CSCI 5742 Cybersecurity Programming and Analytics, Spring 2025

Team Members:
Ishika Upadhyay (111442194),
Yash Hindocha (111456729)

Project Title:
Adaptive Cyber Defense Framework using SDN and Honeypot Integration

1. Introduction:

- Modern cyberattacks are increasingly sophisticated, leveraging automated reconnaissance, scanning, and tailored exploitation to compromise systems rapidly.
- Traditional defense architectures are often static—with fixed IP addresses, predictable services, and unchanging network topologies—making them easy for attackers to fingerprint.
    - This project addresses these vulnerabilities by developing an Adaptive Cyber Defense Framework, integrating:
    - Software-Defined Networking (SDN) for programmable traffic control.
    - Moving Target Defense (MTD) via dynamic IP randomization to increase attacker uncertainty.
    - A Cowrie honeypot to simulate a vulnerable SSH service for deception and detailed attacker logging.

- The framework aims to:
    - Disrupt attacker reconnaissance efforts by altering visible IP addresses.
    - Redirect suspicious traffic to decoys without affecting legitimate users.
    - Log attacker activities for post-event analysis and threat intelligence gathering.

2. Project Objective:

- The project's primary goal was to build and test an integrated cyber defense system that combines SDN programmability with honeypot deception techniques.
- Specific objectives included:
    - Dynamically manipulate traffic flows using SDN controllers.
    - Implement IP obfuscation techniques to randomize host identities visible to attackers.
    - Deploy a Dockerized Cowrie honeypot simulating a vulnerable SSH service for capturing attacker interactions.

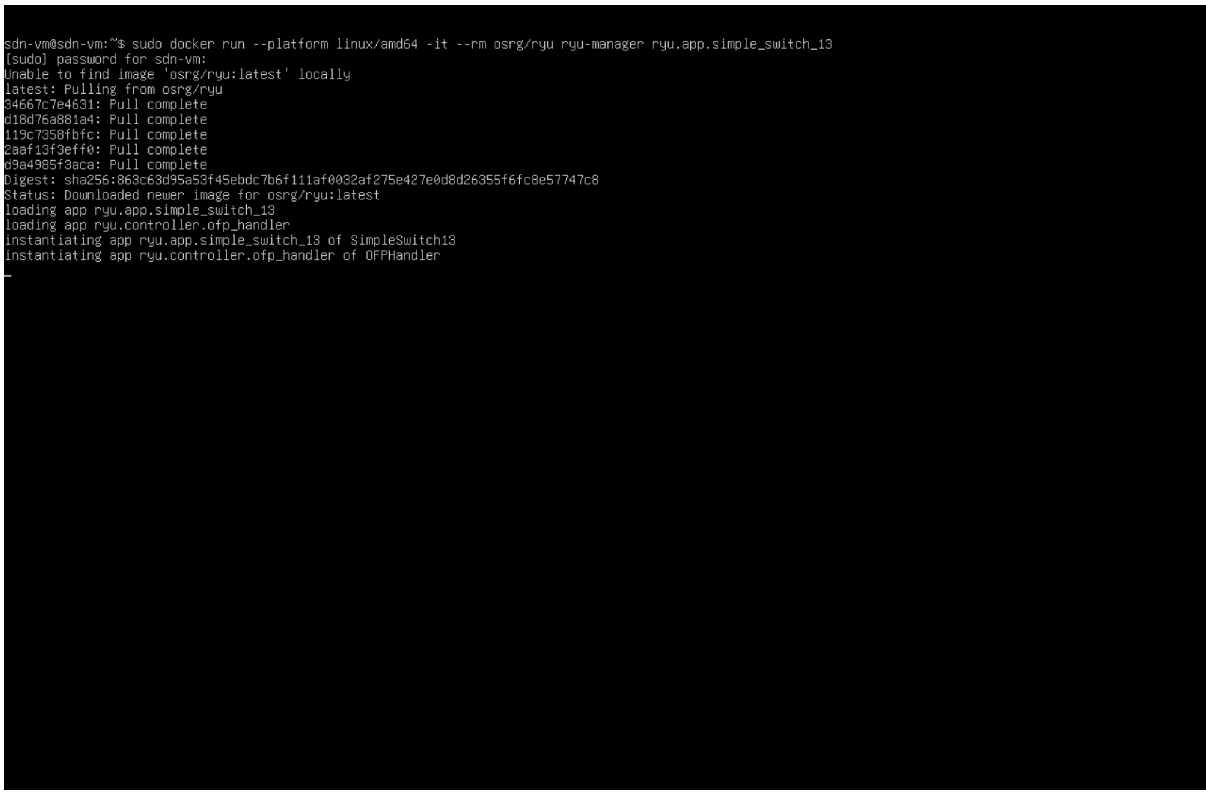
- Validate the framework's effectiveness using controlled attacks from a Kali Linux attacker VM.

