Implementation of Honeypot using Cowrie

REDTEAM
HACKER ACADEMY

Submitted by:
AKHIL K
ADCD

Date:16 April 2025

INDEX

1. Introduction

Cybersecurity threats are increasing in frequency, sophistication, and complexity. As digital infrastructure expands across various sectors, cybercriminals are constantly looking for new ways to exploit vulnerabilities in systems, applications, and networks. These attacks can lead to severe consequences such as data breaches, financial loss, and a significant disruption of services. Traditional security systems like firewalls, antivirus software, and intrusion detection/prevention systems (IDS/IPS) have become increasingly inadequate in addressing advanced persistent threats (APTs), zero-day exploits, and other novel attack vectors.

In response to these challenges, cybersecurity researchers and organizations are turning to **honeypots** as a proactive and innovative method of studying and defending against cyber threats. A honeypot is a system or resource deliberately designed to appear as a legitimate target for attackers, while being isolated from the production environment. The goal is to deceive attackers into interacting with the honeypot, allowing defenders to observe their tactics, techniques, and procedures (TTPs) in real-time.

This project focuses on the implementation of a **Cowrie honeypot**, a medium-interaction honeypot tailored for capturing malicious activities associated with SSH (Secure Shell) and Telnet protocols. Cowrie simulates a vulnerable system by emulating a Linux shell and file system, offering an environment where attackers can attempt unauthorized login attempts and interact with the system as they would in a real attack. The key feature of Cowrie is its ability to log a wide range of activities such as brute-force login attempts, file downloads/uploads, and post-login commands executed by the attackers.

The primary aim of this project is to utilize Cowrie to gather data on the behaviors and strategies employed by cybercriminals when attempting to compromise a system. By deploying a Cowrie honeypot within a controlled environment, we can capture a variety of attacker interactions that would be otherwise difficult to observe in a real-world setting. This data will provide valuable insights into the tools, techniques, and vulnerabilities commonly exploited by attackers, as well as the trends in attack behavior over time. Moreover, it will help inform the development of more robust detection systems and defense mechanisms that can better protect real systems from such threats.

2. Project Objectives

The primary goal of this project is to deploy and configure a **Cowrie honeypot** to simulate a vulnerable SSH and Telnet server, enabling us to capture real-world cyberattacks and study attacker behavior in a controlled environment. Through this, we aim to gain a deeper understanding of the evolving landscape of cyber threats and the techniques employed by adversaries. The specific objectives of the project are as follows:

1. **Deploy a Cowrie Honeypot to Simulate a Vulnerable SSH/Telnet Server**
   The first objective is to set up and deploy Cowrie on a designated machine to simulate an SSH/Telnet server vulnerable to common attack methods such as brute-force login attempts. The honeypot will be configured to imitate an operating system and service environment commonly targeted by attackers, allowing for a realistic and engaging decoy system. The deployment will focus on ensuring the honeypot is isolated from other systems to prevent unintended compromises.

2. **Capture and Analyze Attacker Behavior**
   A key aspect of the project is to gather detailed information about how attackers approach and interact with the honeypot. This will include tracking login attempts, commands issued, files uploaded or downloaded, and any other suspicious activities. By capturing these interactions, we can build a database of attack methods and gain insights into how attackers exploit system vulnerabilities. We aim to monitor attackers' Tactics, Techniques, and Procedures (TTPs) through real-time analysis and logging.

3. **Understand Common Intrusion Techniques**
   By analyzing the interactions with the honeypot, we will identify common attack vectors and intrusion techniques used by cybercriminals. These may include brute-force attacks, exploitation of weak or default credentials, credential stuffing, and other methods often associated with SSH and Telnet service breaches. Understanding these techniques is essential in identifying patterns and trends, which will help enhance defensive strategies. Additionally, this objective will help determine the frequency and distribution of various attack methods.

4. **Provide Insights for Improving Security Posture**
   The ultimate goal of the project is to leverage the data collected from the Cowrie honeypot to provide actionable insights into improving organizational security posture. By understanding the most prevalent attack techniques, we can better protect real systems through targeted defenses such as stronger authentication methods, proper access control, and timely patching of known vulnerabilities. Furthermore, the findings can inform the development of intrusion detection

systems (IDS) and proactive monitoring strategies to detect and mitigate attacks before they reach critical systems.

3. Overview of Honeypots

A **honeypot** is a security resource specifically created to act as a decoy, designed to attract and deceive potential attackers. It appears to be a legitimate, vulnerable system or service, yet it is isolated from critical infrastructure to prevent any real harm. The primary function of a honeypot is to divert attackers from real systems and to observe and analyze their activities in a controlled environment. By engaging attackers in a fake environment, honeypots offer security professionals a unique opportunity to monitor, understand, and learn from adversarial behavior, which can then inform the development of improved defense mechanisms.

