***Cowrie Honeypot Project***

**Author:** Farheena   
 **Date:** September 2025

[Introduction to Cowrie Honeypot 3](#_Toc928194359)

[Key Features of Cowrie: 3](#_Toc1438474289)

[Why Use Cowrie? 4](#_Toc592794107)

[System Preparation 4](#_Toc1618477246)

[System Update and Upgrade 4](#_Toc347410735)

[Installing Required Packages 4](#_Toc1243937759)

[Downloading and Preparing Cowrie in the /opt Directory 5](#_Toc348601886)

[5](#_Toc1272557330)

[Creating and Configuring the Python Virtual Environment for Cowrie 6](#_Toc1215937462)

[Editing the Cowrie Configuration File 7](#_Toc1718031239)

[Configuring the Cowrie Honeypot 8](#_Toc1229270014)

[Starting the Cowrie Honeypot 8](#_Toc1256730042)

[Monitoring Cowrie Logs in Real Time 9](#_Toc1549745061)

[Simulating Attacks on the Cowrie Honeypot 9](#_Toc1529470646)

[Gaining Access via SSH 10](#_Toc1907112655)

[SSH Access Attempt Logs 10](#_Toc1144989348)

[Preparing and Using Wordlists for Brute-Force Testing 11](#_Toc1851628234)

[Verifying Wordlist Availability 11](#_Toc781272077)

[Brute-Force Testing with Hydra 12](#_Toc242114659)

[Analyzing Captured Attack Logs 13](#_Toc86747303)

[Restarting Cowrie Honeypot 15](#_Toc167727200)

[Restarting Cowrie Honeypot Service with Proper User Permissions 15](#_Toc229096677)

[Hydra Brute-Force Attack on Telnet (Failed Attempt) 16](#_Toc1018543950)

[Brute Force SSH Attempt Using Hydra (Successful Attempt) 17](#_Toc1002394471)

[Reverse Shell Interaction Captured by Cowrie Honeypot 18](#_Toc5781008)

[Stopping the Cowrie Honeypot 19](#_Toc946853855)

[Real-Time Honeypot Terminal Output 19](#_Toc604482305)

[Conclusion 19](#_Toc1714897471)

# Introduction to Cowrie Honeypot

A honeypot is a security mechanism designed to attract and monitor malicious activity by simulating a vulnerable system. It allows researchers, security professionals, and organizations to study attacker behavior in a controlled and isolated environment.

Cowrie is a popular medium-interaction SSH and Telnet honeypot. It is specifically built to emulate a real server by providing a fake command shell and filesystem that looks convincing to attackers. When an attacker tries to connect to the system using SSH or Telnet

Cowrie captures:

* Login attempts (usernames and passwords)
* Executed commands
* Files downloaded or uploaded
* The attacker’s behavior and techniques

Cowrie does not give attackers access to a real system. Instead, it tricks them into interacting with a controlled environment where their actions are logged safely for analysis.

## Key Features of Cowrie:

* Emulates an SSH/Telnet server to attract brute-force and automated attacks.
* Records all authentication attempts with usernames and passwords.
* Logs attacker commands, including file downloads via wget or curl.
* Provides a fake filesystem that mimics a real Linux system.
* Can be integrated with external tools like ELK Stack (Elasticsearch, Logstash, Kibana) for visualization.

## Why Use Cowrie?

Deploying Cowrie allows cybersecurity students, researchers, and professionals to:

* Understand real-world attack patterns.
* Collect intelligence on brute-force tools and malware distribution.
* Enhance knowledge in intrusion detection and prevention.
* Improve network defense strategies through data-driven insights.

# System Preparation

Before installing Cowrie, the system must be updated and the necessary packages installed.

## System Update and Upgrade

sudo apt update && sudo apt upgrade -y  
apt update refreshes the list of available packages from repositories, while apt upgrade -y upgrades all installed software to their latest versions without asking for confirmation.

# Installing Required Packages

To install all the dependencies Cowrie needs, run:

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

Breakdown of the command:

* **git** → Used to download Cowrie’s source code from GitHub.
* **python3-venv** → Lets you create a Python virtual environment (isolated space for Cowrie).
* **python3-pip** → Python package manager (to install Python libraries for Cowrie).
* **libssl-dev** → Development library for SSL/TLS (needed for secure connections).
* **libffi-dev** → Library that helps with low-level system functions (some Python packages need this).
* **build-essential** → A package that includes tools like gcc, make, etc. (needed to compile and build software).
* **-y** → Automatically says “yes” to install.

# Downloading and Preparing Cowrie in the /opt Directory

It is common practice to install third-party applications inside the /opt directory. To download Cowrie and set up permissions, run the following commands:

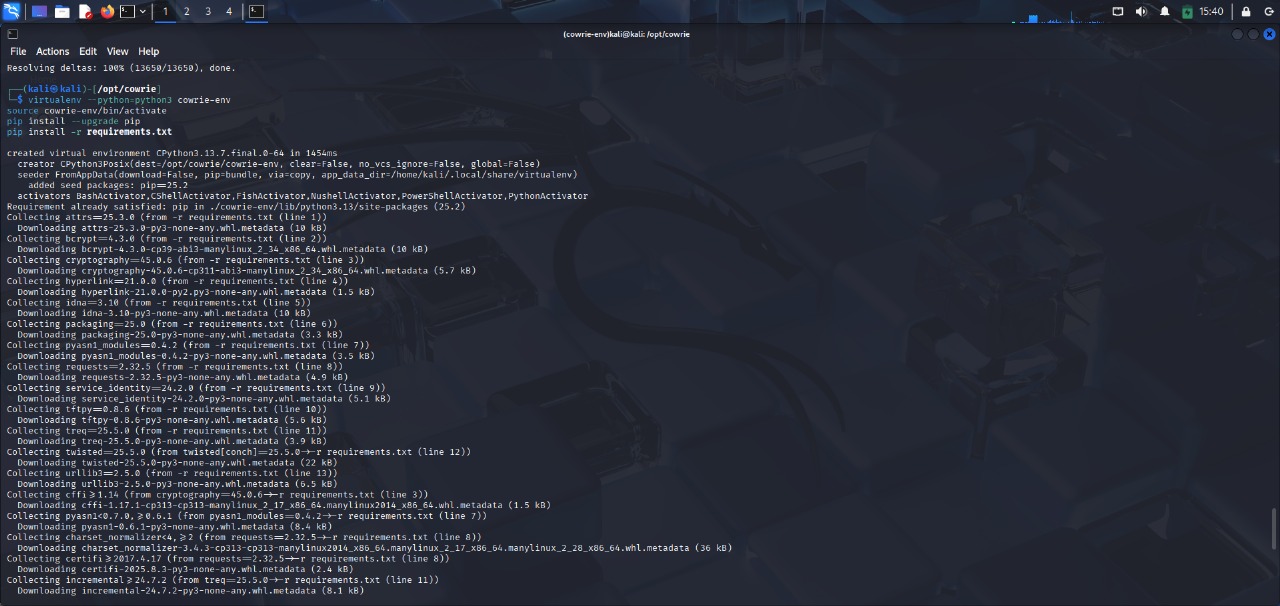
Breakdown of the commands:

* **cd /opt** → Moves into the /opt directory, where external applications are typically stored
* **sudo git clone** [**https://github.com/cowrie/cowrie.git**](https://github.com/cowrie/cowrie.git?utm_source=chatgpt.com)→ Downloads the Cowrie source code from GitHub into /opt/cowrie
* **sudo chown -R $USER:$USER cowrie** → Changes the ownership of the cowrie directory to your current user, ensuring you can modify its contents without needing root privileges

# Creating and Configuring the Python Virtual Environment for Cowrie

Cowrie runs inside a Python virtual environment to keep its dependencies isolated from the rest of the system.

