Author: Maxwell Bosiako Antwi

Project Title: Deploying and Monitoring a High-Interaction SSH/Telnet Honeypot with Cowrie
Environment: Kali Linux with Docker

Date Completed: 11th May 2025

---

1. Project Objective

The goal of this project was to deploy, simulate, and monitor a high-interaction honeypot using **Cowrie**, with the purpose of learning how attackers interact with exposed SSH/Telnet services and understanding how logging and deception environments work.

2. Tools & Technologies Used

- **Docker** (Containerization platform)
- **Cowrie** (SSH/Telnet honeypot)
- **SSH client**
- **Linux Terminal (Kali Linux)**
- **Docker logging**
- **Real-time session monitoring**

3. Deployment Process

Step 1: Environment Setup

- Updated and started Docker service:

```
sudo systemctl start docker

sudo systemctl enable docker
```

Step 2: Pulled the Cowrie Docker Image

- Retrieved the official Cowrie image from Docker Hub:

```
sudo docker pull cowrie/cowrie
```

Step 3: Deployed Cowrie Container

- Ran Cowrie exposing SSH on port 2222 and Telnet on port 2223:

```
sudo docker run -d -p 2222:2222 -p 2223:2223 --name cowrie cowrie/cowrie
```

Step 4: Accessed the Honeypot Shell

- Connected to the honeypot as root using SSH:

```
ssh root@localhost -p 2222
```

- Provided the default password '*admin*' to log into the **fake shell environment** provided by Cowrie. While in the shell environment, I performed couple of directory traversal to see what was inside the cowrie.

4. Monitoring & Analysis

Log Verification

- Inside the host terminal, the following command was used to view captured events:

```
docker logs cowrie
```

This provided detailed events that happened while I was logged into the cowrie shell.

Real-Time Log Streaming

- I wanted to have continuous live monitoring while I still logged in the shell. So I used the command below then I logged back into the shell:

```
docker logs -f cowrie
```

What Was Captured:

- IP address of connecting host
- Username and password used during login
- Timestamp of connection

- Commands entered inside the honeypot shell
- The simulated output Cowrie provided back to the attacker

5. Observations & Behavior

- The fake shell convincingly mimicked a real Linux terminal environment.
- Every single input (including failed logins, typos, commands, etc.) was logged in detail.
- All responses were faked by Cowrie to deceive the intruder and prolong engagement.

This emulation provides valuable insights into **post-exploitation behavior** like:

- Privilege escalation attempts
- Enumeration techniques (ls, cat, whoami)
- File download attempts using wget, curl, etc.

6. Skills & Knowledge Gained

- How to deploy and manage containers using Docker
- Understanding Cowrie's role in a defensive deception strategy
- Real-time monitoring and log analysis for threat behavior
- Using honeypots to gather TTPs (Tactics, Techniques, and Procedures)
- Basics of attacker profiling and forensic logging

7. Challenges Faced

- **Docker exec failure**: Initially I tried to access the container via /bin/bash but Cowrie's container doesn't include an interactive shell environment. Solved by relying on mounted logs and docker logs.
- **Understanding honeypot behavior**: Required studying how Cowrie simulates systems and why it doesn't behave like a normal Linux shell.
- **Networking confusion**: Needed clarity on the difference between real shell and Cowrie's emulated shell, especially when testing SSH connections to port 2222.

8. Key Takeaways

- A honeypot like Cowrie is not just bait, it is a **controlled lab for studying real-world threats**.
- Logging attacker behavior provides invaluable context for defenders, SOC analysts, and blue teams.
- Honeypots must be deployed securely (isolated) and monitored frequently.
- Tools like Cowrie can be connected to larger log platforms like Splunk or ELK for full-scale analysis.