Honeypots are distinct from traditional security tools such as firewalls and intrusion detection systems (IDS) in that they do not focus on stopping attacks or blocking unauthorized access. Instead, they serve as bait for malicious actors, intentionally presenting weaknesses or vulnerabilities that lure attackers. Once an attacker interacts with the honeypot, their behavior can be logged and analyzed without posing any risk to the actual network or systems. This allows for the collection of invaluable intelligence on how cybercriminals operate, their tools, techniques, and procedures (TTPs), as well as the attack vectors they exploit.

1. The main advantage of honeypots over traditional defensive measures is their ability to gather detailed, real-time information on cyberattacks. By carefully observing the strategies employed by attackers, security researchers can gain insight into the types of vulnerabilities most commonly targeted, the tools or exploits being used, and even the geographical location or identity of the attackers. This data is crucial for enhancing proactive defenses and fortifying real systems against future attacks. **Low-Interaction Honeypots:** These honeypots simulate the presence of a service or system but offer limited functionality. They are designed to attract automated attacks such as botnets or scanning tools. Because they do not provide full system emulation, they are easier to deploy and manage but offer less detailed interaction data.

2. **High-Interaction Honeypots:** These honeypots provide a fully functional system or service, allowing attackers to interact with it as they would with a real target. High-interaction honeypots are more resource-intensive and require additional precautions to ensure that attackers cannot access the rest of the network. However, they provide much richer data, including post-compromise actions like command execution, file manipulation, and privilege escalation.

3. **Medium-Interaction Honeypots:** Positioned between low and high-interaction honeypots, medium-interaction systems like **Cowrie** simulate a real service (such as

SSH or Telnet) but do not give attackers full access to the underlying system. These honeypots offer more detailed data than low-interaction honeypots but are safer and easier to manage than high-interaction ones.

4. Introduction to Cowrie

Cowrie is a highly regarded, medium-interaction honeypot designed to simulate a vulnerable system that attackers target via **SSH** (Secure Shell) and **Telnet** protocols. Written in Python, Cowrie is capable of emulating a full Linux shell environment and captures detailed logs of attacker activities, making it an invaluable tool for cybersecurity research and threat intelligence. It is widely used by security professionals to study real-world attack behaviors and gain insights into adversarial tactics, techniques, and procedures (TTPs).

The core functionality of Cowrie revolves around simulating the interaction between an attacker and a compromised system. When an attacker attempts to connect to the honeypot, typically through brute-force login attempts or unauthorized access attempts, Cowrie logs all their actions. These interactions are recorded in real-time, capturing valuable data such as the commands typed into the shell, files uploaded or downloaded, and any further activity carried out by the attacker after a successful login attempt. This allows security analysts to gain deep visibility into the methods attackers use to breach systems

Some of the key features of Cowrie include:

- **SSH and Telnet Emulation:** It captures all login attempts, including brute-force and dictionary attacks.
- **Linux Shell Simulation:** It emulates a shell environment, logging every command the attacker issues.
- **File Upload/Download Simulation:** Cowrie logs all file transfer activities, including malware uploads.
- **Interactive Session Logging:** Cowrie logs attacker interactions with the shell, providing detailed insights into their tactics and tools.
- **Customizable Environment:** Users can customize the simulated Linux environment to mimic specific distributions or configurations.
- **Easy Integration with Other Tools:** Logs can be forwarded to SIEM (Security Information and Event Management) systems for centralized analysis and monitoring.

5.System Requirements

- Operating System: Ubuntu Server 20.04 LTS

- RAM: Minimum 2GB

- Tools Required: Python3, Virtualenv, Git, Cowrie, Logstash, Kibana (optional for analysis)

6. Implementation Steps

1. System Setup

```
sudo apt update && sudo apt upgrade -y

sudo apt install git python3 python3-pip python3-virtualenv -y
```

2. Install Cowrie

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

cd cowrie
```

Create a Python virtual environment for Cowrie to manage its dependencies:

```
virtualenv cowrie-env
source cowrie-env/bin/activate
```

Install the required Python packages:

```
pip install --upgrade pip
pip install -r requirements.txt
```

3.Configuration

```
cp etc/cowrie.cfg.dist etc/cowrie.cfg
cp etc/userdb.txt.dist etc/userdb.txt
```

Edit the cowrie.cfg file to customize ports, enable or disable Telnet/SSH, and set logging options.

