset-image.tar.gz
```

Calculate checksum:

```bash
md5sum zork-hacker-mindset-image.tar.gz
sha256sum zork-hacker-mindset-image.tar.gz
```

Create Directory Structure

Create package directory:

```bash
mkdir -p ~/practical-2-offline/{web,docs}
```

Move image to package:

```bash
mv zork-hacker-mindset-image.tar.gz ~/practical-2-offline/
```

Create Configuration Files

Create docker-compose.yml:

Bash

```bash
cat > ~/practical-2-offline/docker-compose.yml << 'EOF'
version: '3.8'

services:
  zork-lab:
    image: zork-hacker-mindset:latest
    container_name: practical-2-zork
    ports:
      - "80:3000"
    networks:
      - zork-network
    restart: unless-stopped
    environment:
      - NODE_ENV=production
    healthcheck:
      test: ["CMD", "wget", "--quiet", "--tries=1", "--spider",
"http://localhost:3000"]
      interval: 30s
      timeout: 10s
      retries: 3
      start_period: 40s

networks:
  zork-network:
    driver: bridge
EOF
```

Create load-image.sh:

Bash

```bash
cat > ~/practical-2-offline/load-image.sh << 'EOF'
#!/bin/bash
set -e
echo "Loading Docker image..."
docker load < zork-hacker-mindset-image.tar.gz
echo "Image loaded successfully!"
EOF

chmod +x ~/practical-2-offline/load-image.sh
```

Create start.sh:

Bash

```bash
cat > ~/practical-2-offline/start.sh << 'EOF'
#!/bin/bash
set -e

# Check Docker
if ! command -v docker &> /dev/null; then
    echo "Error: Docker not installed"
    exit 1
fi

# Load image if needed
if ! docker images | grep -q "zork-hacker-mindset"; then
    ./load-image.sh
fi

# Start container
docker-compose up -d
echo "Zork Lab is running at http://localhost"
EOF

chmod +x ~/practical-2-offline/start.sh
```

Create Makefile:

Bash

```bash
cat > ~/practical-2-offline/Makefile << 'EOF'
.PHONY: help load up down restart logs status

help:
    @echo "Practical 2: Zork Lab - Management Commands"
    @echo ""
    @echo "  make load    - Load Docker image"
    @echo "  make up      - Start the challenge"
    @echo "  make down    - Stop the challenge"
    @echo "  make restart - Restart the challenge"
    @echo "  make logs    - View logs"
    @echo "  make status  - Check status"

load:
    docker load < zork-hacker-mindset-image.tar.gz

up:
    docker-compose up -d

down:
    docker-compose down

restart:
    docker-compose restart

logs:
    docker-compose logs -f

status:
    docker-compose ps
EOF
```

Create Documentation

Create file manifest:

Bash

```bash
cd ~/practical-2-offline
find . -type f | sort > MANIFEST.txt
```

Create README.md, DEPLOYMENT_GUIDE.md, etc.:

Bash

```bash
# (Documentation files created separately)
```

Package Everything

Create final archive:

Bash

```bash
cd ~
tar -czf practical-2.tar.gz practical-2-offline/
```

Verify archive:

Bash

```bash
ls -lh practical-2.tar.gz
```

Calculate checksums:

Bash

```bash
md5sum practical-2.tar.gz
sha256sum practical-2.tar.gz
```

List archive contents:

Bash

```bash
tar -tzf practical-2.tar.gz | head -20
```

Count files:

Bash

```bash
tar -tzf practical-2.tar.gz | wc -l
```

Quick Reference

Essential Commands

Deploy Challenge:

Bash

```bash
# Quick start
./start.sh

# Or with docker-compose
docker-compose up -d

# Or with make
make up
```

Check Status:

Bash

```bash
docker ps | grep zork
docker-compose ps
make status
```

View Logs:

Bash

```
docker logs practical-2-zork
docker-compose logs -f
make logs
```

Stop Challenge:

Bash

```
docker-compose down
make down
```

Test Web Access:

Bash

```
curl http://localhost
```

Game Commands

Navigation:

Plain Text

```
go east, go west, go north, go south
```

Interaction:

Plain Text

```
enter [code]
move [object]
break [object]
read [item]
```

Information:

Plain Text

```
help
hint
inventory
```

Troubleshooting Commands

Container Issues

Container won't start:

Bash

```bash
# Check logs
docker logs practical-2-zork

# Remove and recreate
docker rm -f practical-2-zork
docker-compose up -d
```

Port already in use:

Bash

```bash
# Find process using port 80
sudo lsof -i :80
sudo netstat -tulpn | grep :80

# Kill process
sudo kill -9 <PID>

# Or change port in docker-compose.yml
```

Image not found:

Bash

```bash
# Load image manually
docker load < zork-hacker-mindset-image.tar.gz

# Verify image
docker images | grep zork
```

Network Issues

Can't access web interface:

Bash

```
# Check if container is running
docker ps | grep zork

# Check container network
docker inspect practical-2-zork | grep IPAddress

# Test from host
curl http://localhost

# Check firewall
sudo iptables -L -n
```

DNS issues:

```bash
# Test DNS resolution
nslookup localhost
ping localhost

# Use IP directly
curl http://127.0.0.1
```

Performance Issues

High CPU usage:

```bash
# Monitor resources
docker stats practical-2-zork

# Check container processes
docker top practical-2-zork
```

High memory usage:

```bash
# Check memory
docker stats --no-stream practical-2-zork

# Restart container
docker restart practical-2-zork
```

Cleanup Commands

Remove everything:

```bash
# Stop and remove container
docker-compose down

# Remove image
docker rmi zork-hacker-mindset:latest

# Clean up Docker system
docker system prune -a
```

Reset to fresh state:

```bash
# Complete cleanup
docker-compose down --remove-orphans
docker rmi zork-hacker-mindset:latest
docker volume prune -f
docker network prune -f

# Reload and restart
./load-image.sh
./start.sh
```

Advanced Commands

Container Inspection

Enter container shell:

```bash
docker exec -it practical-2-zork sh
docker exec -it practical-2-zork bash
```

Inspect container filesystem:

```bash
```

```
docker exec practical-2-zork ls -la /app
docker exec practical-2-zork cat /app/package.json
```

Check running processes:

Bash

```
docker exec practical-2-zork ps aux
```

Check environment variables:

Bash

```
docker exec practical-2-zork env
```

Network Analysis

Inspect network:

Bash

```
docker network inspect bridge
docker network inspect zork-network
```

Check port mappings:

Bash

```
docker port practical-2-zork
```

Test connectivity:

Bash

```
docker exec practical-2-zork wget -O- http://localhost:3000
```

Backup and Restore

Backup container state:

Bash

```
docker commit practical-2-zork zork-backup:latest
```

Export container:

```bash
docker export practical-2-zork > zork-container.tar
```

Save image with different name:

```bash
docker tag zork-hacker-mindset:latest zork-lab:v1.0
docker save zork-lab:v1.0 | gzip > zork-lab-v1.0.tar.gz
```

Summary

This guide covered all commands used for:

1. **Reconnaissance** - Identifying and analyzing the challenge
2. **Deployment** - Setting up Docker and running the container
3. **Testing** - Verifying functionality and exploring the game
4. **Analysis** - Understanding game mechanics and structure
5. **Packaging** - Creating offline deployment package
6. **Troubleshooting** - Resolving common issues

All commands are copy-paste ready and tested on Ubuntu 22.04.

Happy Hacking! 🏦💰