- **The project was implemented on three virtual machines (VMs):**

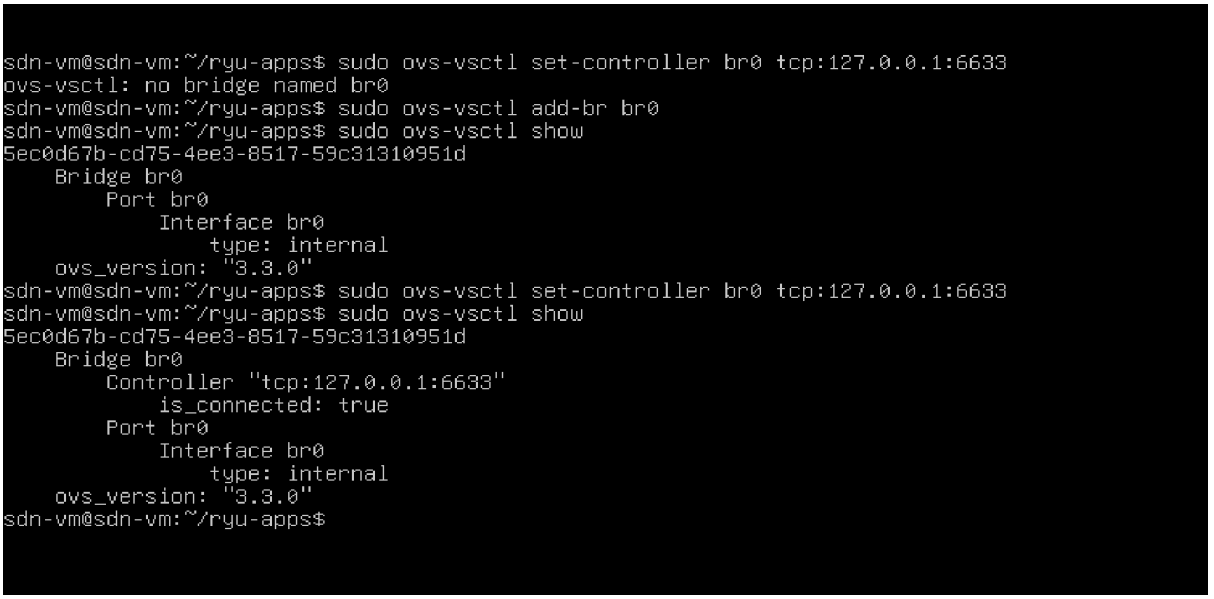
VM Role	IP Address	Function
SDN Controller	10.0.0.175	Hosts Ryu controller + OVS
Honeypot	10.0.0.28	Runs Dockerized Cowrie honeypot
Kali Attacker	10.0.0.130	Simulates attacker interactions

3. Implementation Stages:

- **Stage 1: SDN Environment Setup and Initial Configuration**
  - The first stage established the SDN control plane:
    - Installed Open vSwitch (OVS) and Docker on sdn-vm
    - Created an OVS bridge br0 to handle traffic
    - Assigned IP 10.0.0.175/24 to br0 and brought the bridge online
    - Configured the OVS bridge to connect to a Ryu controller listening on tcp:127.0.0.1:6633
    - Verified controller connection status as is_connected: true
  - **Purpose**:
    - This stage prepared the programmable network infrastructure, enabling OpenFlow rule management and packet redirection by the SDN controller.