```
nano etc/cowrie.cfg
```

4.Run Cowrie

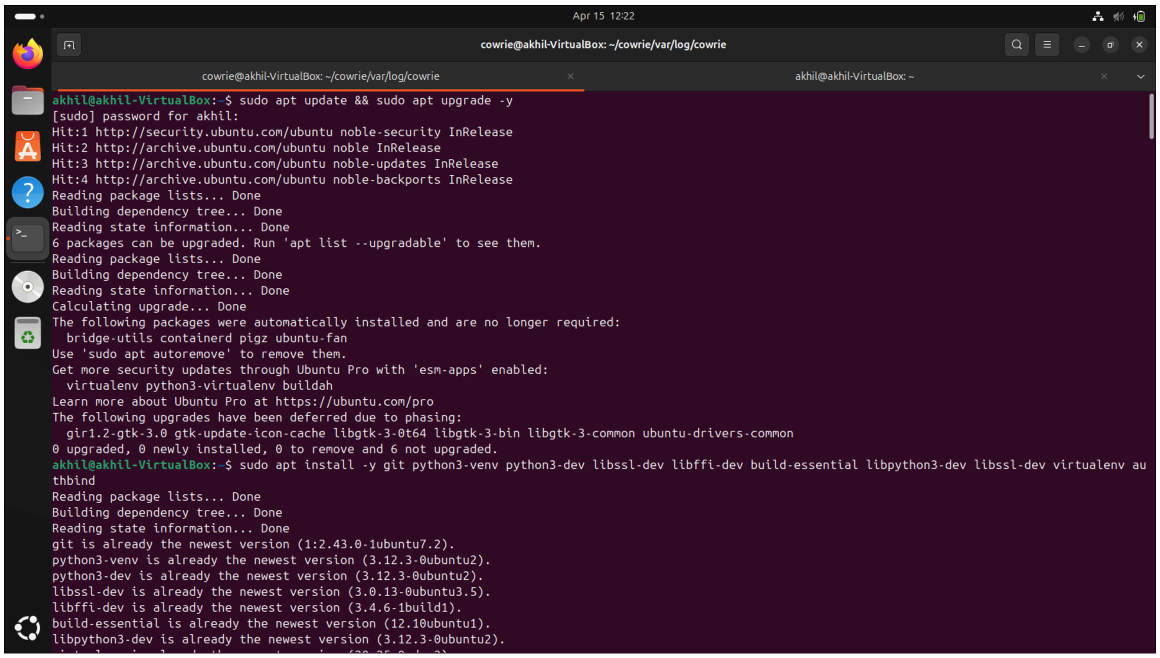
```
bin/cowrie start
```

5.Check status

