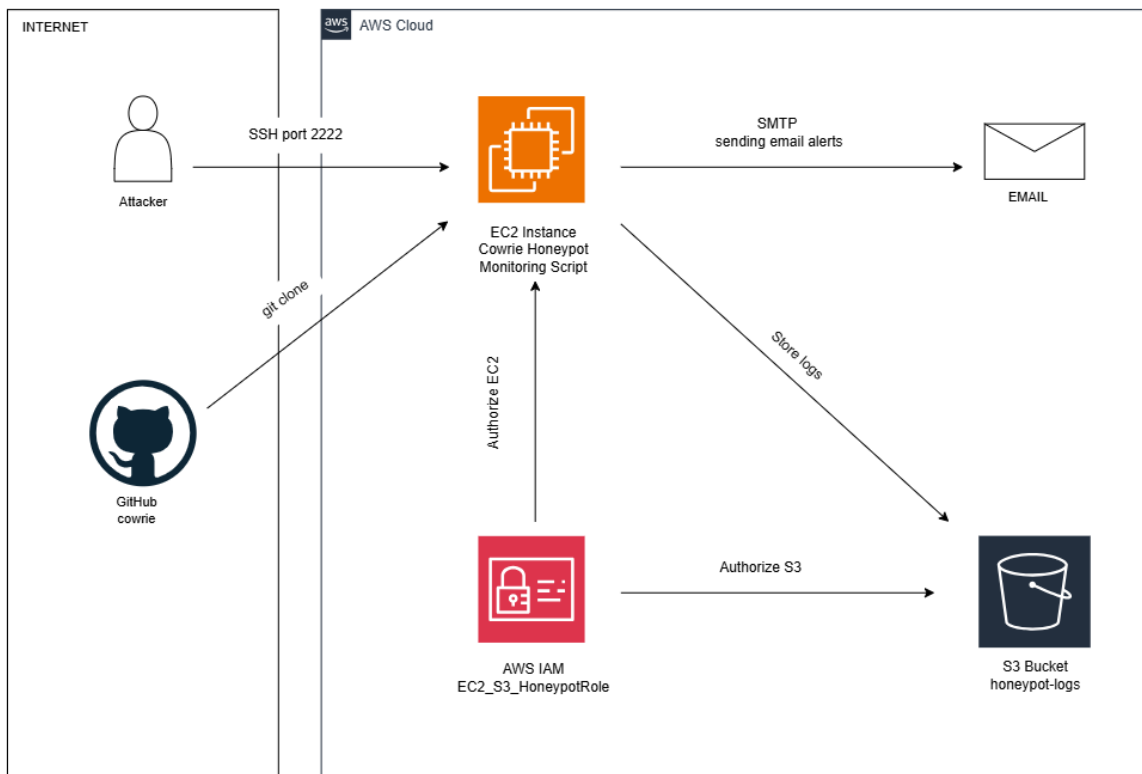


HONEYPOT DEPLOYMENT ON AWS



AQUIL ILYAS

AWS Honeypot Architecture
(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.