Run the following commands inside the /opt/cowrie directory:

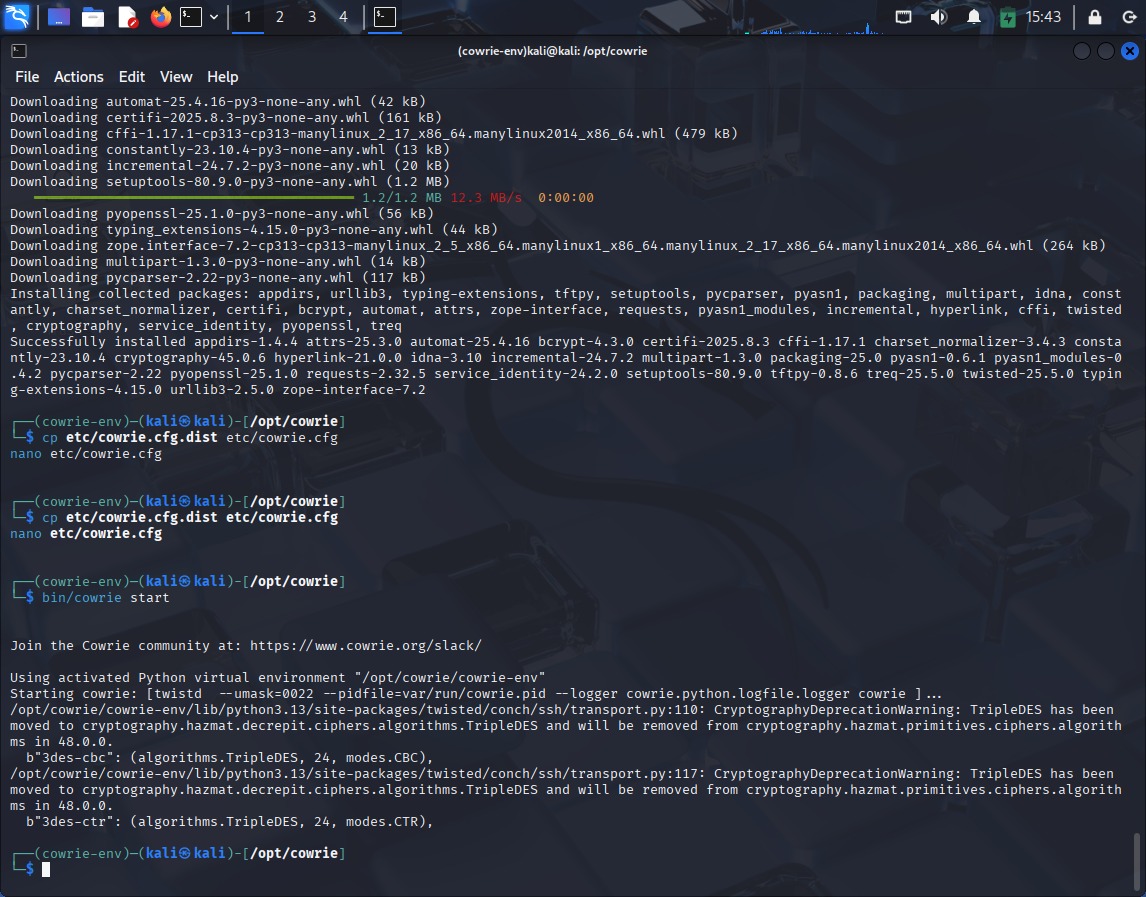
Breakdown of the commands:

* **virtualenv --python=python3 cowrie-env** → Creates a virtual environment named cowrie-env using Python 3
* **source cowrie-env/bin/activate** → Activates the virtual environment so Python and pip use this isolated environment
* **pip install --upgrade pip** → Updates pip to the latest version inside the virtual environment
* **pip install -r requirements.txt** → Installs all Python dependencies listed in Cowrie’s requirements.txt file

# Editing the Cowrie Configuration File

Before starting Cowrie, you need to set up its configuration file.

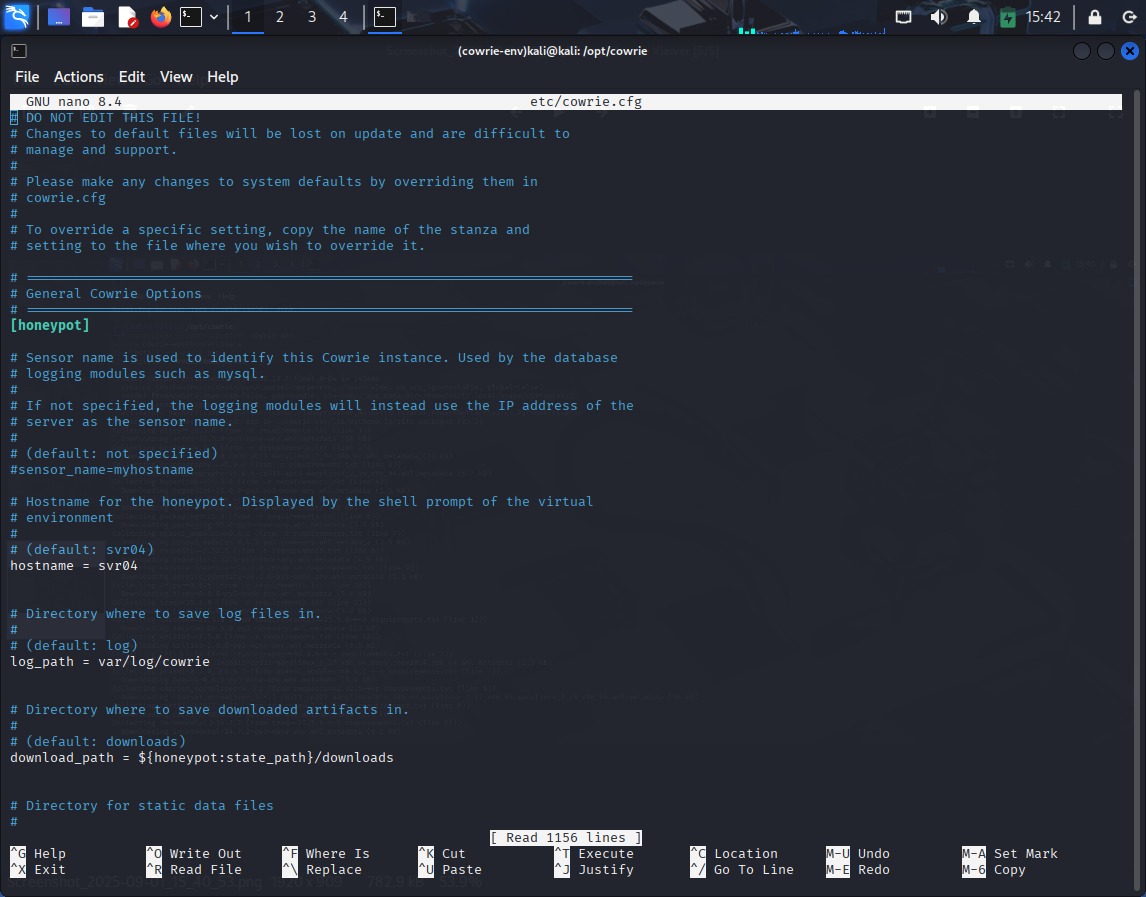
Run the following commands inside the /opt/cowrie directory:



Breakdown of the commands:

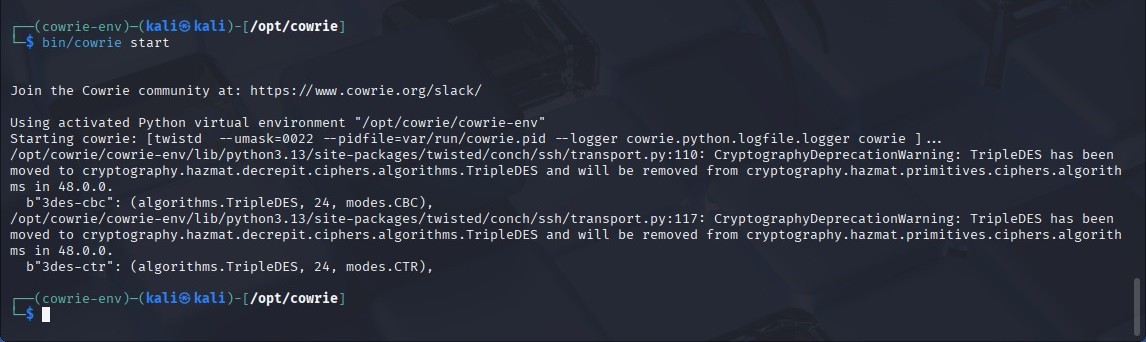
* **cp etc/cowrie.cfg.dist etc/cowrie.cfg** → Copies the sample configuration file (cowrie.cfg.dist) to a new file named cowrie.cfg. This is the file Cowrie will actually use.
* **nano etc/cowrie.cfg** → Opens the configuration file in the Nano text editor, allowing you to customize settings such as listening ports, logging options, and other parameters.

## Configuring the Cowrie Honeypot



# Starting the Cowrie Honeypot

After configuration, you can start the honeypot with:

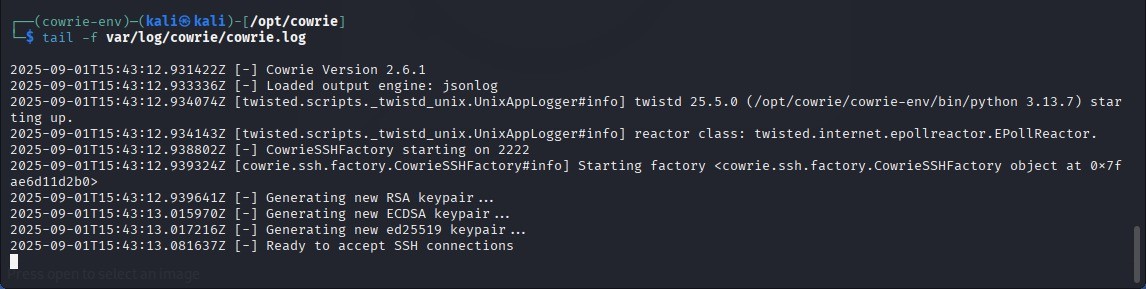
Explanation:

* **bin/cowrie** → The main script used to manage Cowrie.
* **start** → Launches the honeypot in the background, allowing it to begin listening for SSH and Telnet connections.

Once started, Cowrie will log all activity (login attempts, commands, file transfers, etc.) inside the var/log/cowrie/ directory.

# Monitoring Cowrie Logs in Real Time

To view Cowrie’s activity in real time, use the following command:

Explanation:

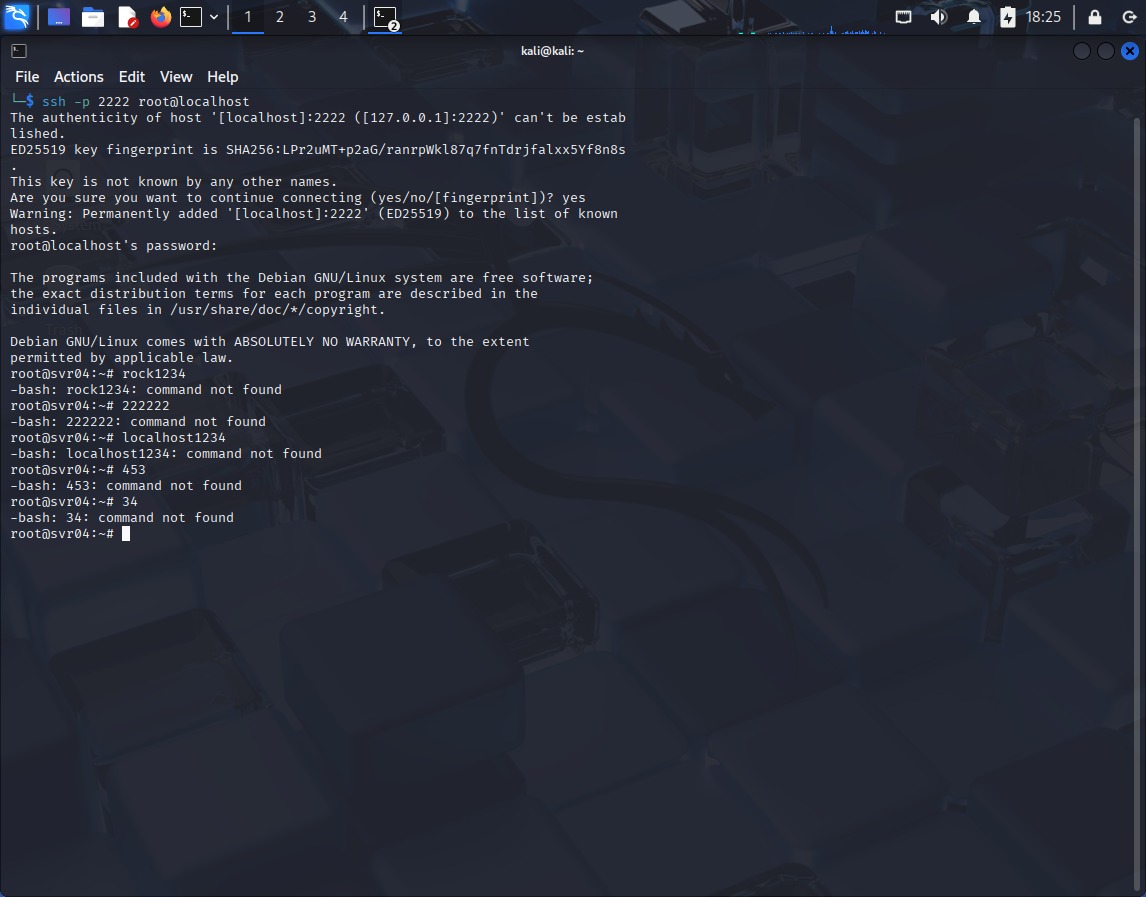
* **tail -f** → Displays the last lines of a file and continuously updates as new entries are added.
* **var/log/cowrie/cowrie.log** → The main log file where Cowrie records SSH/Telnet login attempts, executed commands, and attacker activity.

This allows you to monitor attacks as they happen, without needing to reopen the log file each time.