```
bin/cowrie status
```

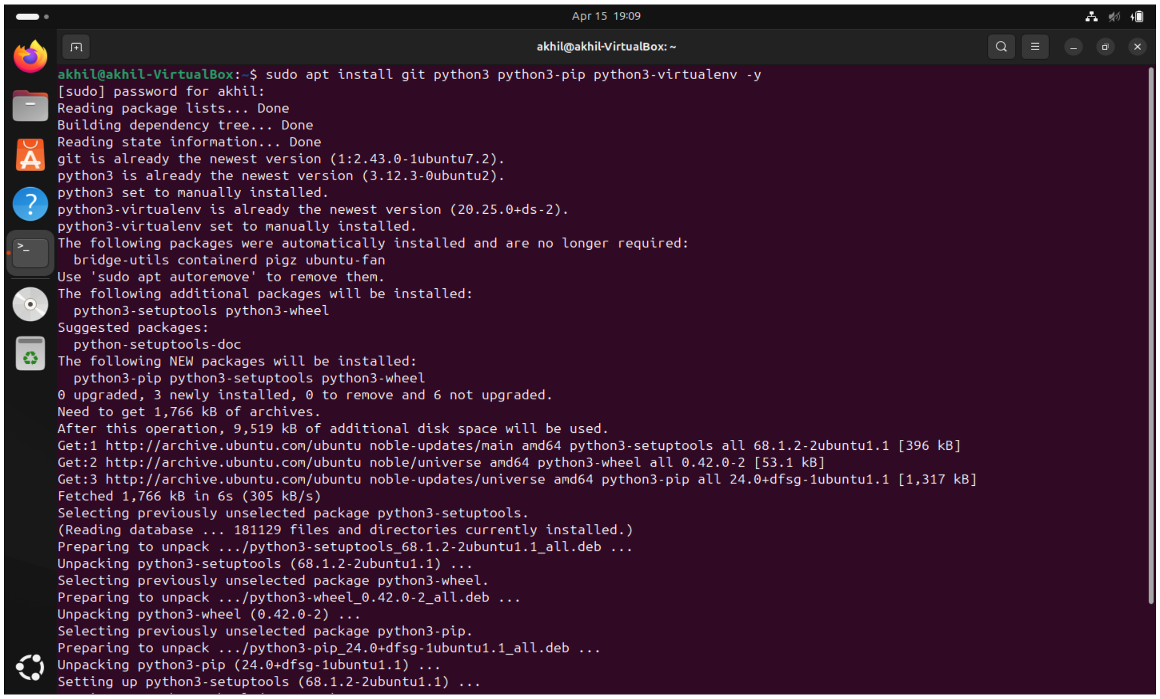
6 Monitoring

```
cowrie/var/log/cowrie/
```

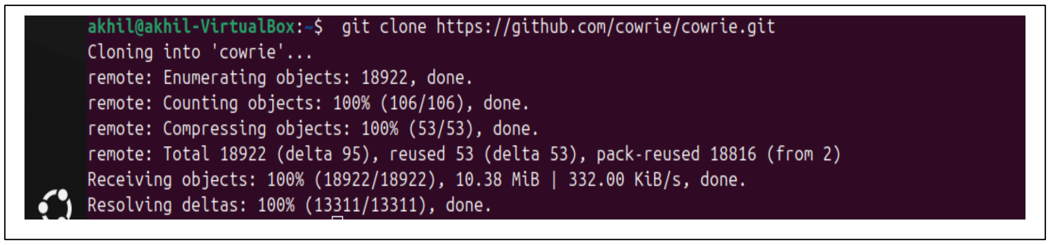


Apr 15 12:22
cowrie@akhil-VirtualBox: ~/cowrie/var/log/cowrie
cowrie@akhil-VirtualBox: ~/cowrie/var/log/cowrie
akhil@akhil-VirtualBox: ~

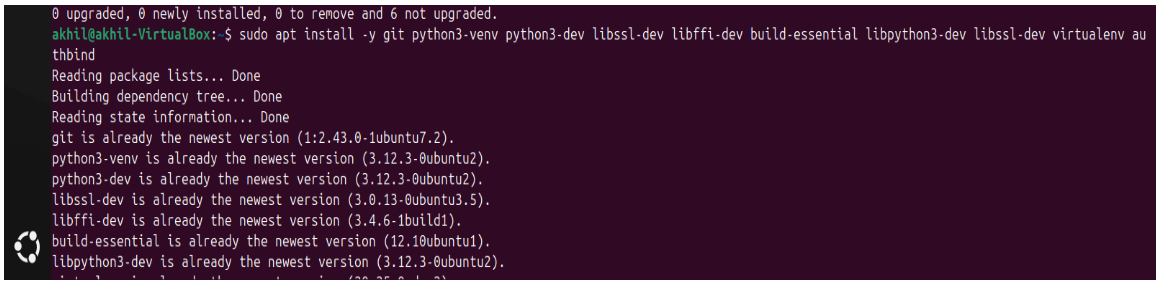
```
akhil@akhil-VirtualBox:~$ sudo apt update && sudo apt upgrade -y
[sudo] password for akhil:
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 http://archive.ubuntu.com/ubuntu noble InRelease
Hit:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
6 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages were automatically installed and are no longer required:
  bridge-utils containerd pigz ubuntu-fan
Use 'sudo apt autoremove' to remove them.
Get more security updates through Ubuntu Pro with 'esm-apps' enabled:
  virtualenv python3-virtualenv buildah
Learn more about Ubuntu Pro at https://ubuntu.com/pro
The following upgrades have been deferred due to phasing:
  gir1.2-gtk-3.0 gtk-update-icon-cache libgtk-3-0t64 libgtk-3-bin libgtk-3-common ubuntu-drivers-common
0 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.
akhil@akhil-VirtualBox:~$ sudo apt install -y git python3-venv python3-dev libssl-dev libffi-dev build-essential libpython3-dev libssl-dev virtualenv au
thbind
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.43.0-1ubuntu7.2).
python3-venv is already the newest version (3.12.3-0ubuntu2).
python3-dev is already the newest version (3.12.3-0ubuntu2).
libssl-dev is already the newest version (3.0.13-0ubuntu3.5).
libffi-dev is already the newest version (3.4.6-1build1).
build-essential is already the newest version (12.10ubuntu1).
libpython3-dev is already the newest version (3.12.3-0ubuntu2).
```



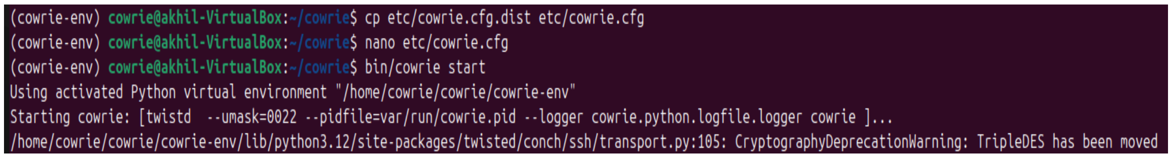
```
akhil@akhil-VirtualBox:~$ sudo apt install git python3 python3-pip python3-virtualenv -y
[sudo] password for akhil:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.43.0-1ubuntu7.2).
python3 is already the newest version (3.12.3-0ubuntu2).
python3 set to manually installed.
python3-virtualenv is already the newest version (20.25.0+ds-2).
python3-virtualenv set to manually installed.
The following packages were automatically installed and are no longer required:
  bridge-utils containerd pigz ubuntu-fan
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  python3-setuptools python3-wheel