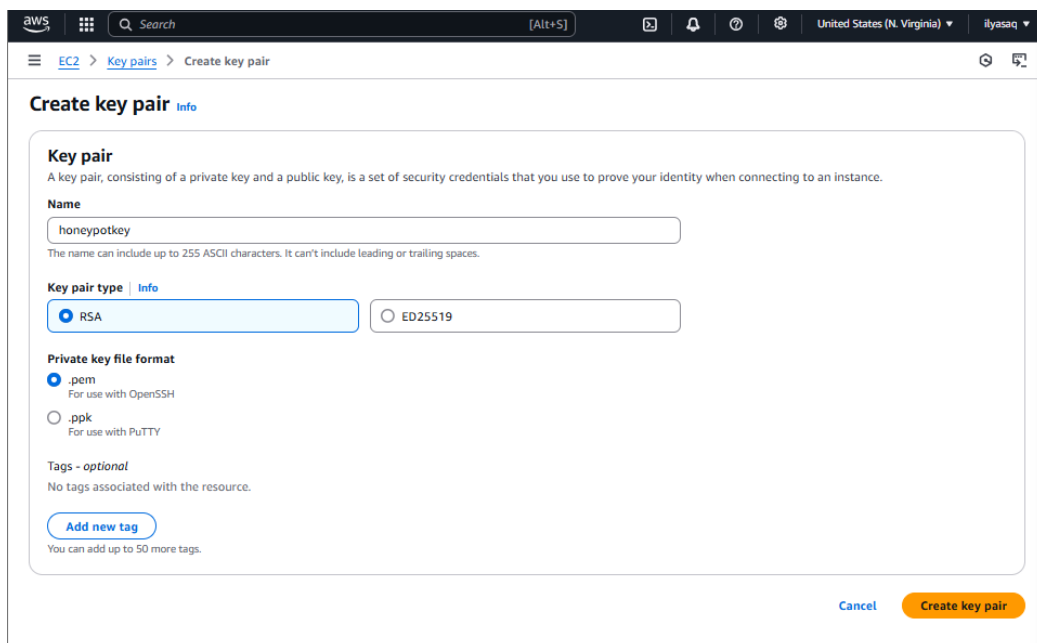
Deployment Steps

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."

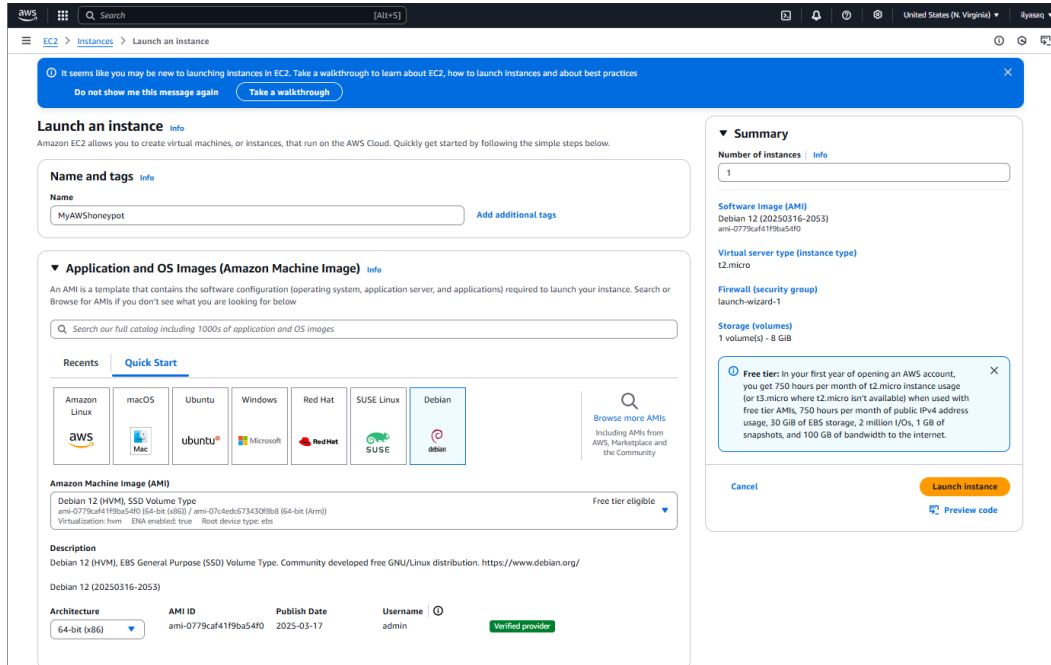
The file downloaded automatically to my browser. I'll use it later to access the instance.



