

This guide will walk you through integrating your Cowrie SSH honeypot with Slack using a Python script that sends alerts in real-time when specific honeypot activity occurs.

✓ Prerequisites

- Cowrie is already installed and running (see the Cowrie Install Guide)
 - A Slack workspace with permissions to create an app
-

1. Create a Python Script

Make sure you are inside your `cowrie` directory:

```
cd ~/cowrie
nano slack_alerts.py
```

```
(cowrie-env)(cowrie@kali)~|~/cowrie|
$ ls
bin          docker      INSTALL.rst  pyproject.toml  requirements-pool.txt  src
CHANGELOG.rst  docs       LICENSE.rst  README.rst      requirements.txt       tox.ini
CONTRIBUTING.rst  etc        Makefile    requirements-dev.txt  setup.cfg             var
cowrie-env     honeyfs    MANIFEST.in requirements-output.txt  setup.py
```

```
(cowrie-env)(cowrie@kali)~|~/cowrie|
$ nano slack_alert.py
```

Paste the contents of the provided `slack_alerts.py` script. You can find this file in the GitHub repo or use the template in this project.

Before saving, update the following variables:

- Replace `WEBHOOK_URL` with your Slack Incoming Webhook URL
 - Make sure `LOG_FILE` points to your Cowrie log file (e.g., `/home/cowrie/cowrie/var/log/cowrie/cowrie.log`)
-

2. Set Up Slack Webhook

1. Go to <https://api.slack.com/apps>

2. Click **Create New App > From Scratch**
3. Name it and select your workspace
4. Select **Incoming Webhooks** from the features list
5. Toggle to enable Incoming Webhooks
6. Click **Add New Webhook to Workspace**
7. Choose a Slack channel to receive notifications
8. Copy the **Webhook URL**

Paste this URL into the `WEBHOOK_URL` field in your script.

Honeypot

Settings

Basic Information

Collaborators

Socket Mode

Install App

Manage Distribution

Features

App Home

Agents & AI Apps NEW

Workflow Steps NEW

Org Level Apps

Incoming Webhooks

Interactivity & Shortcuts

Slash Commands

Steps from Apps LEGACY

OAuth & Permissions

Event Subscriptions

User ID Translation

Incoming Webhooks

Activate Incoming Webhooks

On

Incoming webhooks are a simple way to post messages from external sources into Slack. They make use of normal HTTP requests with a JSON payload, which includes the message and a few other optional details. You can include [message attachments](#) to display richly-formatted messages.

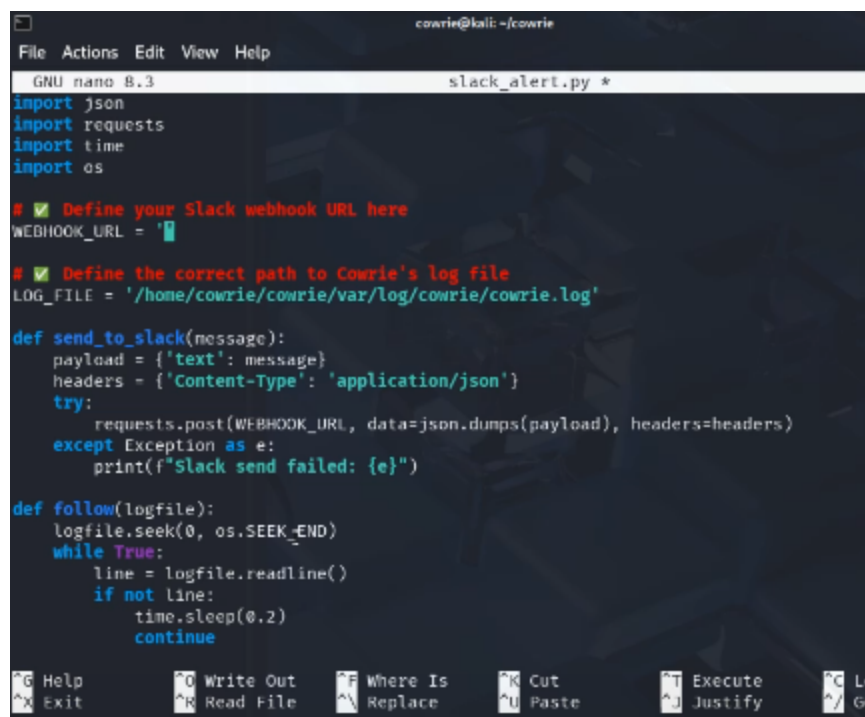
Adding incoming webhooks requires a bot user. If your app doesn't have a [bot user](#), we'll add one for you.