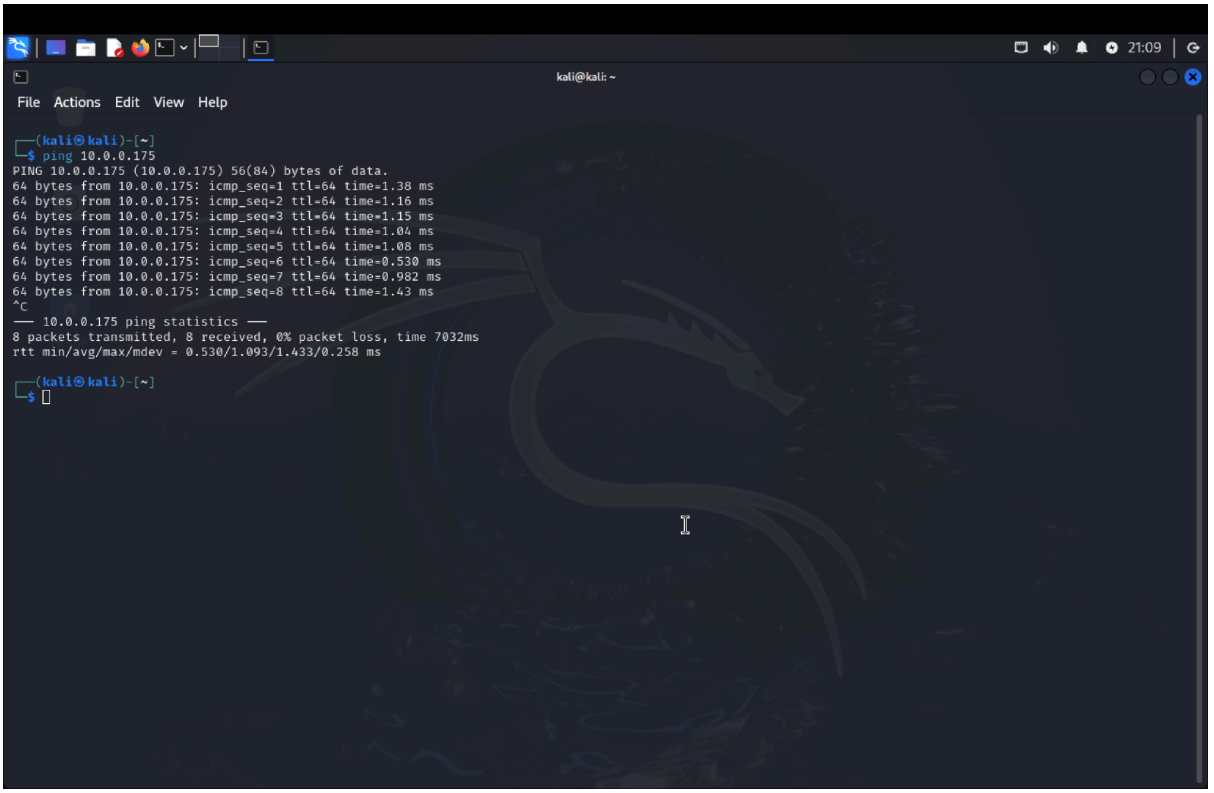


- **Stage 2: Deploying Ryu Controller with IP Randomization Application**
    - In the second stage, we developed and deployed a custom OpenFlow application:
        - Created ip_randomizer.py (Python script) to intercept traffic targeting 10.0.0.28 (honeypot) and dynamically rewrite the destination IP to randomized addresses
        - Ran the Ryu controller inside a Docker container with this app loaded
        - Verified incoming packets from kali-vm were processed by the controller (via packet-in events logged)
        - Tested connectivity by pinging 10.0.0.175 from kali-vm and observing responses logged by Ryu.
    - **Purpose:**
        - This stage validated that the SDN controller could intercept traffic and modify flow rules in real-time, forming the foundation for an IP obfuscation defense mechanism.



```
sdn-vm@sdn-vm:~/ryu-apps$ sudo ovs-vsctl set-controller br0 tcp:127.0.0.1:6633
ovs-vsctl: no bridge named br0
sdn-vm@sdn-vm:~/ryu-apps$ sudo ovs-vsctl add-br br0
sdn-vm@sdn-vm:~/ryu-apps$ sudo ovs-vsctl show
5ec0d67b-cd75-4ee3-8517-59c31310951d
    Bridge br0
        Port br0
            Interface br0
                type: internal
    ovs_version: "3.3.0"
sdn-vm@sdn-vm:~/ryu-apps$ sudo ovs-vsctl set-controller br0 tcp:127.0.0.1:6633
sdn-vm@sdn-vm:~/ryu-apps$ sudo ovs-vsctl show
5ec0d67b-cd75-4ee3-8517-59c31310951d
    Bridge br0
        Controller "tcp:127.0.0.1:6633"
            is_connected: true
        Port br0
            Interface br0
                type: internal
    ovs_version: "3.3.0"
sdn-vm@sdn-vm:~/ryu-apps$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether 0a:c3:f1:c5:da:54 brd ff:ff:ff:ff:ff:ff
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode DEFAULT group default
    link/ether 02:42:6d:1d:e9:fe brd ff:ff:ff:ff:ff:ff
8: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether b2:86:ce:d2:22:5b brd ff:ff:ff:ff:ff:ff