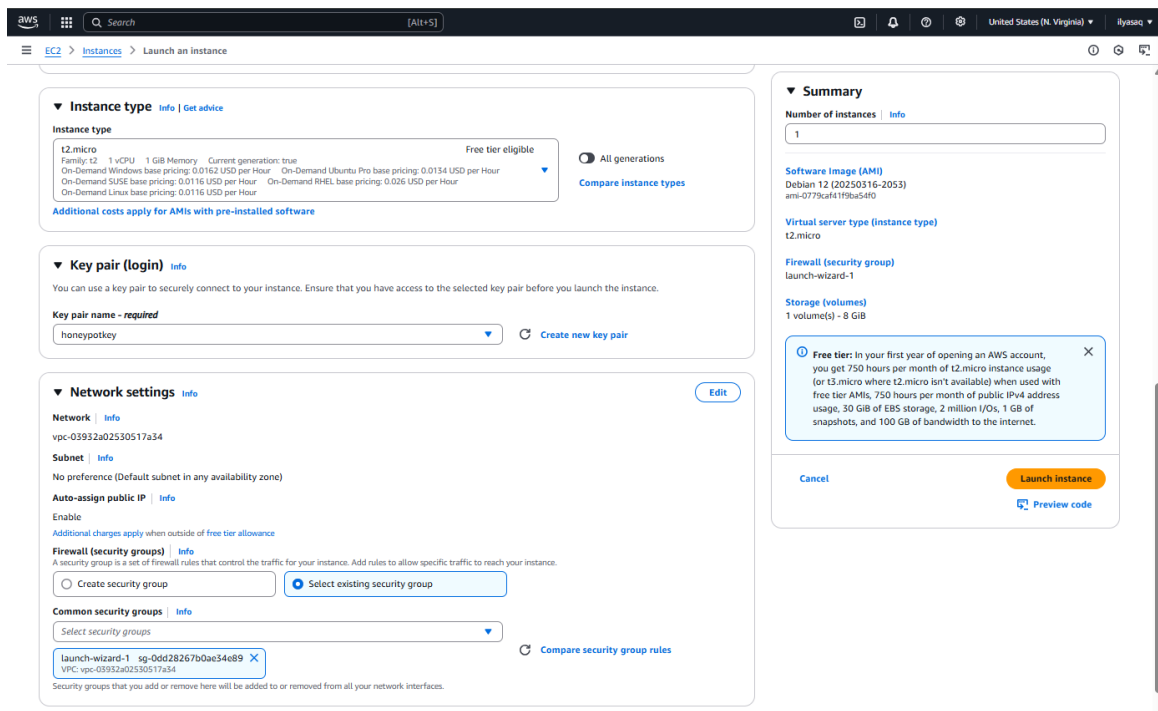
The screenshot shows the AWS Management Console interface for creating a new key pair. The breadcrumb navigation at the top indicates the path: EC2 > Key pairs > Create key pair. The main heading is "Create key pair" with an "Info" link. Below this, a descriptive text states: "A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance." The form includes a "Name" field with the value "honeypotkey" and a note: "The name can include up to 255 ASCII characters. It can't include leading or trailing spaces." The "Key pair type" section has two radio buttons: "RSA" (selected) and "ED25519". The "Private key file format" section has two radio buttons: ".pem" (selected, with the note "For use with OpenSSH") and ".ppk" (with the note "For use with PuTTY"). There is a "Tags - optional" section with the text "No tags associated with the resource." and an "Add new tag" button, with a note below: "You can add up to 50 more tags." At the bottom right, there are two buttons: "Cancel" and "Create key pair".

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.



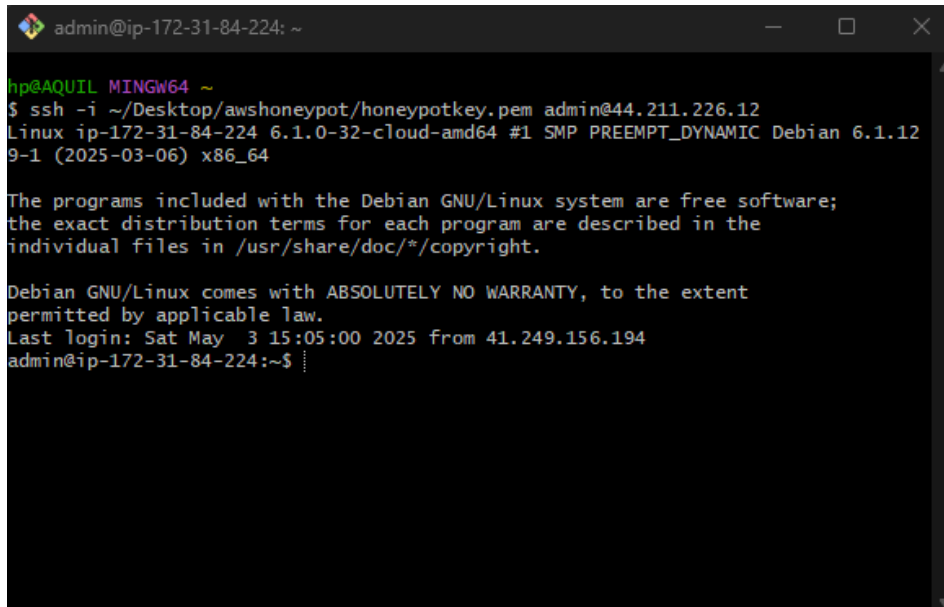
During the setup process, I chose the desired configuration for my instance, including the Amazon Machine Image (AMI), instance type, key pair, storage, and security group settings.