# Simulating Attacks on the Cowrie Honeypot

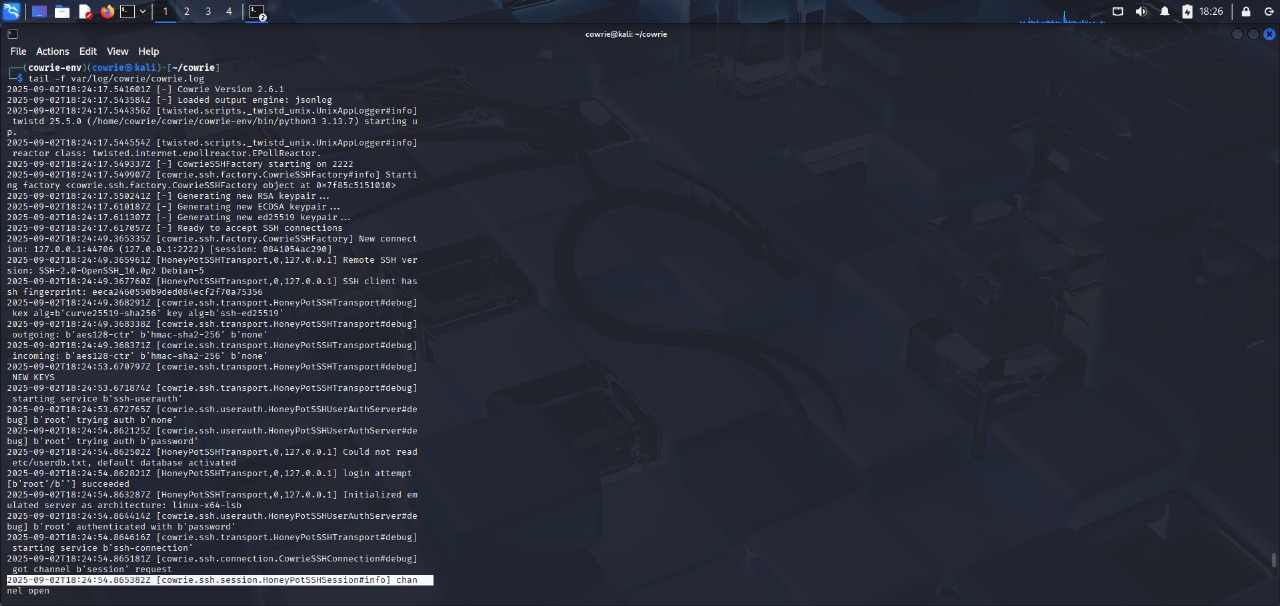
## Gaining Access via SSH

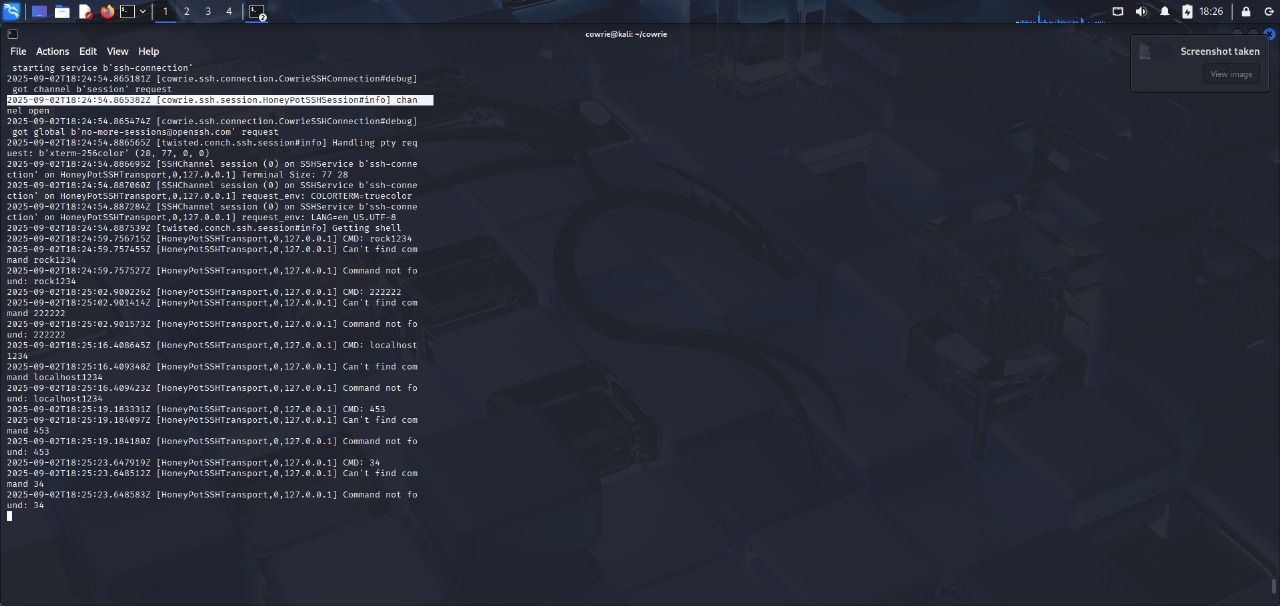
You can test Cowrie by trying to connect to it as if you were an attacker. Run the command:

Explanation:

* **ssh** → Secure Shell, used to connect to remote systems.
* **-p 2222** → Specifies the port number (Cowrie listens on port 2222 by default instead of the standard SSH port 22).
* [**root@127.0.0.1**](mailto:root@127.0.0.1)→ Attempts to log in as the user root on the local machine (127.0.0.1 = localhost).

## SSH Access Attempt Logs

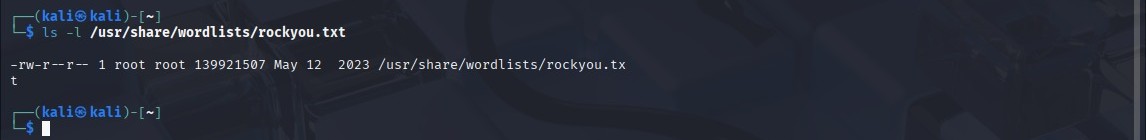




# Preparing and Using Wordlists for Brute-Force Testing

## Verifying Wordlist Availability

You can use wordlists to perform password guessing attacks or testing against Cowrie. To view a specific wordlist, run:

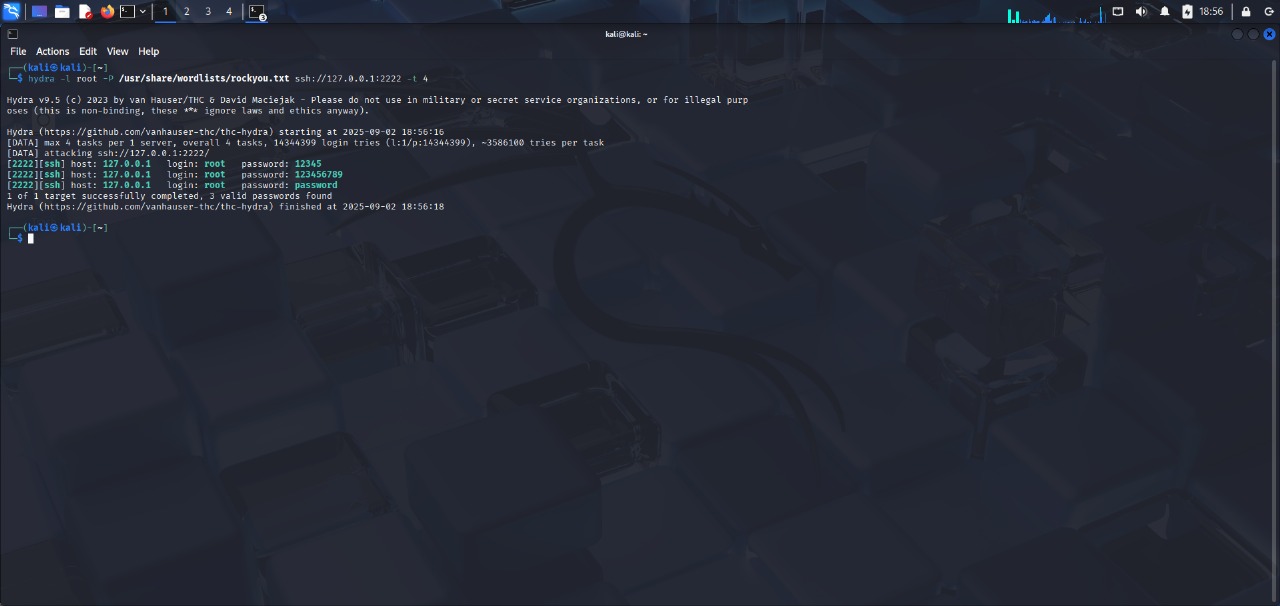
Explanation:

* **ls -l** → Lists the file in long format, showing details such as permissions, owner, size, and modification date.
* **/usr/share/wordlists/rockyou.txt** → A commonly used password wordlist included with many penetration testing tools (like Kali Linux).