9: br0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 1a:01:72:99:8e:4a brd ff:ff:ff:ff:ff:ff
sdn-vm@sdn-vm:~/ryu-apps$
```

```
sdn-vm@sdn-vm:~/ryu-apps$ sudo ovs-vsctl set-controller br0 tcp:127.0.0.1:6633
ovs-vsctl: no bridge named br0
sdn-vm@sdn-vm:~/ryu-apps$ sudo ovs-vsctl add-br br0
sdn-vm@sdn-vm:~/ryu-apps$ sudo ovs-vsctl show
5ec0d67b-cd75-4ee3-8517-59c31310951d
    Bridge br0
        Port br0
            Interface br0
                type: internal
    ovs_version: "3.3.0"
sdn-vm@sdn-vm:~/ryu-apps$ sudo ovs-vsctl set-controller br0 tcp:127.0.0.1:6633
sdn-vm@sdn-vm:~/ryu-apps$ sudo ovs-vsctl show
5ec0d67b-cd75-4ee3-8517-59c31310951d
    Bridge br0
        Controller "tcp:127.0.0.1:6633"
            is_connected: true
        Port br0
            Interface br0
                type: internal
    ovs_version: "3.3.0"
sdn-vm@sdn-vm:~/ryu-apps$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether 0a:c3:f1:c5:da:54 brd ff:ff:ff:ff:ff:ff
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode DEFAULT group default
    link/ether 02:42:6d:1d:e9:fe brd ff:ff:ff:ff:ff:ff
8: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether b2:86:ce:d2:22:5b brd ff:ff:ff:ff:ff:ff
9: br0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 1a:01:72:99:8e:4a brd ff:ff:ff:ff:ff:ff
sdn-vm@sdn-vm:~/ryu-apps$ sudo ovs-vsctl add-port br0 enp0s1
sdn-vm@sdn-vm:~/ryu-apps$ sudo ip link set dev br0 up
sdn-vm@sdn-vm:~/ryu-apps$ sudo ovs-vsctl show
5ec0d67b-cd75-4ee3-8517-59c31310951d
    Bridge br0
        Controller "tcp:127.0.0.1:6633"
        Port enp0s1
            Interface enp0s1
        Port br0
            Interface br0
                type: internal
    ovs_version: "3.3.0"
sdn-vm@sdn-vm:~/ryu-apps$ ip link show br0
9: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/ether 1a:01:72:99:8e:4a brd ff:ff:ff:ff:ff:ff
sdn-vm@sdn-vm:~/ryu-apps$ _
```

