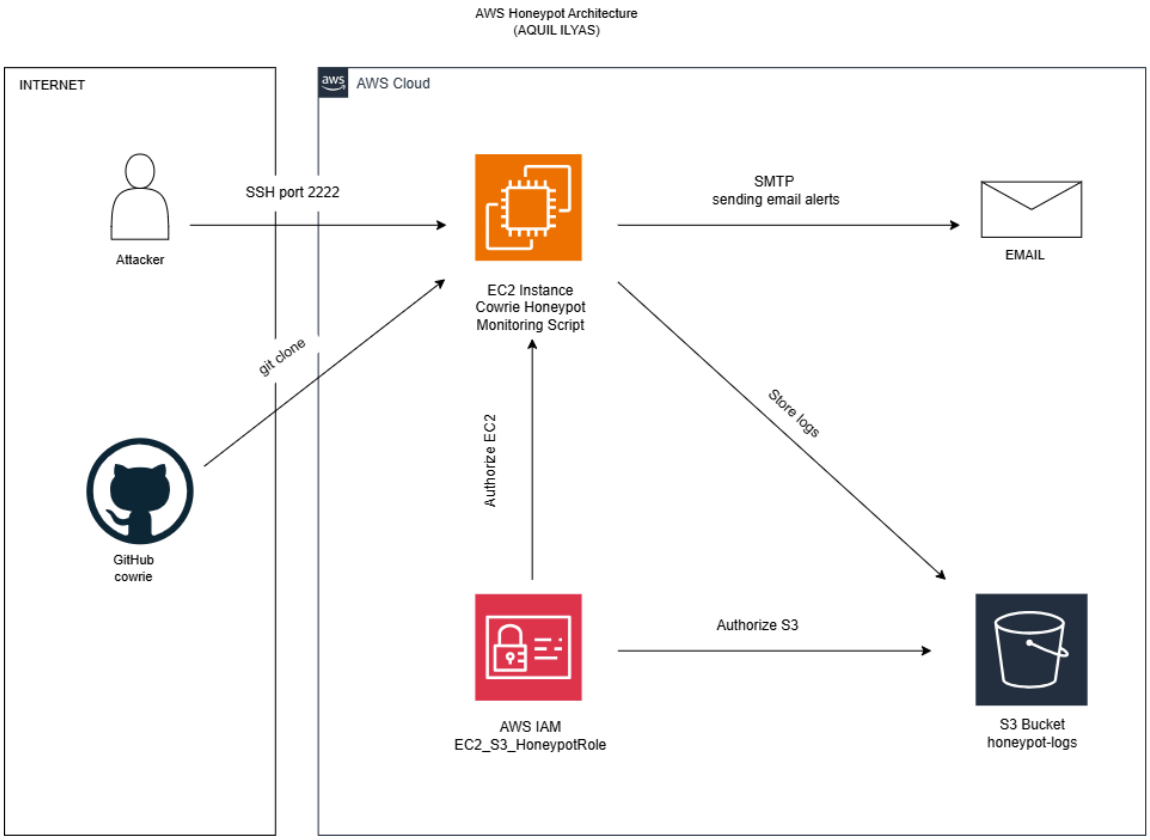
HONEYPOT DEPLOYMENT ON AWS

AQUIL ILYAS



Introduction

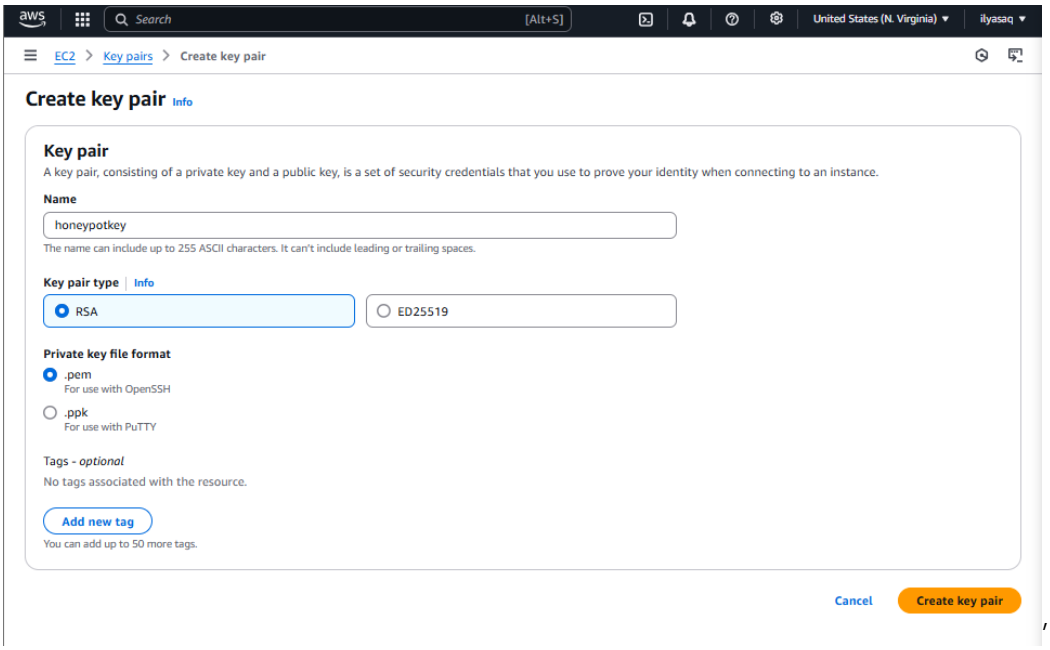
The growing threat of cyberattacks necessitates proactive defense mechanisms, and honeypots serve as an invaluable tool for cybersecurity professionals. In this project, we focus on deploying a Cowrie honeypot on Amazon Web Services (AWS), a cloud computing platform known for its scalability and flexibility. The purpose of this honeypot is to simulate vulnerabilities in a controlled environment, attracting malicious actors and logging their activity for analysis. By leveraging AWS, we ensure that the honeypot is scalable, reliable, and capable of running efficiently in a cloud-based infrastructure. In addition, the project enhances its functionality by incorporating attacker IP geolocation, using AWS services like S3 for data storage. The insights gained from analyzing this data will aid in understanding attack patterns and improving overall cybersecurity measures.