Suggested packages:
  python-setuptools-doc
The following NEW packages will be installed:
  python3-pip python3-setuptools python3-wheel
0 upgraded, 3 newly installed, 0 to remove and 6 not upgraded.
Need to get 1,766 kB of archives.
After this operation, 9,519 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 python3-setuptools all 68.1.2-2ubuntu1.1 [396 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble/universe amd64 python3-wheel all 0.42.0-2 [53.1 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 python3-pip all 24.0+dfsg-1ubuntu1.1 [1,317 kB]
Fetched 1,766 kB in 6s (305 kB/s)
Selecting previously unselected package python3-setuptools.
(Reading database ... 181129 files and directories currently installed.)
Preparing to unpack .../python3-setuptools_68.1.2-2ubuntu1.1_all.deb ...
Unpacking python3-setuptools (68.1.2-2ubuntu1.1) ...
Selecting previously unselected package python3-wheel.
Preparing to unpack .../python3-wheel_0.42.0-2_all.deb ...
Unpacking python3-wheel (0.42.0-2) ...
Selecting previously unselected package python3-pip.
Preparing to unpack .../python3-pip_24.0+dfsg-1ubuntu1.1_all.deb ...
Unpacking python3-pip (24.0+dfsg-1ubuntu1.1) ...
Setting up python3-setuptools (68.1.2-2ubuntu1.1) ...
```



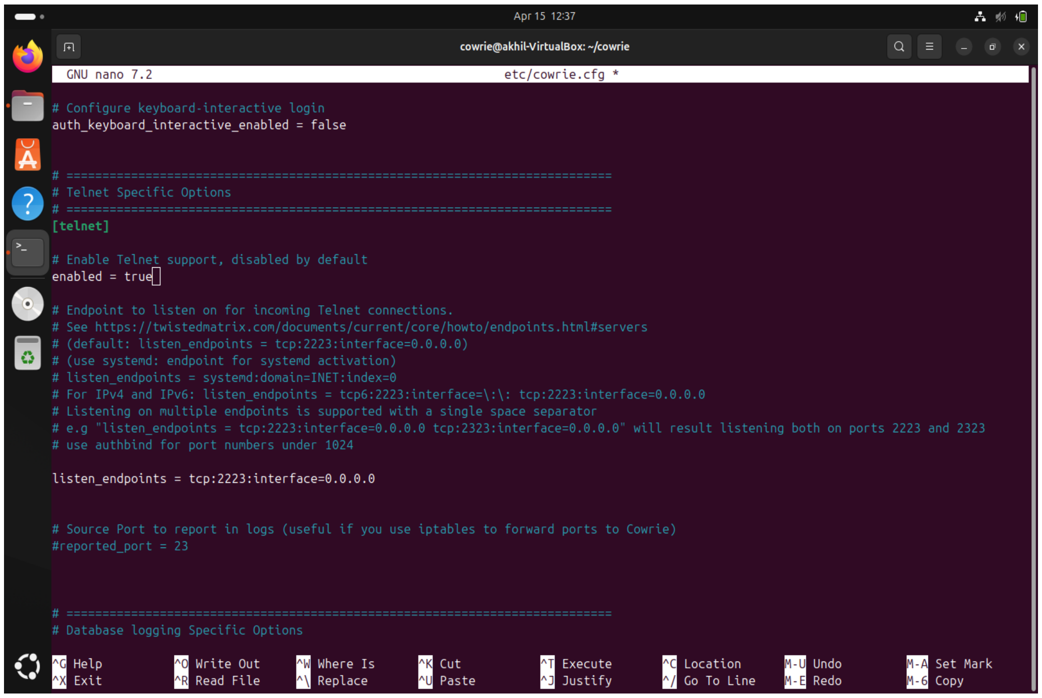
```
akhil@akhil-VirtualBox:~$ git clone https://github.com/cowrie/cowrie.git
Cloning into 'cowrie'...
remote: Enumerating objects: 18922, done.
remote: Counting objects: 100% (106/106), done.
remote: Compressing objects: 100% (53/53), done.
remote: Total 18922 (delta 95), reused 53 (delta 53), pack-reused 18816 (from 2)
Receiving objects: 100% (18922/18922), 10.38 MiB | 332.00 KiB/s, done.
Resolving deltas: 100% (13311/13311), done.
```



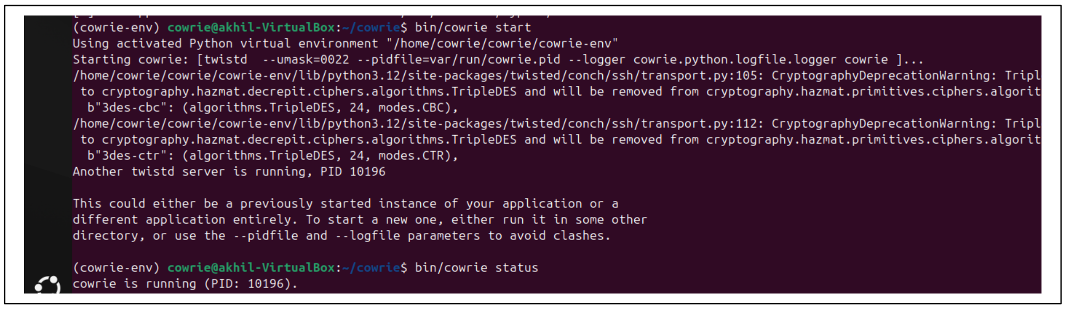
```
0 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.
akhil@akhil-VirtualBox:~$ sudo apt install -y git python3-venv python3-dev libssl-dev libffi-dev build-essential libpython3-dev libssl-dev virtualenv au
thbind
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.43.0-1ubuntu7.2).
python3-venv is already the newest version (3.12.3-0ubuntu2).
python3-dev is already the newest version (3.12.3-0ubuntu2).
libssl-dev is already the newest version (3.0.13-0ubuntu3.5).
libffi-dev is already the newest version (3.4.6-1build1).
build-essential is already the newest version (12.10ubuntu1).
libpython3-dev is already the newest version (3.12.3-0ubuntu2).
```



