EXP-2: Experiment for Set up a honey pot and monitor the honey pot on the network

What is a Honeypot?

A **honeypot** is a **fake system or service** designed to look vulnerable, so that **attackers are tricked into interacting** with it.

We can then **monitor, analyze, and log their behavior** — like open ports they probe, malware they upload, etc.

Step-by-Step: Set Up and Monitor a Honeypot on the Network

Step 1: Choose the Right Environment

You can run your honeypot on:

- A **Virtual Machine (VM)** (best for isolation)
- A **Docker container**
- A **physical device** (like Raspberry Pi)
- In the **cloud** (with caution — it can get attacked fast)

For beginners, a **VM running Kali Linux or Ubuntu** is recommended.

Step 2: Install a Honeypot Tool

There are many open-source honeypots. Here are 3 good options:

Honeypot Name	Purpose	Easy to Use?
Cowrie	SSH & Telnet trap	⭐⭐⭐⭐
Dionaea	Malware collection	⭐⭐⭐
Kippo	Older SSH honeypot	⭐⭐
Honeyd	Simulates entire fake networks	⭐⭐⭐

We'll use **Cowrie** for this example — it's a powerful SSH honeypot.

Step 3: Install Cowrie (SSH Honeypot)

Step-by-Step on Ubuntu/Kali Linux:

Step 1: Install dependencies

```
sudo apt update
sudo apt install git python3-venv python3-dev libssl-dev libffi-dev build-essential
```

Step 2: Clone the Cowrie repository

```
git clone https://github.com/cowrie/cowrie.git
cd cowrie
```

Step 3: Create a virtual environment

```
python3 -m venv cowrie-env
source cowrie-env/bin/activate
```

Step 4: Install Python requirements

```
pip install --upgrade pip
pip install -r requirements.txt
```

Step 5: Copy default config

```
cp etc/cowrie.cfg.dist etc/cowrie.cfg
```

Step 6: Start honeypot

```
bin/cowrie start
```

Now Cowrie will listen on a **fake SSH port (usually 2222)**.

Step 4: Monitor the Honeypot

Cowrie logs **everything attackers do**, like:

- Login attempts
- Commands typed
- Files uploaded

You can see logs here:

```
tail -f log/cowrie.log
```

Example log:

2025-06-27T10:35Z login attempt with username: root and password: 123456

You can also record session replays:

bin/playlog log/tty/...

Step 5: Redirect Real Port 22 to Cowrie Port 2222

You can use iptables to forward real SSH traffic:

sudo iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --to-ports 2222

So when attackers try to connect to port 22, it goes to Cowrie.

⚠️ **Important**: Do not expose this on your production system unless isolated properly.

Step 6: Watch in Real-Time (Optional Monitoring Tools)

Use **tools to visualize attacks**:

- **Kibana + Elasticsearch**: Great for dashboards
- **Grafana + Loki**: Logs in real-time
- **DShield Honeypot project**: Share data with global researchers

Summary of Commands

```
# Setup
    git clone https://github.com/cowrie/cowrie.git
    cd cowrie
    python3 -m venv cowrie-env
    source cowrie-env/bin/activate
    pip install -r requirements.txt
    cp etc/cowrie.cfg.dist etc/cowrie.cfg

# Start honeypot
    bin/cowrie start

# View logs
    tail -f log/cowrie.log

# Optional: redirect SSH port
sudo iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --to-ports 2222
```

Use Case: What You Learn from a Honeypot?

- Common usernames/passwords attackers try
- Typical scripts or malware they upload
- Tools attackers use (e.g., wget, curl, netcat)
- Their IP addresses and behavior patterns

Note: Security Tip

- A honeypot should never be run on a critical production system.
- Use it in isolated networks or VMs only. Honeypots invite attacks, so safety first!

Additional information

Indicators That Cowrie Started Successfully:

1. ✅ Starting cowrie: [twistd ... cowrie]...
   ➤ This shows Cowrie is starting the Twisted-based SSH honeypot engine.
2. ✅ No **fatal errors** or FAILED status reported
   ➤ Only a **deprecation warning**, not a blocking issue.
3. ✅ You see the **Twisted command** running (that's the Cowrie engine):

   twistd --umask=0022 --pidfile=var/run/cowrie.pid ...
4. ✅ It uses the Python virtual environment correctly:

   Using activated Python virtual environment "/home/kali/cowrie/cowrie-env"

⚠️ **About the Warning:**

CryptographyDeprecationWarning: TripleDES ...