This command helps confirm that the wordlist exists and is accessible before using it in tools like Hydra or other password-guessing utilities.

## Brute-Force Testing with Hydra

To test Cowrie against password-guessing attacks, use Hydra with a wordlist:

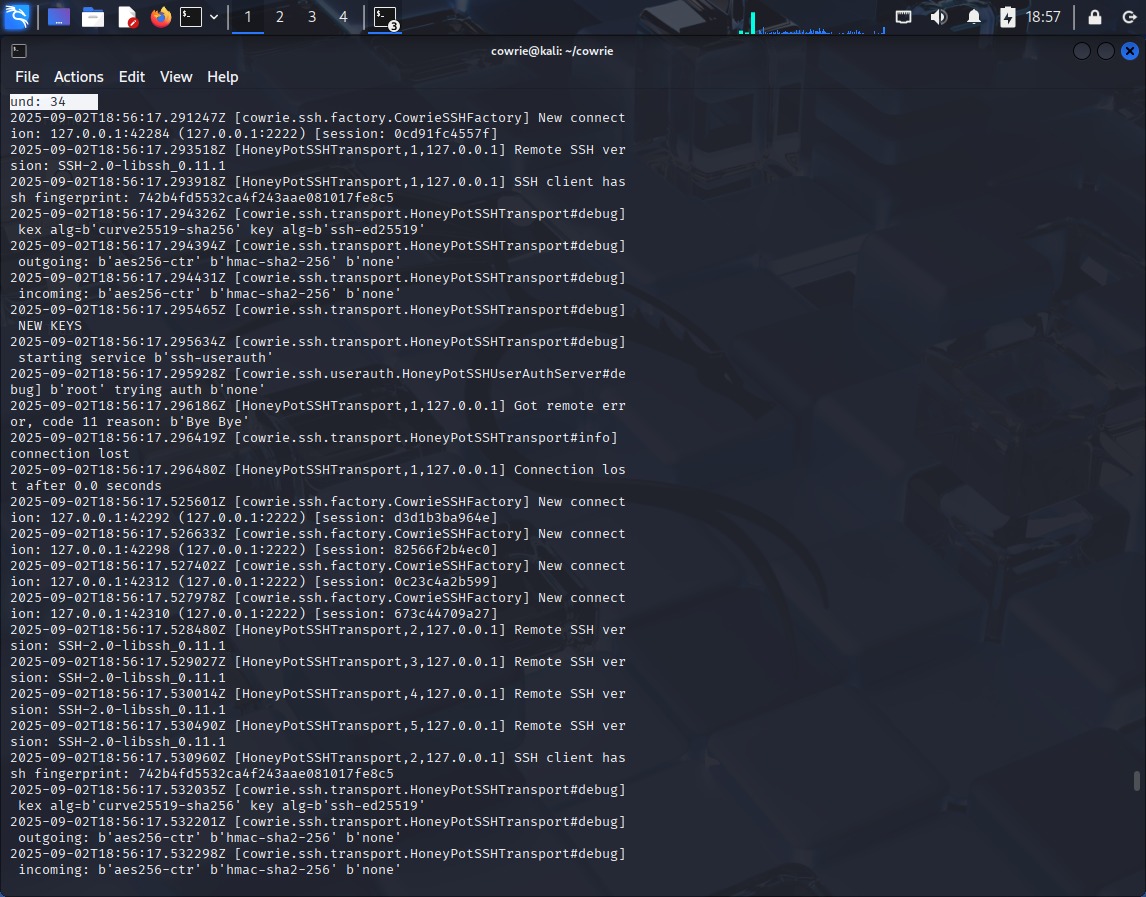


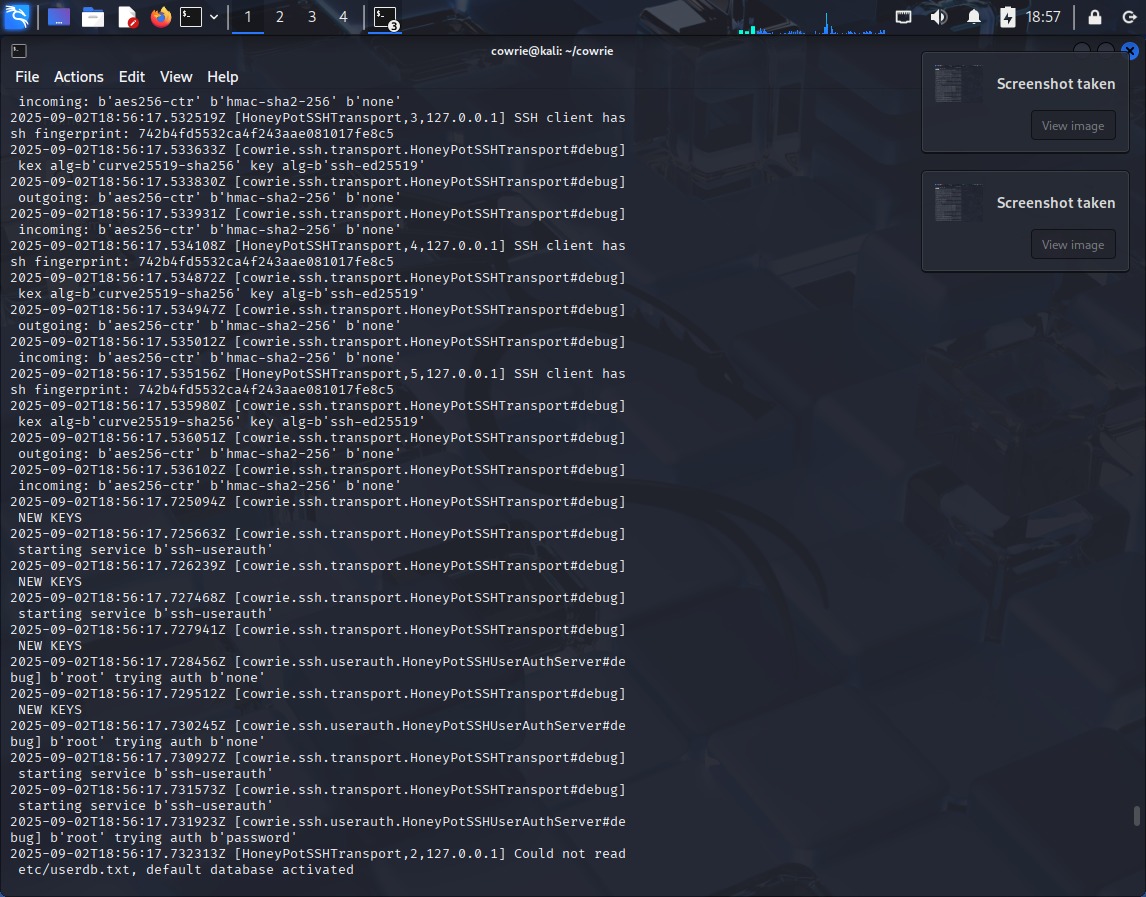
Explanation:

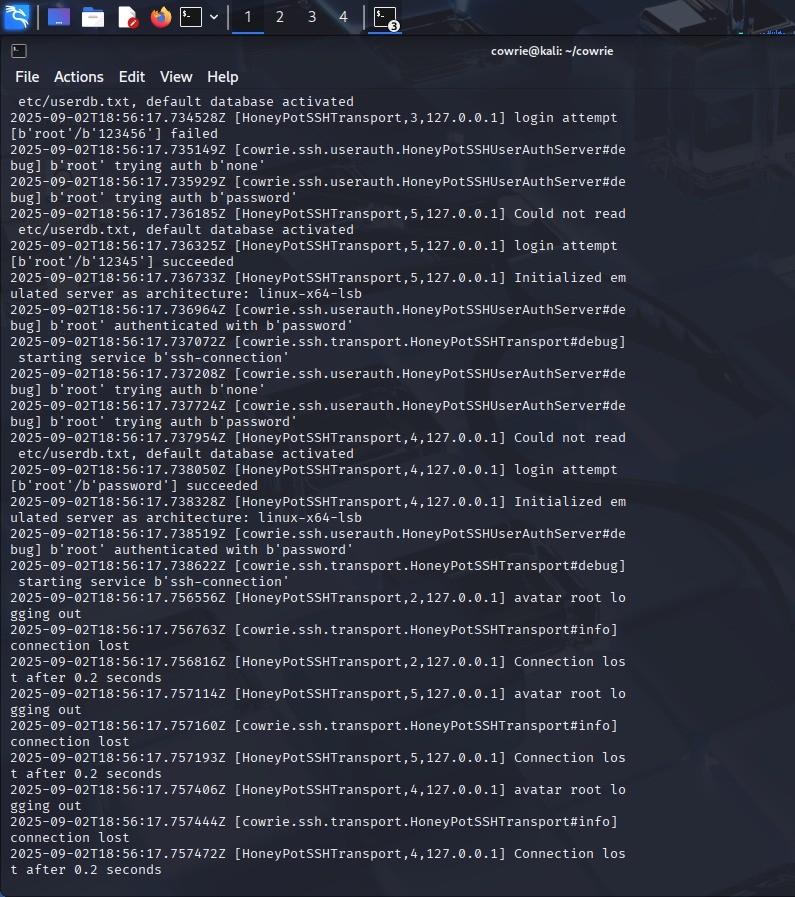
* **hydra** → A tool for performing brute-force attacks on login services.
* **-l root** → Specifies the username to test (root in this case).
* **-P /usr/share/wordlists/rockyou.txt** → Path to the password wordlist to use (-P is for a file of passwords).
* **ssh://127.0.0.1:2222** → Target service and port (Cowrie listens on port 2222).
* **-t 4** → Number of parallel tasks (threads) to speed up the attack.

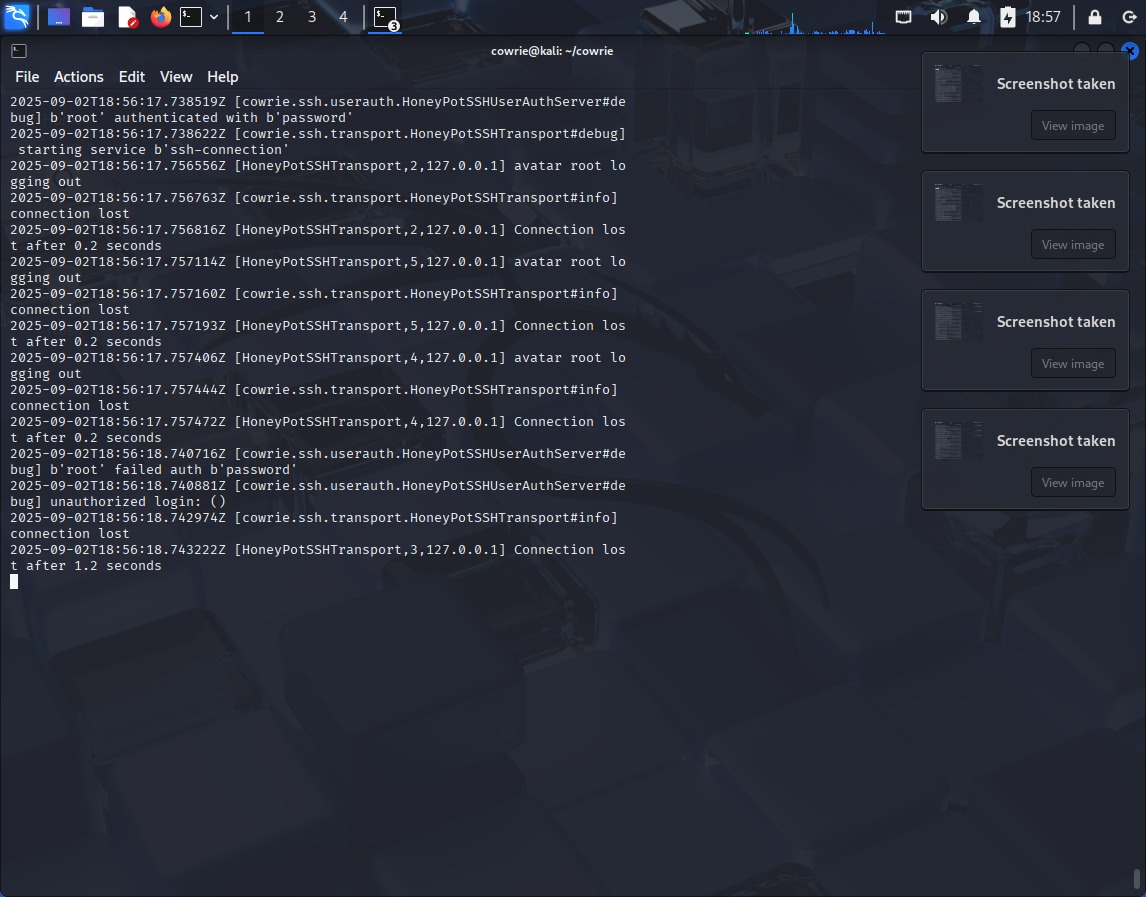
Cowrie will log all login attempts, even if the attack is unsuccessful, providing insight into attacker behavior for analysis.

## Analyzing Captured Attack Logs









# Restarting Cowrie Honeypot

Sometimes the honeypot terminal may close accidentally. Restarting Cowrie ensures it continues capturing SSH and other activity logs for monitoring and analysis. In a lab environment, resetting the honeypot is normal and useful for practice.