```
(cowrie-env) cowrie@akhil-VirtualBox:~/cowrie$ cp etc/cowrie.cfg.dist etc/cowrie.cfg
(cowrie-env) cowrie@akhil-VirtualBox:~/cowrie$ nano etc/cowrie.cfg
(cowrie-env) cowrie@akhil-VirtualBox:~/cowrie$ bin/cowrie start
Using activated Python virtual environment "/home/cowrie/cowrie/cowrie-env"
Starting cowrie: [twistd  --umask=0022 --pidfile=var/run/cowrie.pid --logger cowrie.python.logfile.logger cowrie ]...
/home/cowrie/cowrie/cowrie-env/lib/python3.12/site-packages/twisted/conch/ssh/transport.py:105: CryptographyDeprecationWarning: TripleDES has been moved
```



```
  GNU nano 7.2                                      etc/cowrie.cfg *

# Configure keyboard-interactive login
auth_keyboard_interactive_enabled = false


# ============================================================================
# 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 separator
# e.g "listen_endpoints = tcp:2223:interface=0.0.0.0 tcp:2323:interface=0.0.0.0" will result listening both on ports 2223 and 2323
# use authbind for port numbers under 1024

listen_endpoints = tcp:2223:interface=0.0.0.0


# Source Port to report in logs (useful if you use iptables to forward ports to Cowrie)
#reported_port = 23



# ============================================================================
# Database logging Specific Options

^G Help         ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Undo        M-A Set Mark
^X Exit         ^R Read File    ^\ Replace      ^U Paste        ^J Justify      ^/ Go To Line   M-E Redo        M-6 Copy
```



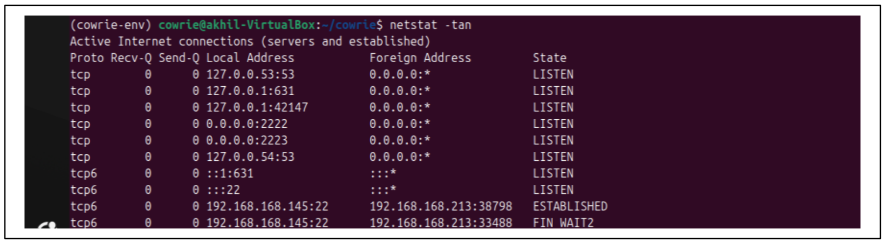
```
(cowrie-env) cowrie@akhil-VirtualBox:~/cowrie$ bin/cowrie start
Using activated Python virtual environment "/home/cowrie/cowrie/cowrie-env"
Starting cowrie: [twistd  --umask=0022 --pidfile=var/run/cowrie.pid --logger cowrie.python.logfile.logger cowrie ]...
/home/cowrie/cowrie/cowrie-env/lib/python3.12/site-packages/twisted/conch/ssh/transport.py:105: CryptographyDeprecationWarning: Tripl
 to cryptography.hazmat.decrepit.ciphers.algorithms.TripleDES and will be removed from cryptography.hazmat.primitives.ciphers.algorit
  b"3des-cbc": (algorithms.TripleDES, 24, modes.CBC),
/home/cowrie/cowrie/cowrie-env/lib/python3.12/site-packages/twisted/conch/ssh/transport.py:112: CryptographyDeprecationWarning: Tripl
 to cryptography.hazmat.decrepit.ciphers.algorithms.TripleDES and will be removed from cryptography.hazmat.primitives.ciphers.algorit
  b"3des-ctr": (algorithms.TripleDES, 24, modes.CTR),
Another twistd server is running, PID 10196

This could either be a previously started instance of your application or a
different application entirely. To start a new one, either run it in some other
directory, or use the --pidfile and --logfile parameters to avoid clashes.