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>
```

A terminal window titled 'admin@ip-172-31-84-224: ~' showing the execution of an SSH command. The command is '\$ ssh -i ~/Desktop/awshoneypot/honeypotkey.pem admin@44.211.226.12'. The output shows the connection to a Linux instance with IP 172-31-84-224, kernel 6.1.0-32-cloud-amd64, and Debian 6.1.12. It includes the standard Debian GNU/Linux welcome message and the last login timestamp: 'Last login: Sat May 3 15:05:00 2025 from 41.249.156.194'. The prompt returns to 'admin@ip-172-31-84-224:~\$'.

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

```
chmod 400 honeypotkey.pem
```

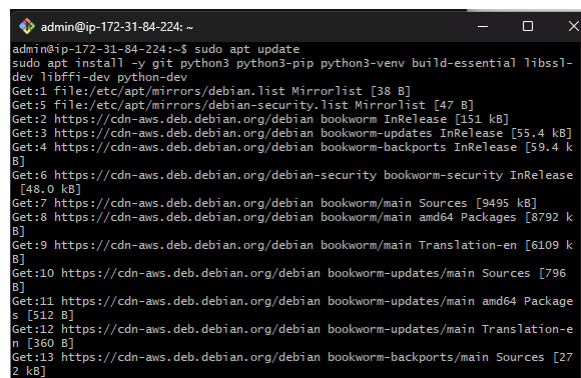
Then I replaced the file path and IP address with the correct ones for my instance. This let me log in to the server from my computer.

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
```

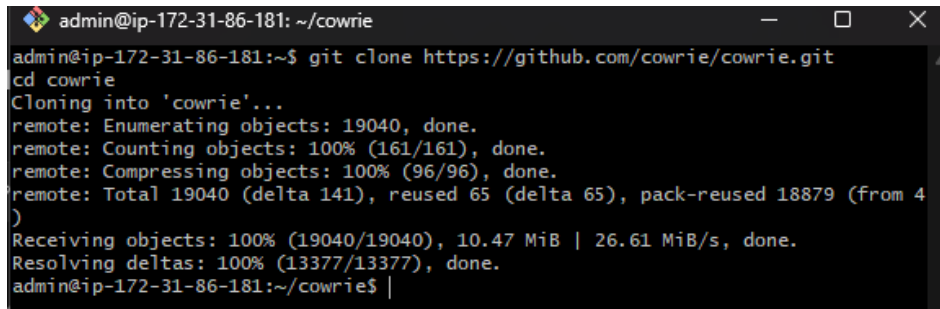
A terminal window titled 'admin@ip-172-31-84-224: ~' showing the execution of 'sudo apt update' followed by 'sudo apt install -y git python3 python3-pip python3-venv build-essential libssl-dev'. The output shows the system being updated and then various packages being installed, including git, python3, python3-pip, python3-venv, build-essential, and libssl-dev. The terminal shows progress bars and download sizes for each package.

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:

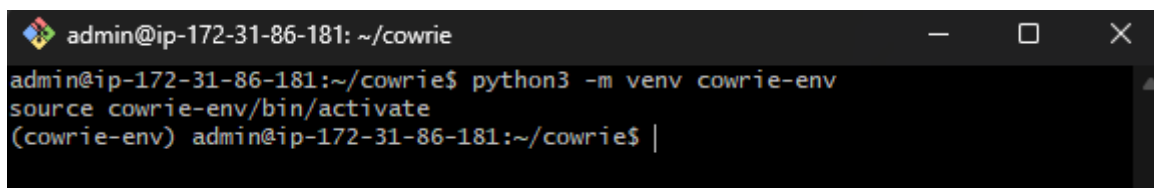
```
git clone https://github.com/cowrie/cowrie.git
cd cowrie
```



```
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$ |
```