Deployment Steps

In this section, I outline the steps I followed to deploy the Cowrie honeypot on AWS. These steps include setting up the AWS EC2 instance, installing the required software, configuring Cowrie, and setting up monitoring and logging to capture attacker data.

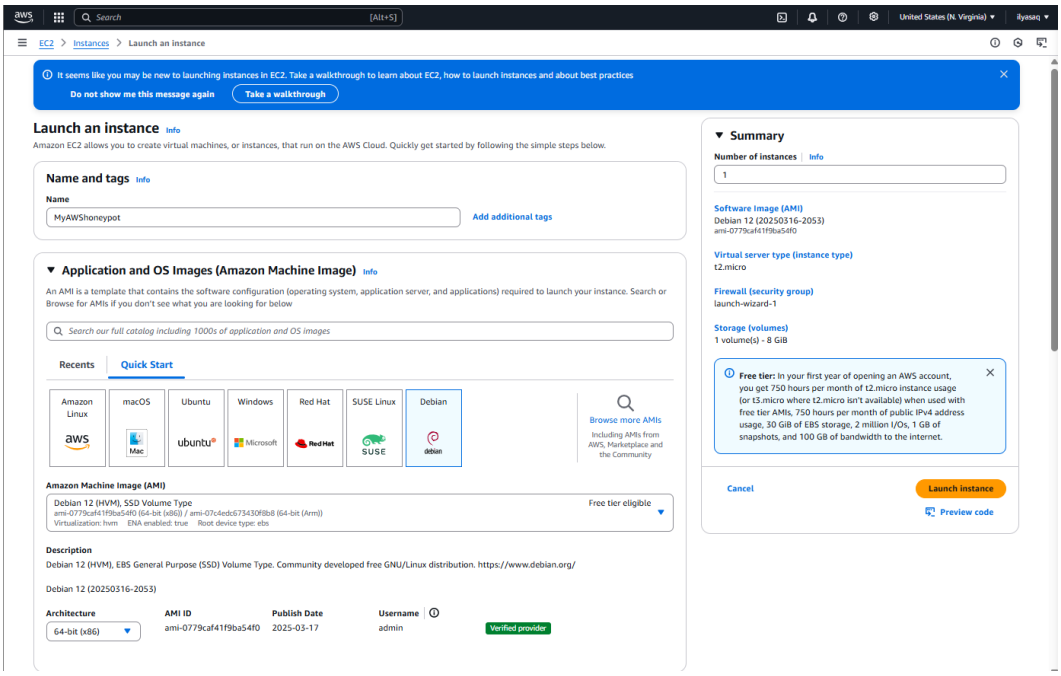
1. Creating a Key Pair :

To connect to EC2 instance that I will setup securely, I created a key pair. I chose the RSA type and the format, then I clicked "Create key pair."
The file downloaded automatically to my browser. I'll use it later to access the instance.