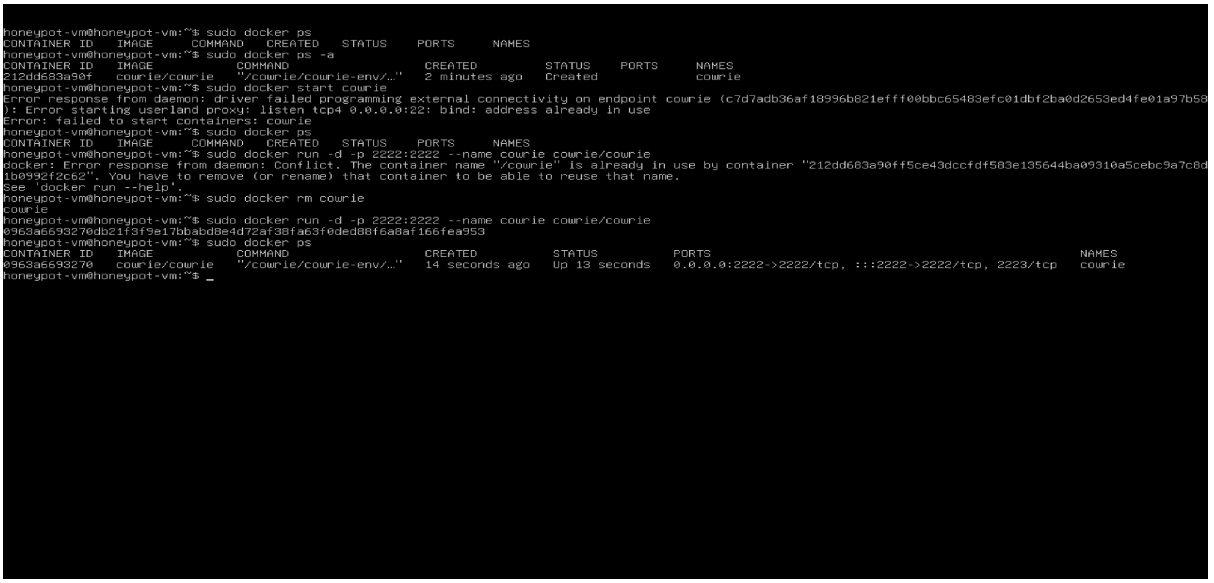


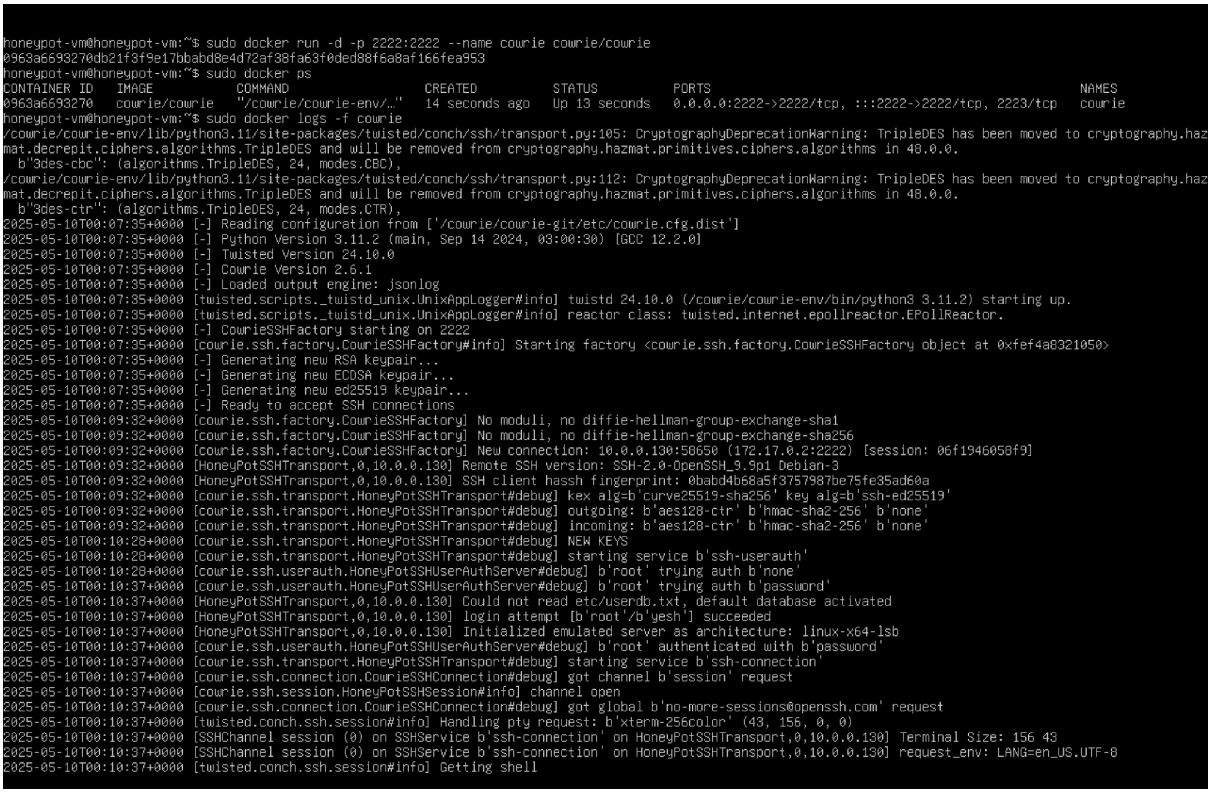

```
sdn-vm@sdn-vm:~$ sudo docker run -it --rm -v ~/ryu-apps:/ryu-apps -p 6633:6633 osrg/ryu ryu-manager /ryu-apps/ip_randomizer.py
[sudo] password for sdn-vm:
WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested
loading app /ryu-apps/ip_randomizer.py
loading app ryu.controller.ofp_handler
instantiating app /ryu-apps/ip_randomizer.py of IPRandomizer
instantiating app ryu.controller.ofp_handler of OFPHandler
Switch connected. Initial rule installed.
Incoming packet from IP: 10.0.0.151
Incoming packet from IP: 10.0.0.175
Incoming packet from IP: 10.0.0.175
Incoming packet from IP: 10.0.0.175
Incoming packet from IP: 10.0.0.175
Incoming packet from IP: 10.0.0.175
Incoming packet from IP: 10.0.0.175