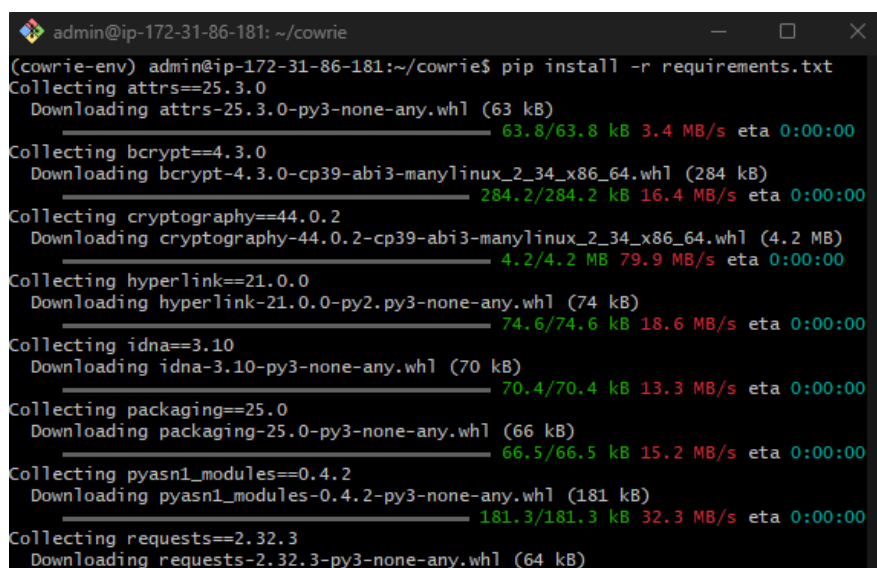
This downloaded the Cowrie honeypot project and moved me into its folder to begin setup.

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$ |
```

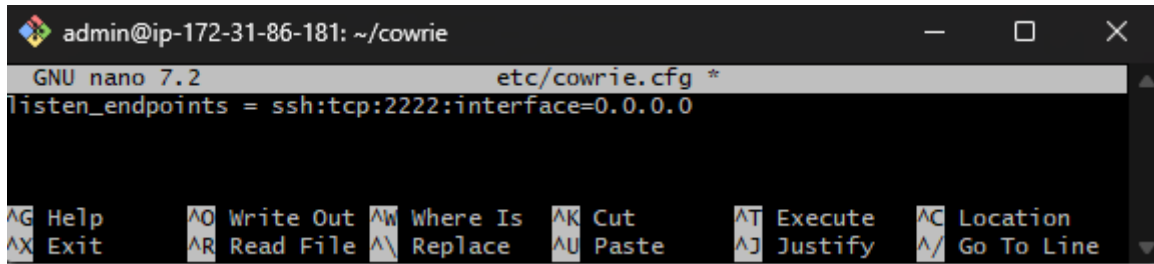
7. Install Python Dependencies:



```
(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`



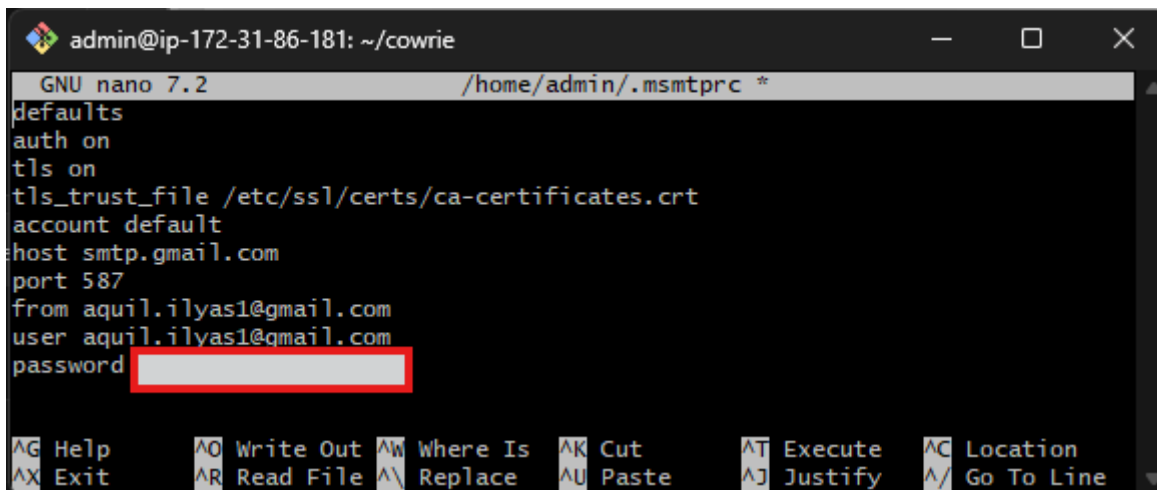
```
admin@ip-172-31-86-181: ~/cowrie
GNU nano 7.2      etc/cowrie.cfg *
listen_endpoints = ssh:tcp:2222:interface=0.0.0.0
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify    ^_/ Go To Line
```