(cowrie-env) cowrie@akhil-VirtualBox:~/cowrie$ bin/cowrie status
cowrie is running (PID: 10196).
```

7. Results and Findings

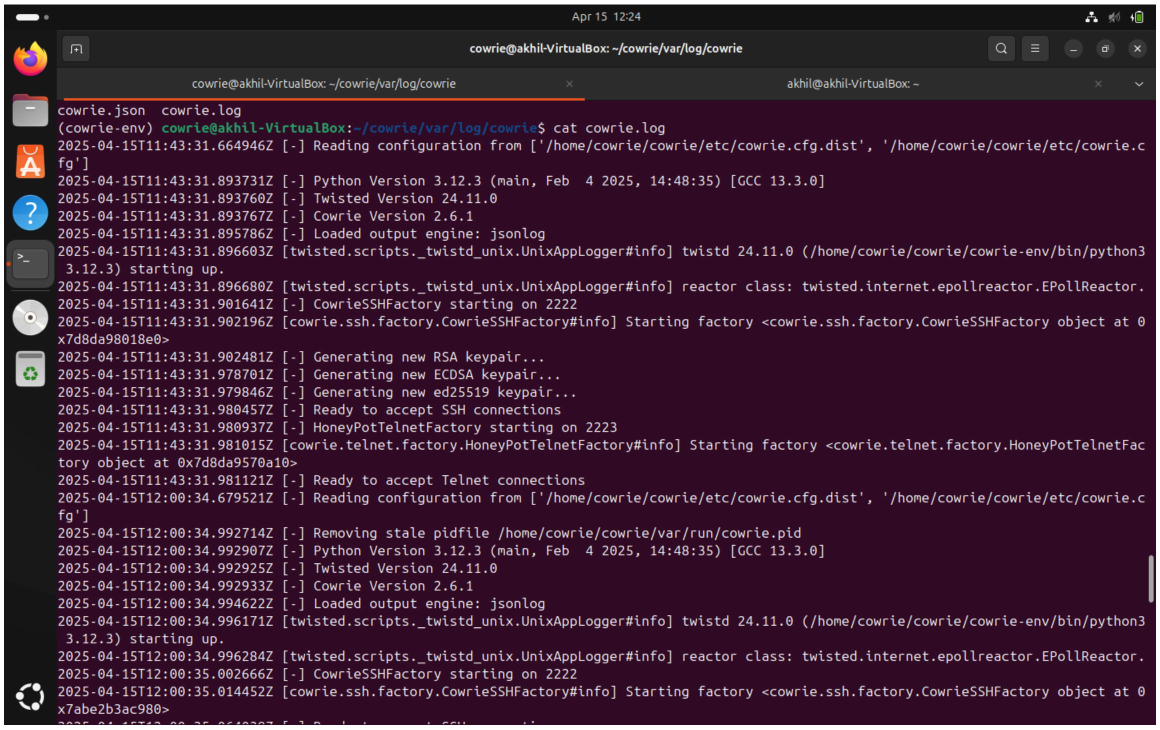
- Most attacks originated from botnets or automated scripts.

- Common ports targeted: 22 (SSH) and 23 (Telnet).

- Attackers often attempted to download shell scripts and cryptominers.



```
(cowrie-env) cowrie@akhil-VirtualBox:~/cowrie$ netstat -tan
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:42147         0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:2222            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:2223            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.54:53           0.0.0.0:*               LISTEN
tcp6       0      0 ::1:631                 :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 192.168.168.145:22      192.168.168.213:38798   ESTABLISHED
tcp6       0      0 192.168.168.145:22      192.168.168.213:33488   FIN WAIT2
```



Apr 15 12:24
cowrie@akhil-VirtualBox: ~/cowrie/var/log/cowrie
cowrie@akhil-VirtualBox: ~/cowrie/var/log/cowrie
akhil@akhil-VirtualBox: ~