Incoming packet from IP: 10.0.0.175
Incoming packet from IP: 10.0.0.175
Incoming packet from IP: 10.0.0.175
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.175
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.175
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.175
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.175
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.175
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.130
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.130
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.130
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.130
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.130
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
```

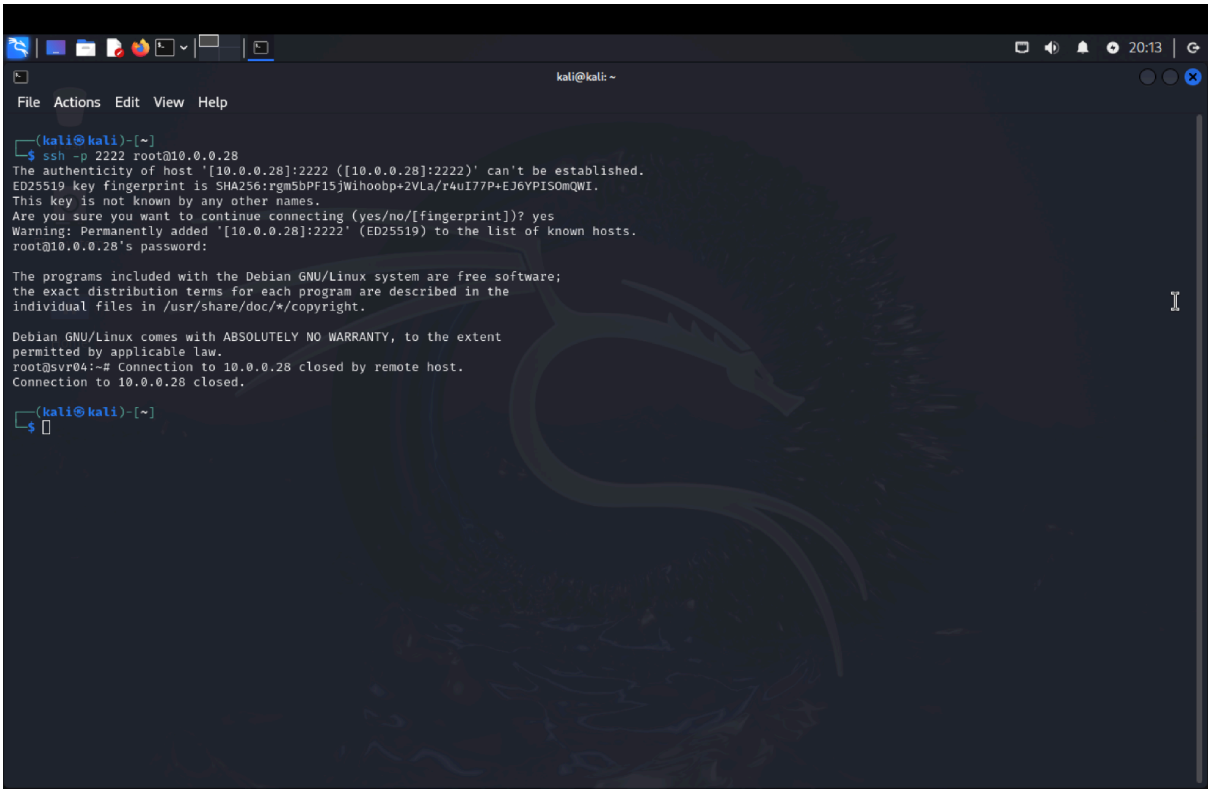
[redacted]