Each time your app is installed, a new Webhook URL will be generated.

If you deactivate incoming webhooks, new Webhook URLs will not be generated when your app is installed to your team. If you'd like to remove access to existing Webhook URLs, you will need to [Revoke All OAuth Tokens](#).

Webhook URLs for Your Workspace

To dispatch messages with your webhook URL, send your [message](#) in JSON as the body of an `application/json` POST request.

A screenshot of a terminal window showing the nano text editor editing a file named slack_alert.py. The script contains Python code for sending Slack alerts and following log files. The code includes imports for json, requests, time, and os. It defines a WEBHOOK_URL variable, a LOG_FILE variable, a send_to_slack function, and a follow function. The nano editor's menu bar at the top shows File, Actions, Edit, View, and Help. The status bar at the bottom shows various keyboard shortcuts like G Help, X Exit, O Write Out, R Read File, F Where Is, N Replace, K Cut, U Paste, T Execute, J Justify, C Lo, and Gt.

```

cowrie@kali: ~/cowrie
File Actions Edit View Help
GNU nano 8.3 slack_alert.py *
import json
import requests
import time
import os

# Define your Slack webhook URL here
WEBHOOK_URL = ''

# Define the correct path to Cowrie's log file
LOG_FILE = '/home/cowrie/cowrie/var/log/cowrie/cowrie.log'

def send_to_slack(message):
    payload = {'text': message}
    headers = {'Content-Type': 'application/json'}
    try:
        requests.post(WEBHOOK_URL, data=json.dumps(payload), headers=headers)
    except Exception as e:
        print(f"Slack send failed: {e}")

def follow(logfile):
    logfile.seek(0, os.SEEK_END)
    while True:
        line = logfile.readline()
        if not line:
            time.sleep(0.2)
            continue

```

Once you have the script and make sure that your webhook url and log file are correct we will save the file

3. Save and Run the Script

Save the script and make it executable (if necessary):

```
chmod +x slack_alerts.py
python3 slack_alerts.py
```

A screenshot of a terminal window showing the execution of the script. The prompt is (cowrie-env)(cowrie@kali)~[~/cowrie]. The user enters 'cd /home/cowrie/cowrie' and then 'python3 slack_alert.py'. The cursor is at the end of the second command.

```

(cowrie-env)(cowrie@kali)~[~/cowrie]
$ cd /home/cowrie/cowrie
python3 slack_alert.py

```

Leave this script running in a terminal or consider running it as a background process using **tmux** or **screen**.

4. Test the Integration

Trigger some activity in your honeypot:

ssh cowrie@localhost

```
(kali@kali)-[~]
$ ssh root@localhost

root@localhost'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:~# kmkk
-bash: kmkk: command not found
root@svr04:~# whoami
root
root@svr04:~# ls
root@svr04:~# cd
root@svr04:~#
```

```
test message from cowrie honeypot setup

HoneyPot APP 12:35 AM
New Session Started: 2025-04-02T00:35:29.937182Z [cowrie.ssh.factory.CowrieSSHFactory] New connection: 127.0.0.1:54842 (127.0.0.1:22)
[session: fc946b2e8677]

Login Attempt: 2025-04-02T00:35:32.636651Z [HoneyPotSSHTransport,6,127.0.0.1] login attempt [b'root'/b'kali'] succeeded






Command Executed: 2025-04-02T00:35:36.892048Z [HoneyPotSSHTransport,6,127.0.0.1] CMD: kmkk

Command Executed: 2025-04-02T00:35:44.704206Z [HoneyPotSSHTransport,6,127.0.0.1] CMD: whoami

Command Executed: 2025-04-02T00:35:51.214841Z [HoneyPotSSHTransport,6,127.0.0.1] CMD: ls

Command Executed: 2025-04-02T00:35:53.965362Z [HoneyPotSSHTransport,6,127.0.0.1] CMD: cd
```

Try running some commands or just logging in with random passwords. You should see messages in your Slack channel like:

-  Login Attempt
-  Command Executed
-  File Download Attempt
-  New Session Started
-  Session Ended

 **Success!**

You've successfully integrated Slack with Cowrie, yayyy!! Every time an attacker interacts with your honeypot, Slack will notify you in real-time, allowing you to monitor attacks as they happen.

Tip: You can extend this integration to send alerts to other platforms like Discord, email, or log to a database.

Next Steps

- Add support for log rotation
- Log alerts to a file for offline analysis
- Build a dashboard with ELK, Wazuh, or Splunk