```
cowrie.json  cowrie.log
(cowrie-env) cowrie@akhil-VirtualBox:~/cowrie/var/log/cowrie$ cat cowrie.log
2025-04-15T11:43:31.664946Z [-] Reading configuration from ['/home/cowrie/cowrie/etc/cowrie.cfg.dist', '/home/cowrie/cowrie/etc/cowrie.c
fg']
2025-04-15T11:43:31.893731Z [-] Python Version 3.12.3 (main, Feb  4 2025, 14:48:35) [GCC 13.3.0]
2025-04-15T11:43:31.893760Z [-] Twisted Version 24.11.0
2025-04-15T11:43:31.893767Z [-] Cowrie Version 2.6.1
2025-04-15T11:43:31.895786Z [-] Loaded output engine: jsonlog
2025-04-15T11:43:31.896603Z [twisted.scripts._twistd_unix.UnixAppLogger#info] twistd 24.11.0 (/home/cowrie/cowrie/cowrie-env/bin/python3
 3.12.3) starting up.
2025-04-15T11:43:31.896680Z [twisted.scripts._twistd_unix.UnixAppLogger#info] reactor class: twisted.internet.epollreactor.EPollReactor.
2025-04-15T11:43:31.901641Z [-] CowrieSSHFactory starting on 2222
2025-04-15T11:43:31.902196Z [cowrie.ssh.factory.CowrieSSHFactory#info] Starting factory <cowrie.ssh.factory.CowrieSSHFactory object at 0
x7d8da98018e0>
2025-04-15T11:43:31.902481Z [-] Generating new RSA keypair...
2025-04-15T11:43:31.978701Z [-] Generating new ECDSA keypair...
2025-04-15T11:43:31.979846Z [-] Generating new ed25519 keypair...
2025-04-15T11:43:31.980457Z [-] Ready to accept SSH connections
2025-04-15T11:43:31.980937Z [-] HoneyPotTelnetFactory starting on 2223
2025-04-15T11:43:31.981015Z [cowrie.telnet.factory.HoneyPotTelnetFactory#info] Starting factory <cowrie.telnet.factory.HoneyPotTelnetFac
tory object at 0x7d8da9570a10>
2025-04-15T11:43:31.981121Z [-] Ready to accept Telnet connections
2025-04-15T12:00:34.679521Z [-] Reading configuration from ['/home/cowrie/cowrie/etc/cowrie.cfg.dist', '/home/cowrie/cowrie/etc/cowrie.c
fg']
2025-04-15T12:00:34.992714Z [-] Removing stale pidfile /home/cowrie/cowrie/var/run/cowrie.pid
2025-04-15T12:00:34.992907Z [-] Python Version 3.12.3 (main, Feb  4 2025, 14:48:35) [GCC 13.3.0]
2025-04-15T12:00:34.992925Z [-] Twisted Version 24.11.0
2025-04-15T12:00:34.992933Z [-] Cowrie Version 2.6.1
2025-04-15T12:00:34.994622Z [-] Loaded output engine: jsonlog
2025-04-15T12:00:34.996171Z [twisted.scripts._twistd_unix.UnixAppLogger#info] twistd 24.11.0 (/home/cowrie/cowrie/cowrie-env/bin/python3
 3.12.3) starting up.
2025-04-15T12:00:34.996284Z [twisted.scripts._twistd_unix.UnixAppLogger#info] reactor class: twisted.internet.epollreactor.EPollReactor.
2025-04-15T12:00:35.002666Z [-] CowrieSSHFactory starting on 2222
2025-04-15T12:00:35.014452Z [cowrie.ssh.factory.CowrieSSHFactory#info] Starting factory <cowrie.ssh.factory.CowrieSSHFactory object at 0
x7abe2b3ac980>
```

8. Challenges and Limitations

- Risk of detection by advanced attackers.
- Must isolate honeypot to prevent actual system compromise.
- Requires frequent updates to remain relevant.
- High volume of false positives from benign scanners.
- Limited visibility into encrypted attacks (e.g., HTTPS).
- Resource-intensive if running multiple honeypots.
- Cannot detect zero-day exploits without specific signatures.
- Difficulty distinguishing between real threats and automated noise.
- Legal and ethical concerns when interacting with attackers.
- Potential for honeypot to be used as a launchpad if compromised.

9. Conclusion

Cowrie honeypot proved effective in capturing real-world attacks and provided valuable insights into attacker behavior. It is a cost-effective and educational tool for learning about intrusion techniques and improving cybersecurity readiness. The data collected offered valuable insights into common tactics, techniques, and procedures (TTPs) used by malicious actors, including the use of automated scripts, botnets, and the exploitation of well-known ports and services. The ability to observe these behaviors without putting production systems at risk highlights the practical benefits of honeypot technology in both research and operational cybersecurity. From an educational standpoint, Cowrie served as a hands-on learning platform, helping to reinforce concepts such as intrusion detection, log analysis, and threat intelligence. For students, researchers, and IT professionals alike, it offers a low-cost yet powerful way to explore modern attack vectors and strengthen defensive strategies. In addition, Cowrie's modularity and ease of configuration make it an accessible tool for institutions and individuals interested in cybersecurity experimentation. However, like any security tool, it must be carefully maintained, updated, and properly isolated to ensure that it remains a safe and effective part of a broader defensive strategy.

Overall, the project demonstrated that honeypots—when implemented thoughtfully—can play a key role in improving situational awareness, enhancing proactive defenses, and building practical cybersecurity skills.