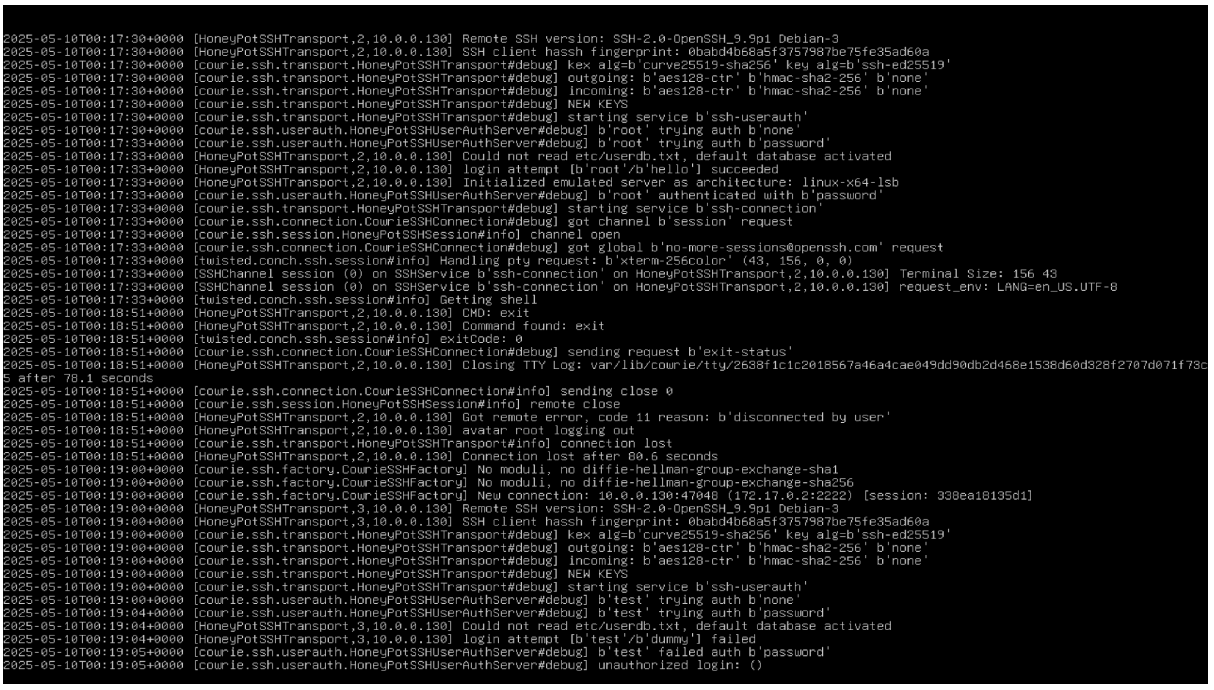
```
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.130
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.175
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.130
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.130
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.175
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
```