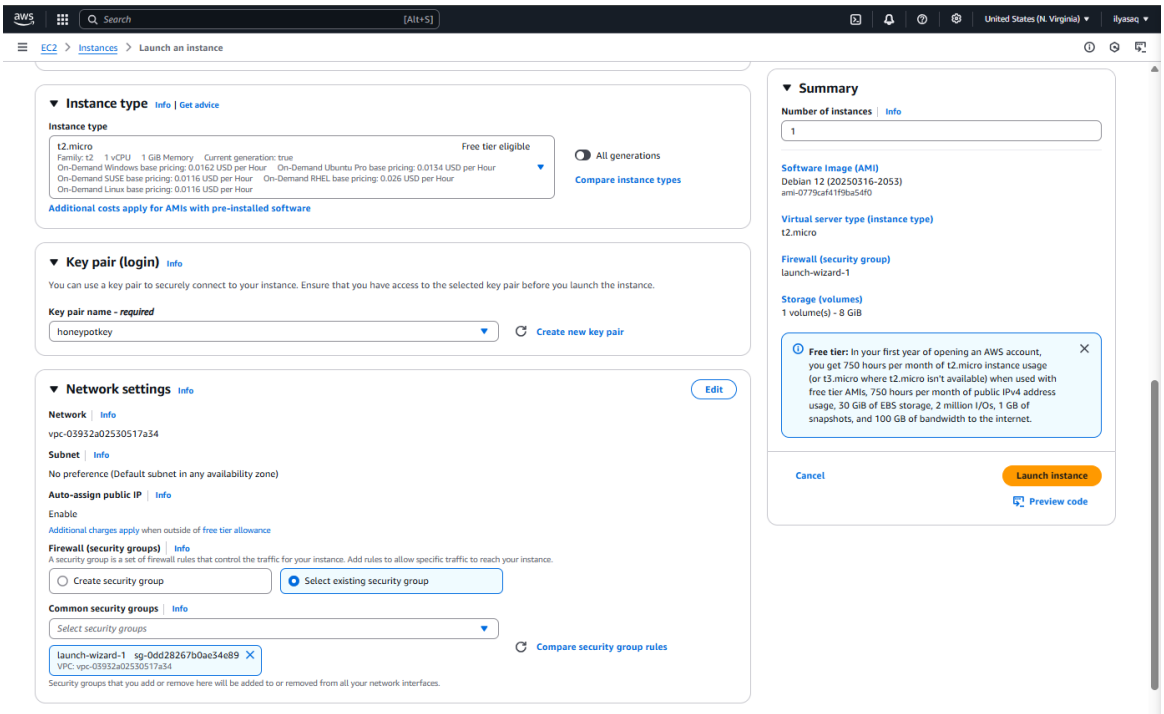


2. Setting Up the EC2 Instance on AWS

I signed into my AWS account and searched for EC2 in the AWS Management Console. I selected “Launch Instance” to begin setting up a new virtual machine.



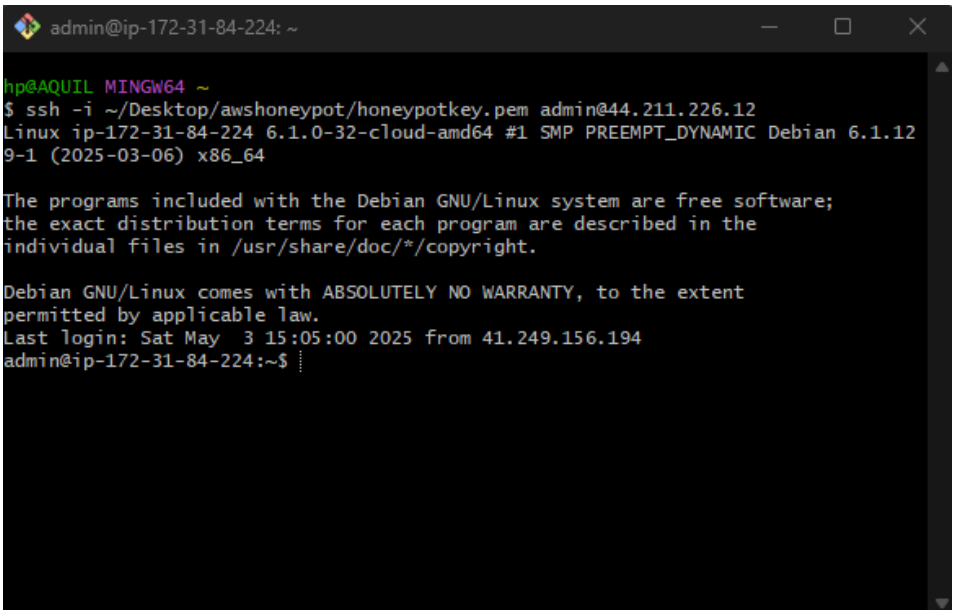
During the setup process, I chose the desired configuration for my instance, including the Amazon Machine Image (AMI), instance type, key pair, storage, and security group settings.