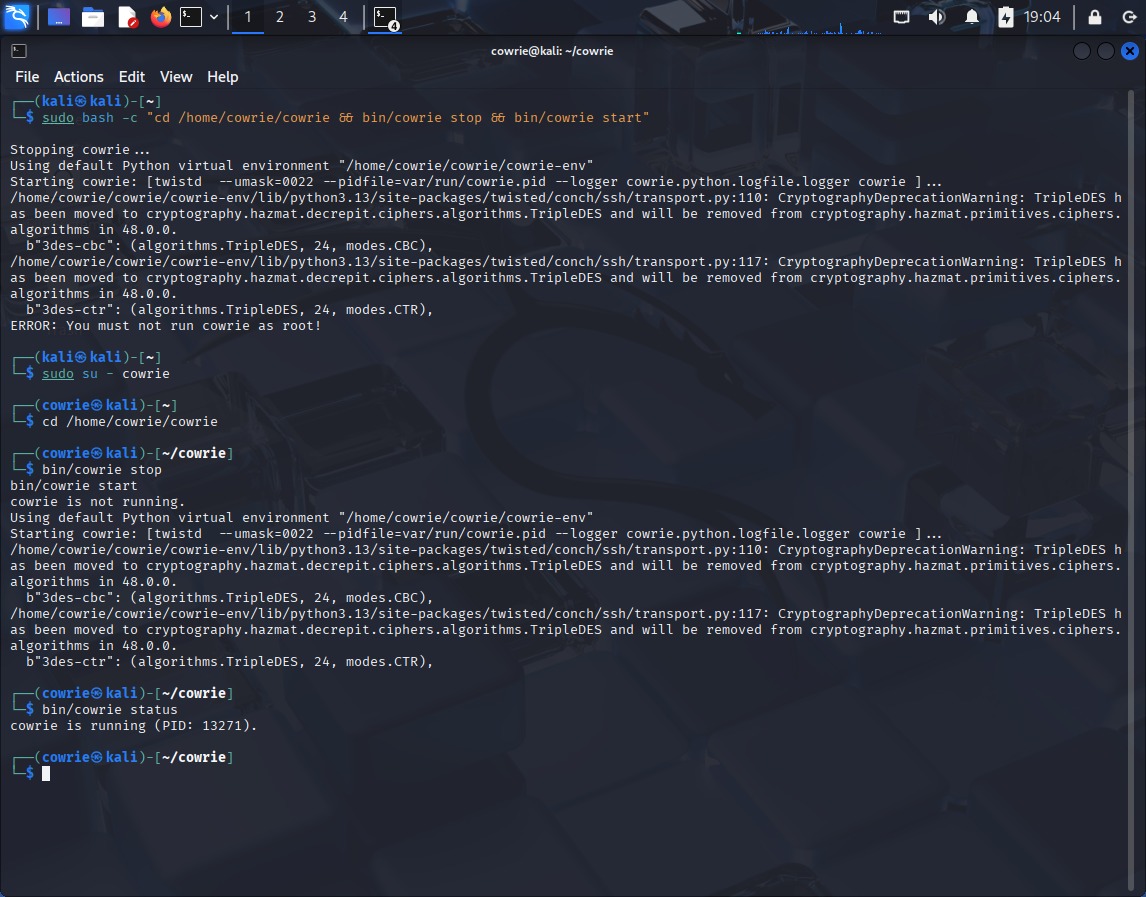
## Restarting Cowrie Honeypot Service with Proper User Permissions

Cowrie must be run under its dedicated user (cowrie) rather than as root. This restriction prevents potential damage if the honeypot is compromised.

Initially, starting Cowrie as root produces the error:

*You must not run cowrie as root!*

To correct this,I switch to the cowrie account, navigates to the installation directory, and starts the honeypot.

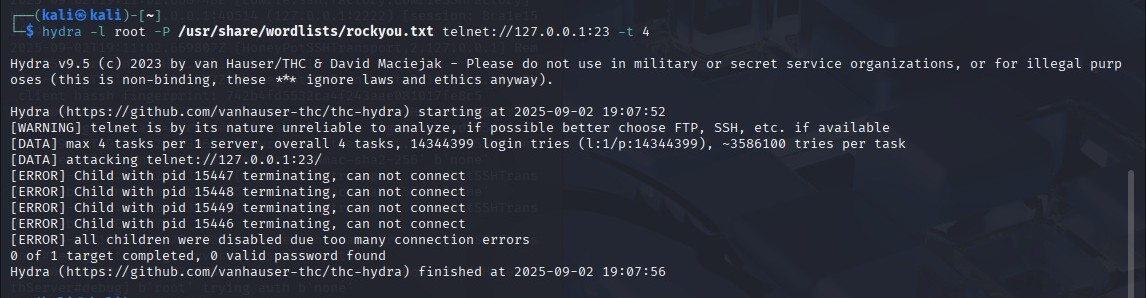


Explanation:

* **sudo bash -c "..."** → Executes multiple commands as root in one line (useful for stopping and restarting Cowrie).
* **sudo su - cowrie** → Switches to the cowrie user account.
* **cd /home/cowrie/cowrie** → Moves to the Cowrie installation directory.
* **bin/cowrie stop** → Stops the honeypot if it’s running.
* **bin/cowrie start** → Starts Cowrie under the correct user.
* **bin/cowrie status** → Checks that Cowrie is running and shows its process ID.

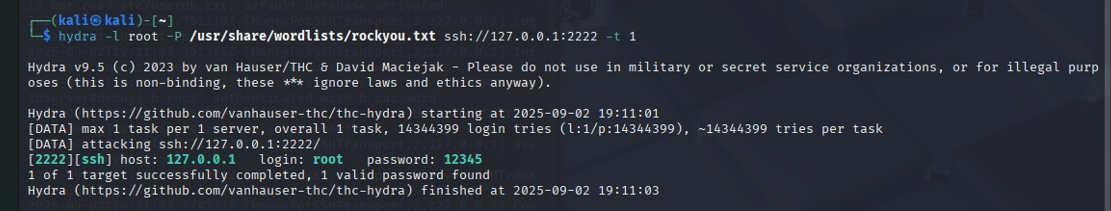
## Hydra Brute-Force Attack on Telnet (Failed Attempt)

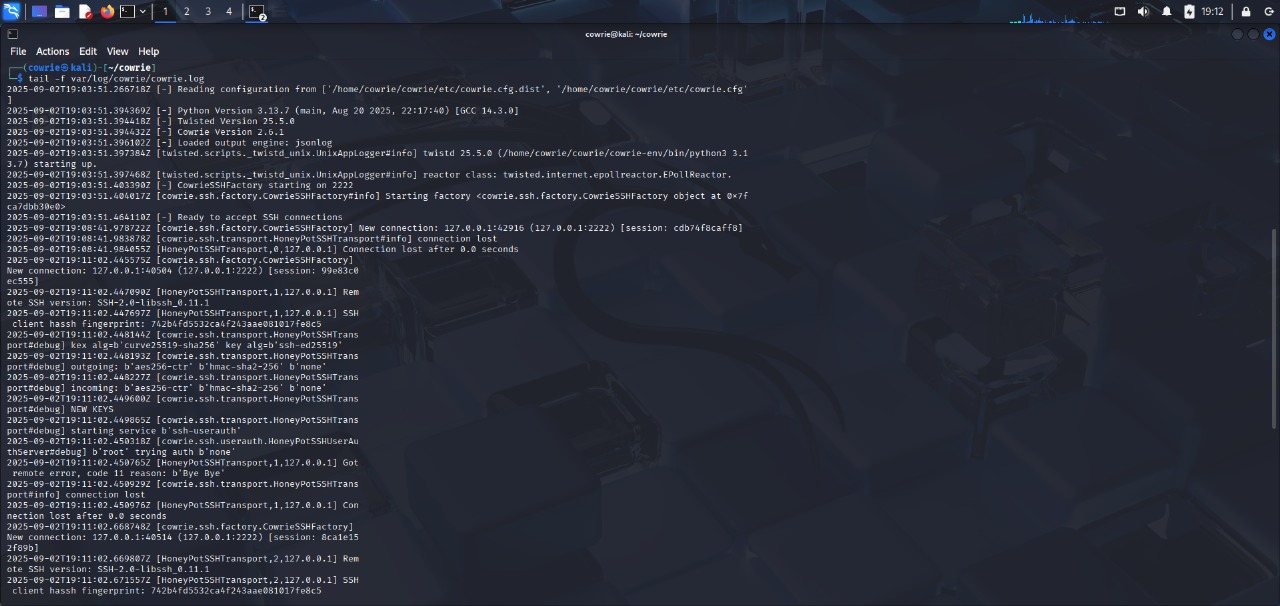
The output shows a failed brute-force attempt on the Telnet service at 127.0.0.1 using the rockyou.txt wordlist, blocked by security measures. This demonstrates that the system’s defenses, such as rate limiting or connection restrictions, successfully prevented unauthorized access.