9. Set Up Email Alerts (via msmtprc) :

To set up email alerts, I used msmtprc, 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:



```
admin@ip-172-31-86-181: ~/cowrie
GNU nano 7.2      /home/admin/.msmtprc *
defaults
auth on
tls on
tls_trust_file /etc/ssl/certs/ca-certificates.crt
account default
host smtp.gmail.com
port 587
from aquil.ilyas1@gmail.com
user aquil.ilyas1@gmail.com
password
```

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.

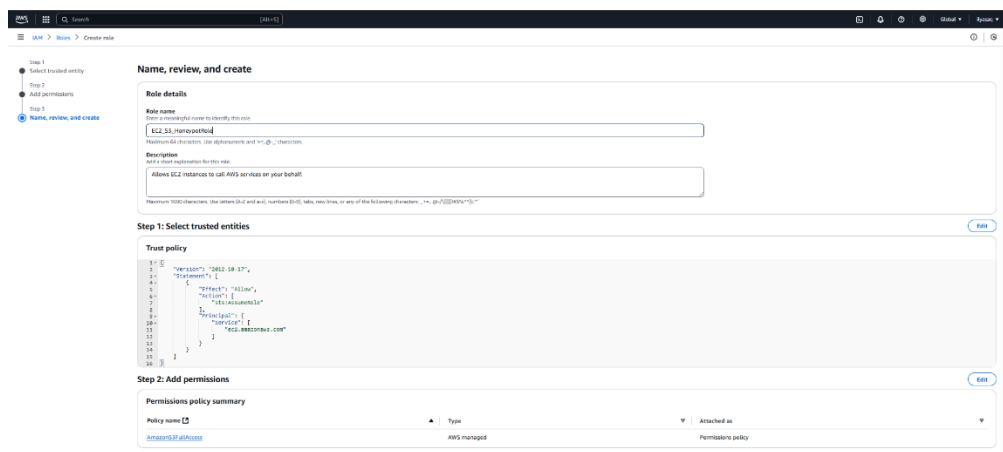
10. Create an S3 Bucket to store logs :

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

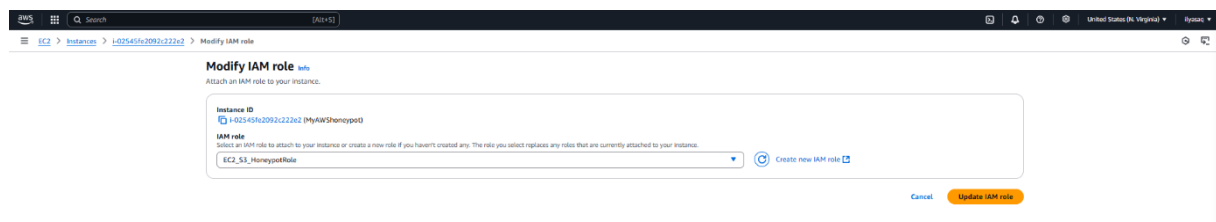
```
admin@ip-172-31-84-224: ~/cowrie
admin@ip-172-31-84-224:~/cowrie$ aws s3 mb s3://honeypot-logs-$(date +%Y%m%d%H%M%S)
make_bucket: honeypot-logs-20250503223051
admin@ip-172-31-84-224:~/cowrie$ aws s3 ls
2025-05-03 22:30:53 honeypot-logs-20250503223051
admin@ip-172-31-84-224:~/cowrie$ |
```

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\([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 :