3. SSH Into the Instance :

To access my EC2 instance, I used Git Bash and connected via SSH using the command:

```
ssh -i /path/to/key.pem ec2-user@<EC2-Public-IP>
```



I made sure the key file had the right permissions by using :

```
chmod 400 honeypotkey.pem
```

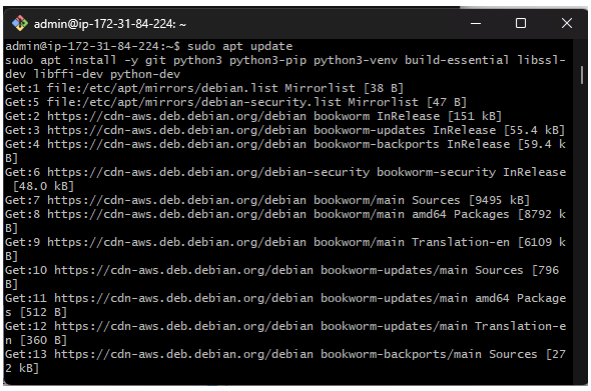
Then I replaced the file path and IP address with the correct ones for my instance. This let me log in to the server from my computer.

4. Install Required Packages:

After connecting to the instance, I updated the system and installed the necessary packages using the following commands:

```
sudo apt update

sudo apt install -y git python3 python3-pip python3-venv build-essential libssl-dev
```



This installed tools like Git, Python 3, pip, and other important libraries needed for development.

5. Clone the Cowrie Repository:

Next, I cloned the Cowrie repository to the EC2 instance using the following commands:

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

cd cowrie
```

```
admin@ip-172-31-86-181: ~/cowrie
admin@ip-172-31-86-181:~$ git clone https://github.com/cowrie/cowrie.git
cd cowrie
Cloning into 'cowrie'...
remote: Enumerating objects: 19040, done.
remote: Counting objects: 100% (161/161), done.
remote: Compressing objects: 100% (96/96), done.
remote: Total 19040 (delta 141), reused 65 (delta 65), pack-reused 18879 (from 4
)
Receiving objects: 100% (19040/19040), 10.47 MiB | 26.61 MiB/s, done.
Resolving deltas: 100% (13377/13377), done.
admin@ip-172-31-86-181:~/cowrie$
```

This downloaded the Cowrie honeypot project and moved me into its folder to begin setup.

6. Create a Python Virtual Environment :

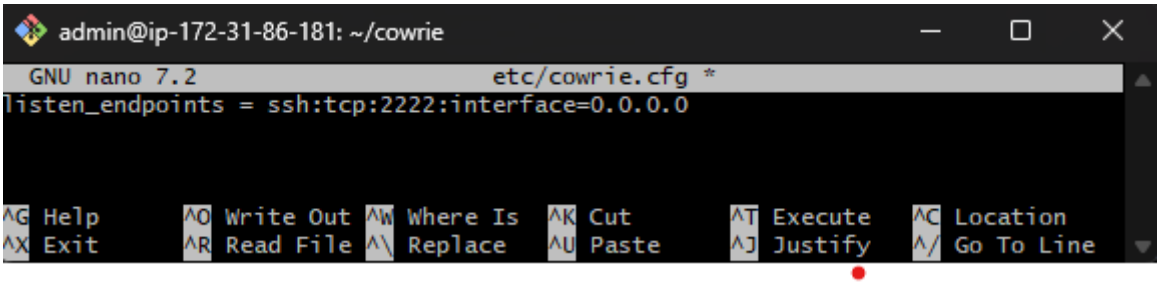
```
admin@ip-172-31-86-181: ~/cowrie
admin@ip-172-31-86-181:~/cowrie$ python3 -m venv cowrie-env
source cowrie-env/bin/activate
(cowrie-env) admin@ip-172-31-86-181:~/cowrie$
```

7. Install Python Dependencies:

```
admin@ip-172-31-86-181: ~/cowrie
(cowrie-env) admin@ip-172-31-86-181:~/cowrie$ pip install -r requirements.txt
Collecting attrs==25.3.0
  Downloading attrs-25.3.0-py3-none-any.whl (63 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 63.8/63.8 kB 3.4 MB/s eta 0:00:00
Collecting bcrypt==4.3.0
  Downloading bcrypt-4.3.0-cp39-abi3-manylinux_2_34_x86_64.whl (284 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 284.2/284.2 kB 16.4 MB/s eta 0:00:00
Collecting cryptography==44.0.2
  Downloading cryptography-44.0.2-cp39-abi3-manylinux_2_34_x86_64.whl (4.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.2/4.2 MB 79.9 MB/s eta 0:00:00
Collecting hyperlink==21.0.0
  Downloading hyperlink-21.0.0-py2.py3-none-any.whl (74 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 74.6/74.6 kB 18.6 MB/s eta 0:00:00
Collecting idna==3.10
  Downloading idna-3.10-py3-none-any.whl (70 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 70.4/70.4 kB 13.3 MB/s eta 0:00:00
Collecting packaging==25.0
  Downloading packaging-25.0-py3-none-any.whl (66 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 66.5/66.5 kB 15.2 MB/s eta 0:00:00
Collecting pyasn1_modules==0.4.2
  Downloading pyasn1_modules-0.4.2-py3-none-any.whl (181 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 181.3/181.3 kB 32.3 MB/s eta 0:00:00
Collecting requests==2.32.3
  Downloading requests-2.32.3-py3-none-any.whl (64 kB)
```

8. Configure Cowrie :

Edit config to set port 2222: nano etc/cowrie.cfg

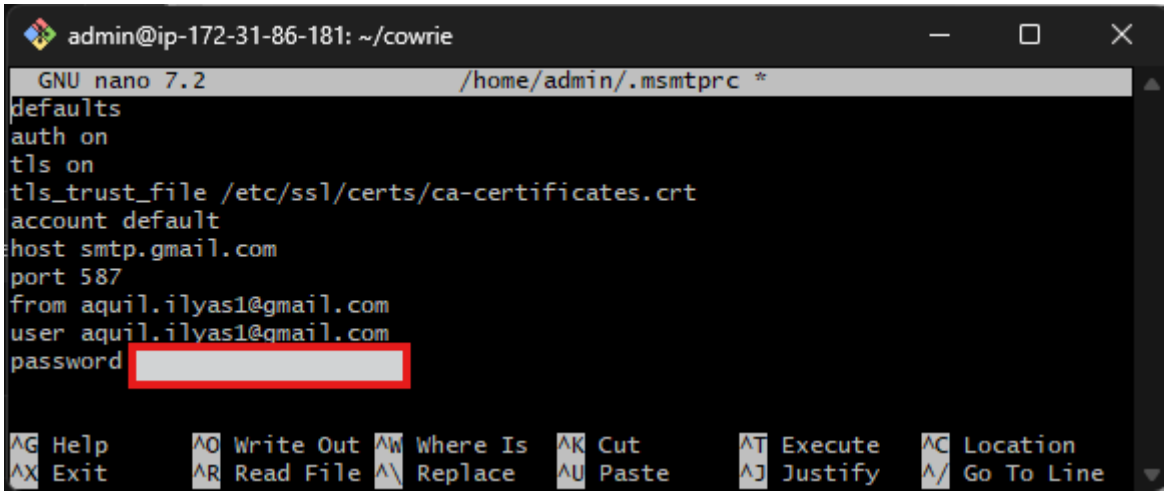


9. Set Up Email Alerts (via msmtp) :

To set up email alerts, I used msmtp, a simple tool for sending emails from the command line. I created a configuration file using this command:

```
nano ~/.msmtprc
```

Inside the file, I added the following settings to connect to Gmail's SMTP server:



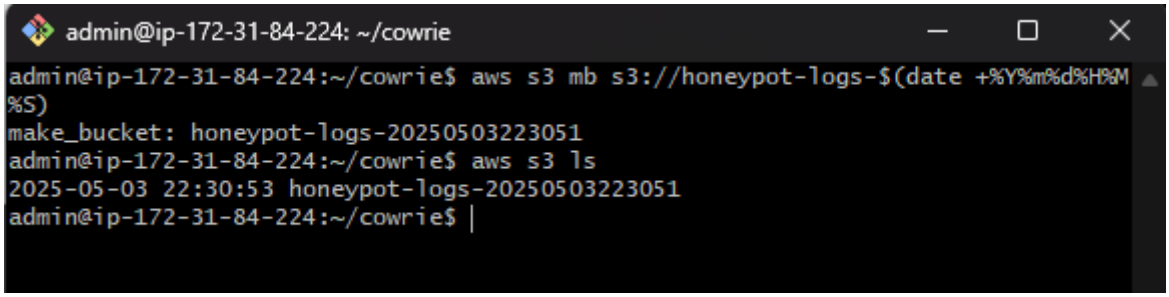
Then, I set the correct permissions for the file to keep it secure:

```
chmod 600 ~/.msmtprc
```

This setup allows the server to send email alerts through my Gmail account.

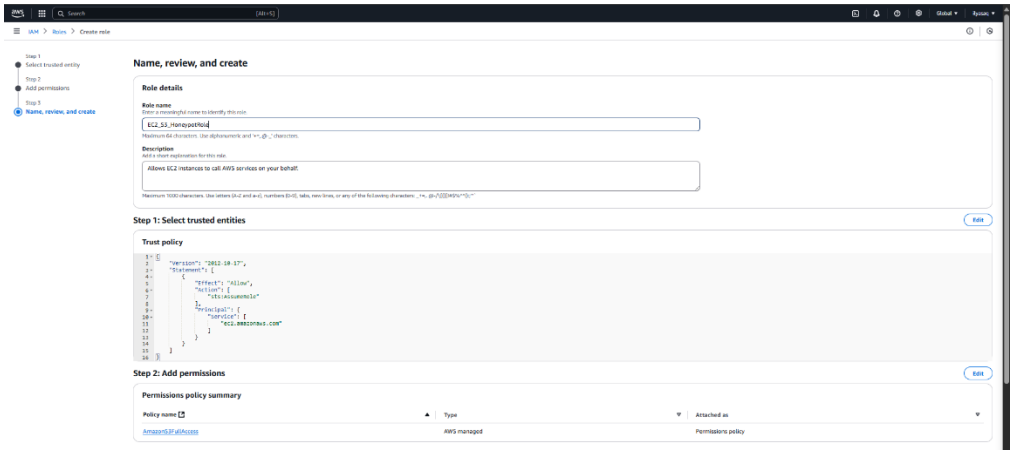
10. Create an S3 Bucket to store logs :

To store logs and other data from the Cowrie honeypot, I created an S3 bucket on AWS.

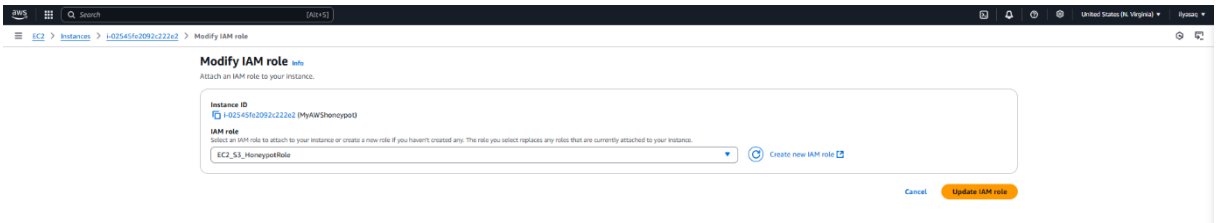


11. Add IAM Role to EC2 Instance (For S3 Access):

I created an IAM role in the AWS IAM console, selecting EC2 as the trusted entity and attaching the AmazonS3FullAccess policy. The role was named EC2_S3 _HoneypoteRole



I assigned the CowrieEC2Role IAM role to the EC2 instance by navigating to Actions > Security > Modify IAM Role in the EC2 console and selecting the newly created role.



12. Create the Monitoring Script :

Create a file called cowrie_email_alerts.sh:

```
nano ~/cowrie/cowrie_email_alerts.sh
```

Here's a sample script to capture login credentials, commands, send email alerts, and upload logs to S3, including details such as username, password, attacker IP, location, and executed commands:

```
#!/bin/bash

# Set the correct log file path

LOG_FILE="/home/admin/cowrie/var/log/cowrie/cowrie.log"

EMAIL="aquil.ilyas1@gmail.com"

S3_BUCKET="honeypot-logs-20250503223051"  #

# Temporary variables to store session data

SESSION_ID=""

USERNAME=""

PASSWORD=""

COMMANDS=()

ATTACKER_IP=""

CITY=""

COUNTRY=""

# Function to send email with login details

send_login_email() {

    local username="$1"

    local password="$2"

    EMAIL_CONTENT="Username: $username\nPassword: $password\nAttacker IP: $ATTACKER_IP\nLocation: $CITY,
$COUNTRY"

    echo -e "$EMAIL_CONTENT" | mail -s "Cowrie Attack - Login Credentials" "$EMAIL"

}

# Function to send email with collected commands after the session ends

send_commands_email() {

    local username="$1"

    local password="$2"

    local commands="$3"

    EMAIL_CONTENT="Username: $username\nPassword: $password\nAttacker IP: $ATTACKER_IP\nLocation: $CITY,
$COUNTRY\nCommands:\n$commands"

    echo -e "$EMAIL_CONTENT" | mail -s "Cowrie Attack - Executed Commands" "$EMAIL"

}

# Function to save session data to S3

save_to_s3() {

    local commands="$1"

    local file_name="cowrie-session-${SESSION_ID}.json"
```

```
    echo "{
  \"session_id\": \"$SESSION_ID\",
  \"ip\": \"$ATTACKER_IP\",
  \"city\": \"$CITY\",
  \"country\": \"$COUNTRY\",
  \"username\": \"$USERNAME\",
  \"password\": \"$PASSWORD\",
  \"commands\": \"$commands\"
}" > "$file_name"
    aws s3 cp "$file_name" "s3://$S3_BUCKET/logs/" --quiet
    rm "$file_name"
}
# Monitor Cowrie log file
tail -n 0 -f "$LOG_FILE" | while read -r line
do
    # Extract IP address
    if [[ "$line" =~ New\ connection:\ ([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+) ]]; then
        ATTACKER_IP="${BASH_REMATCH[1]}"
        SESSION_ID=$(date +%s)
        # Geo IP lookup
        GEO_INFO=$(curl -s http://ip-api.com/json/$ATTACKER_IP)
        CITY=$(echo "$GEO_INFO" | jq -r '.city')
        COUNTRY=$(echo "$GEO_INFO" | jq -r '.country')
    fi
    # Detect login attempt
    if [[ "$line" =~ login\ attempt\ \[b\'([^\']+)\'/b\'([^\']+)\' ]]; then
        USERNAME="${BASH_REMATCH[1]}"
        PASSWORD="${BASH_REMATCH[2]}"
        send_login_email "$USERNAME" "$PASSWORD"
        COMMANDS=()
    fi
    # Detect command execution
    if [[ "$line" =~ CMD:\ (.*) ]]; then
```