This is **just a warning**, not an error. It's telling you that the 3DES cipher is outdated and will be removed in the future. Cowrie is still running, but eventually you'll need to update Cowrie or Twisted to avoid future issues.

How to Verify It's Listening (Optional Check):

Option 1: Check port 22 is open (default Cowrie port):

```
sudo netstat -tulnp | grep 22
```

Option 2: Check Cowrie logs:

```
tail -f log/cowrie.log
```

You'll see messages if anyone connects (or tries to).

Conclusion:

Cowrie honeypot is started and listening, despite the deprecation warning. You're good to go for capturing SSH brute-force or recon attempts.

Would you like:

- Help changing Cowrie's port (e.g., from 2222 to 22)?
- Visual log analysis?
- Guide to integrate with ELK or Grafana for monitoring?

In **Cowrie**, the **username and password** are **not fixed** — Cowrie is a **fake SSH server (honeypot)** designed to **accept almost any credentials** to log attacker behavior.

Default Behavior of Cowrie:

1. Cowrie accepts any username/password (unless configured otherwise):

- Example:

```
ssh root@<honeypot-ip>
password: 1234
```

Cowrie **logs this attempt**, even if the credentials are wrong or random.

2. It will simulate a successful login for:

- Common usernames: root, admin, user, etc.
- Any password (even incorrect ones)
- Attacker sees a **fake shell** (emulated environment)

Purpose:

The goal is **not to stop attackers**, but to **observe their commands, tools, and methods** inside a fake system.

Where are credentials stored?

Cowrie logs login attempts to:

/opt/cowrie/var/log/cowrie/cowrie.json

Or:

~/cowrie/log/cowrie.json

You can see:

- username
- password
- source IP
- timestamp

You can also define valid/fake usernames/passwords in the config:

cowrie/etc/cowrie.cfg

Look for:

[honeypot]

auth_class = cowrie.core.auth.HoneyPotAuth

[auth]

Define fake accounts here if needed

userdb_file = etc/userdb.txt

Create userdb.txt:

makefile

admin:x:root

vijay:password123:root

This way you **control fake accounts**.

Summary:

Question	Answer
Is username = password by default?	❌ No, Cowrie accepts almost any
Does it allow login?	✅ Yes, to observe attacker behavior
Can I set fake accounts?	✅ Yes, via userdb.txt
Are attempts logged?	✅ Yes, in cowrie.log and cowrie.json

Would you like help:

- Viewing captured usernames/passwords?
- Sending logs to a central server (SIEM)?
- Analyzing attacker behavior using Cowrie?

(cowrie-env)─(kali㉿kali)-[~/cowrie]

This means:

- You're inside the cowrie project directory.
- The **Python virtual environment** (cowrie-env) is activated.
- You are ready to run or modify Cowrie.

How to connect again to this environment later (if closed)

When you reboot or close the terminal, the environment will be **deactivated**. Here's how to **reconnect and activate Cowrie's environment** manually:

Step-by-step:

1. **Go to Cowrie directory:**

```
cd ~/cowrie
```

2. **Activate the virtual environment:**

```
source cowrie-env/bin/activate
```

You'll see this again in your prompt: (cowrie-env)─(kali㉿kali)─[~/cowrie]

3. **Start Cowrie:**

```
bin/cowrie start
```

If You See Deprecation Warnings (like TripleDES):

These are **safe to ignore for now**, but to suppress them, you can:

```
export PYTHONWARNINGS="ignore::CryptographyDeprecationWarning"
```

Or you can **upgrade** the Twisted and cryptography packages in the virtual environment later.

Optional: List Current Cowrie Logs

To see who tried connecting:

```
tail -f var/log/cowrie/cowrie.log
```

Or JSON log:

```
tail -f var/log/cowrie/cowrie.json
```