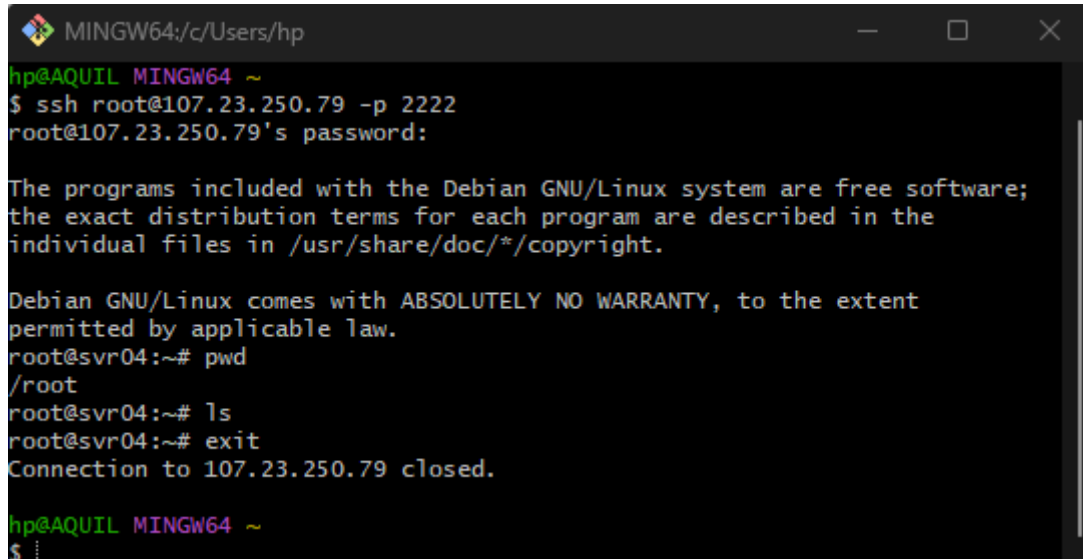
```
./cowrie_email_alerts.sh
```

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>
```



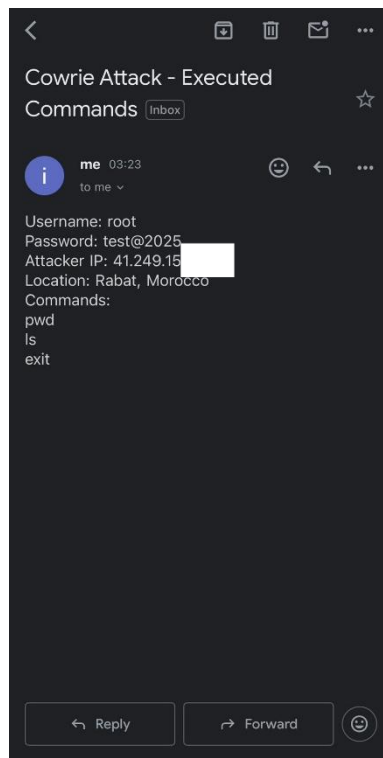
```
MINGW64/c/Users/hp
hp@AQUIL MINGW64 ~
$ ssh root@107.23.250.79 -p 2222
root@107.23.250.79's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@svr04:~# pwd
/root
root@svr04:~# ls
root@svr04:~# exit
Connection to 107.23.250.79 closed.

hp@AQUIL MINGW64 ~
$
```

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 :

Amazon S3

General purpose buckets

Directory buckets

Table buckets

Access Grants

Access Points

Object Lambda Access Points

Multi-Region Access Points

Batch Operations

IAM Access Analyzer for S3

Block Public Access settings for this account

Storage Lens

Dashboards

Storage Lens groups

AWS Organizations settings

Feature spotlight 11

AWS Marketplace for S3

logs/

Copy S3 URI

Objects

Properties

Objects (13)

Copy S3 URI

Copy URL

Download

Open

Delete

Actions

Create folder

Upload

Find objects by prefix

1

Find objects by prefix

	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	cowrie-session-1746311886.json	json	May 3, 2025, 23:39:12 (UTC+01:00)	207.0 B	Standard
<input type="checkbox"/>	cowrie-session-1746313151.json	json	May 3, 2025, 23:59:27 (UTC+01:00)	179.0 B	Standard
<input type="checkbox"/>	cowrie-session-1746313192.json	json	May 4, 2025, 00:00:04 (UTC+01:00)	188.0 B	Standard
<input type="checkbox"/>	cowrie-session-1746313375.json	json	May 4, 2025, 00:03:59 (UTC+01:00)	180.0 B	Standard
<input type="checkbox"/>	cowrie-session-1746313645.json	json	May 4, 2025, 00:07:58 (UTC+01:00)	184.0 B	Standard
<input type="checkbox"/>	cowrie-session-1746313973.json	json	May 4, 2025, 00:13:13 (UTC+01:00)	193.0 B	Standard
<input type="checkbox"/>	cowrie-session-1746314481.json	json	May 4, 2025, 00:23:01 (UTC+01:00)	287.0 B	Standard

```
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.