```
        CMD="${BASH_REMATCH[1]}"
        COMMANDS+=("$CMD")
    fi
    # End session on "exit"
    if [[ "$line" =~ "exit" ]]; then
        if [[ -n "$SESSION_ID" && -n "$USERNAME" && -n "$PASSWORD" && ${#COMMANDS[@]} -gt 0 ]]; then
            CMD_LIST=$(printf "%s\n" "${COMMANDS[@]}")
            send_commands_email "$USERNAME" "$PASSWORD" "$CMD_LIST"
            save_to_s3 "$CMD_LIST"
        fi
        # Reset session data
        SESSION_ID=""
        USERNAME=""
        PASSWORD=""
        COMMANDS=()
        ATTACKER_IP=""
        CITY=""
        COUNTRY=""
    fi
done
```

Make the script executable: `chmod +x ~/cowrie/cowrie_email_alerts.sh`

13. Start Cowrie :

To start the Cowrie honeypot, I navigated to the Cowrie directory and ran the following command:

```
cd ~/cowrie
bin/cowrie start
```

14. Run the Script :

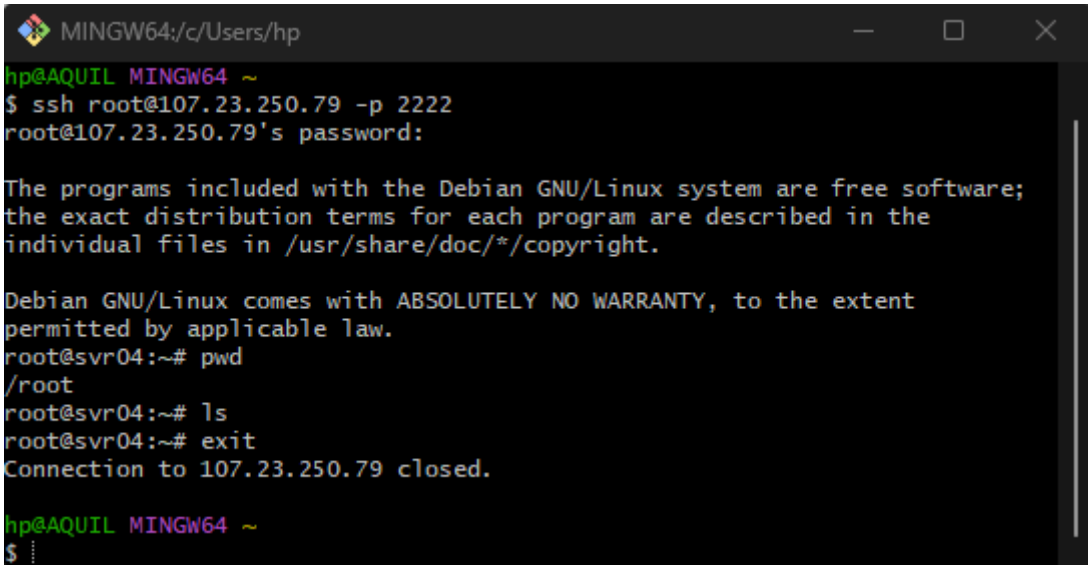
```
./cowrie_email_alerts.sh
```

This script continuously monitors the Cowrie log file, sends an email alert when a new login is detected, and uploads the log to the S3 bucket.

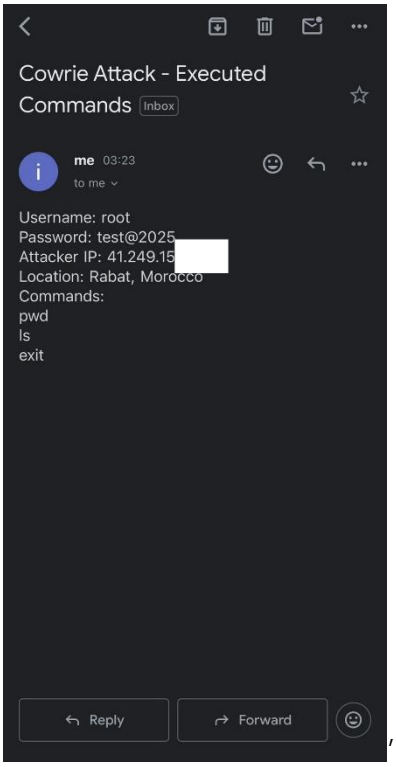
15. Test the Setup :

To test the Cowrie honeypot, I SSH'd into the EC2 instance from a remote machine using the following command:

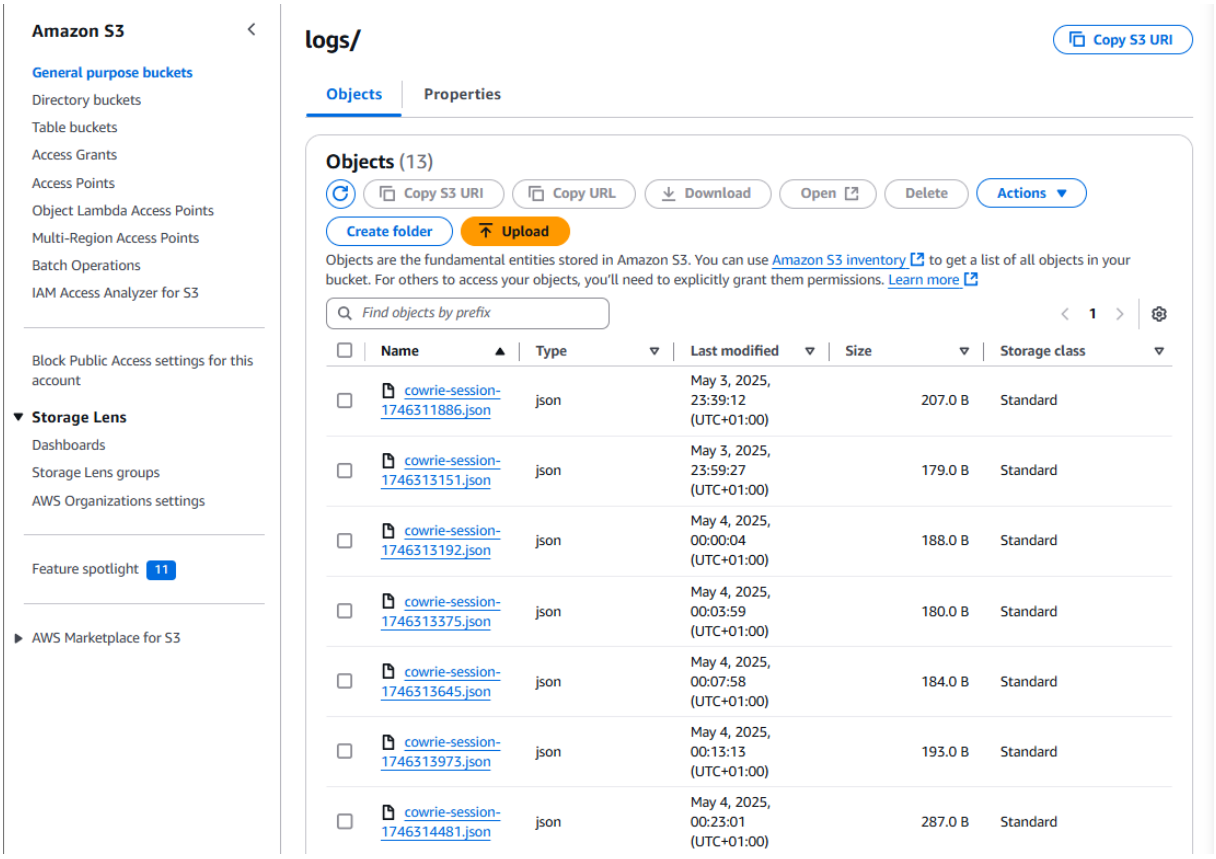
```
ssh -p 2222 root@<your-ec2-public-ip>
```



After executing some commands to trigger the honeypot, I verified that the system sent the login credentials and executed commands to my email :



I also checked the S3 bucket to ensure the logs were uploaded correctly :




```
Pretty-print
{
  "session_id": "1746325412",
  "ip": "41.249.15[redacted]",
  "city": "Rabat",
  "country": "Morocco",
  "username": "root",
  "password": "test@2025",
  "commands": "pwd
ls
exit"
}
```

Conclusion

The deployment of a Cowrie honeypot on AWS has provided valuable insights into the behavior of malicious actors and the types of attacks they typically employ. By utilizing AWS's robust infrastructure, we have been able to seamlessly scale and manage the honeypot, ensuring its high availability and security. The integration of attacker IP geolocation and cloud storage solutions has further enhanced the project's capabilities, enabling effective data analysis and storage. Overall, this honeypot serves as both a learning tool and a practical resource for enhancing cybersecurity defenses, with the potential to detect new attack vectors and gather data to improve future defensive strategies.