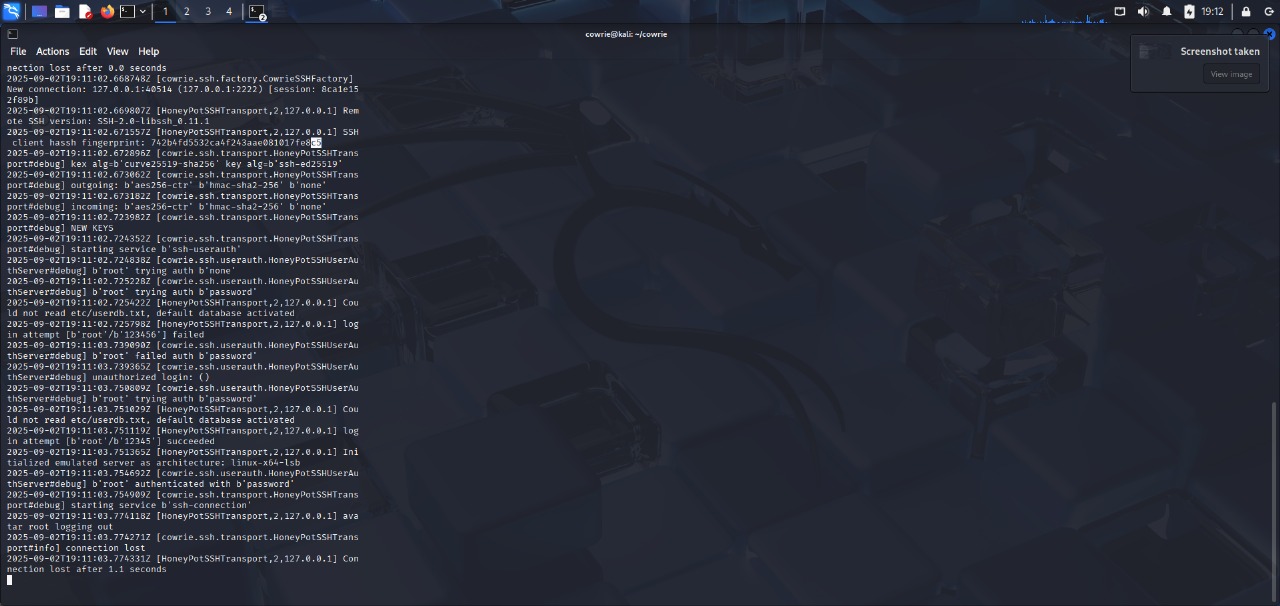


## Brute Force SSH Attempt Using Hydra (Successful Attempt)

Using Hydra on the local Cowrie honeypot, I performed an SSH brute-force attack with the rockyou.txt wordlist, which successfully revealed the credentials root:12345. This demonstrates how the honeypot logs attacker activity and can capture credential attempts.

Analyzing Captured Attack Logs





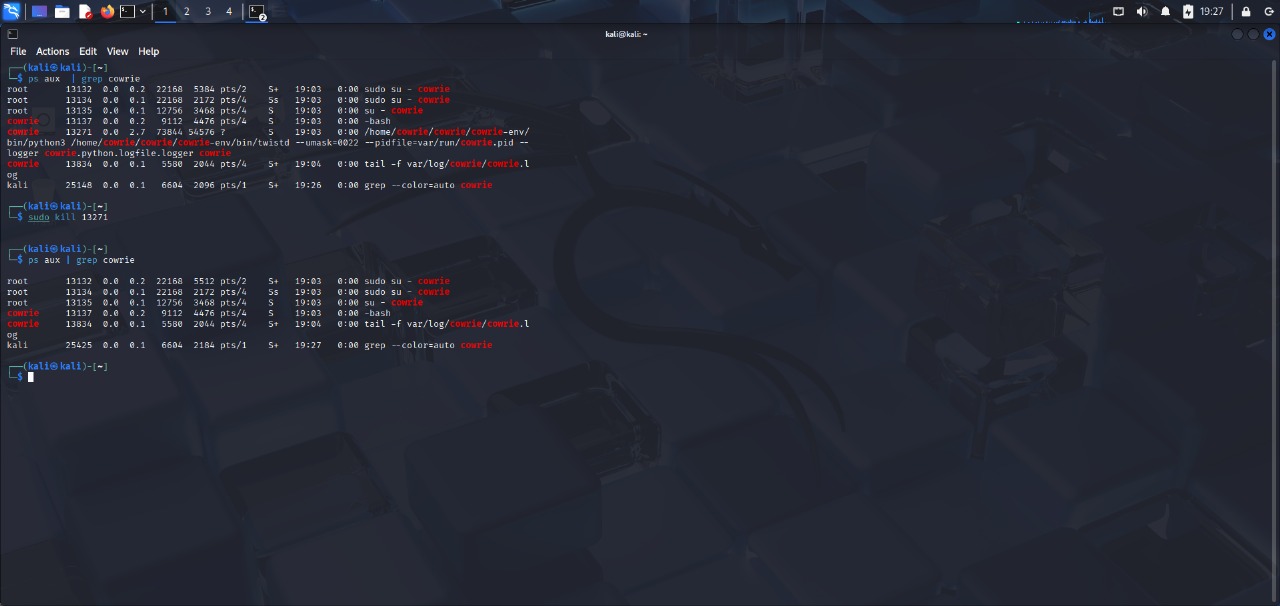
# Reverse Shell Interaction Captured by Cowrie Honeypot

From the attacker’s terminal, a Netcat listener was initiated and a reverse shell attempted. The Cowrie honeypot on the server side logged the incoming connection and recorded all executed commands, capturing the complete interaction for analysis.

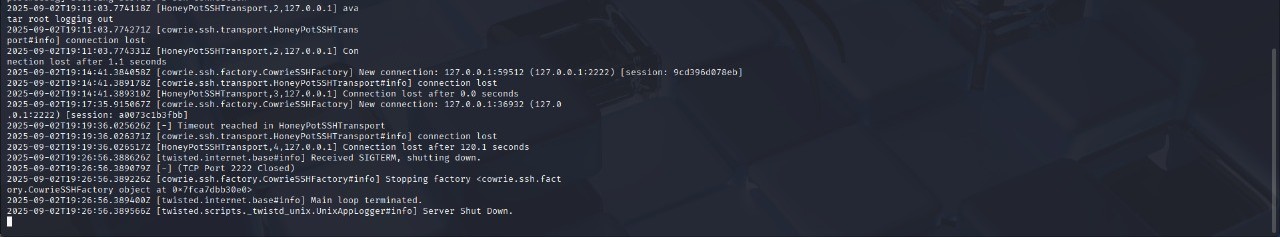


# Stopping the Cowrie Honeypot

The terminal commands show that Cowrie is running, allow safely stopping a specific honeypot process, and provide real-time monitoring of its logs to track all attacker activity.



Real-Time Honeypot Terminal Output

Conclusion

This project provided hands-on experience with honeypots and cybersecurity monitoring. Key takeaways include:

* Gained hands-on experience with honeypots and cybersecurity monitoring.
* Understood attacker behavior and common brute-force techniques.
* Learned how honeypots log commands, login attempts, and file transfers.
* Applied deception techniques to safely study attacks.
* Strengthened practical knowledge of cybersecurity defenses and attack analysis.