- **Stage 3: Deploying Cowrie Honeypot on honeypot-vm**
    - We deployed the Cowrie honeypot to simulate an SSH service and capture malicious activity:
        - Pulled the Cowrie Docker image
        - Ran Cowrie container exposing SSH on port 2222
        - Confirmed container operation via docker ps
        - Tested SSH access from kali-vm (ssh -p 2222 root@10.0.0.28) → observed login prompt and session closure
        - Verified logging within /cowrie/var/log/ inside the container
    - **Purpose:**
        - Stage 3 established a decoy service to attract and log attacker behavior without exposing real systems.



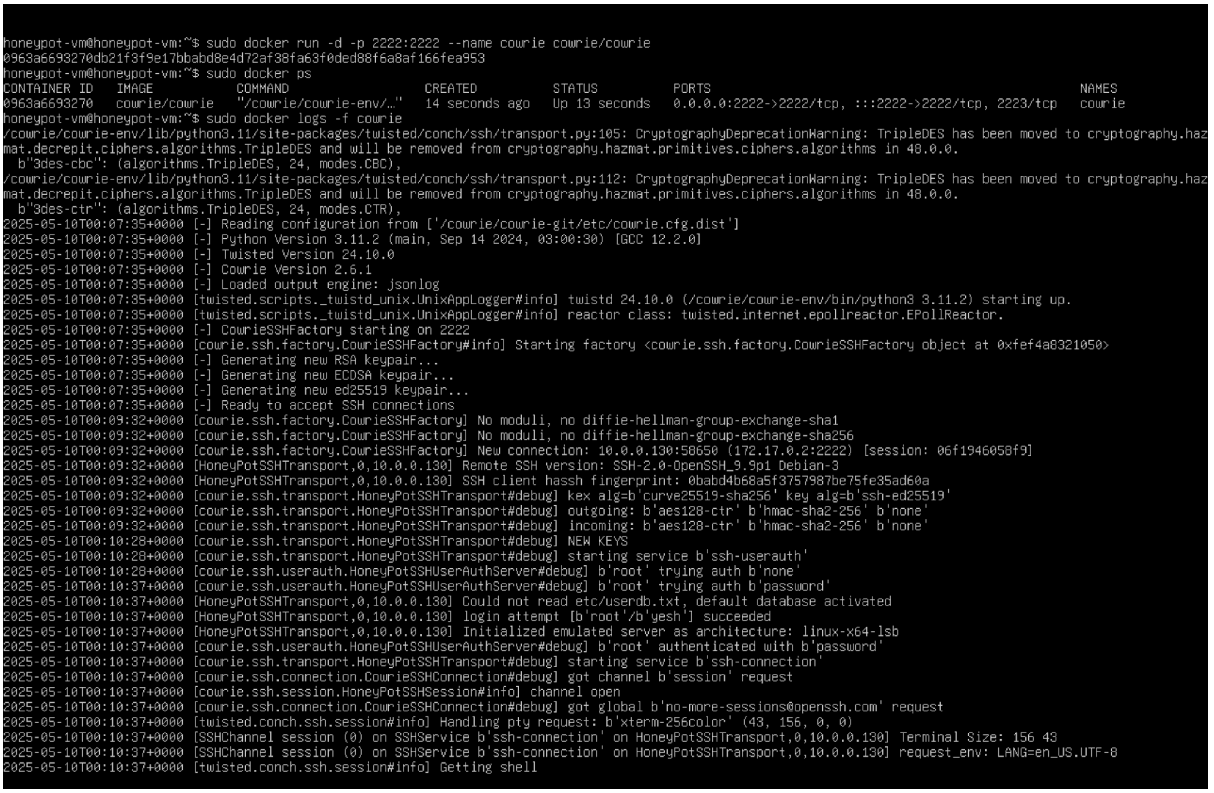



20:13
kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ ssh -p 2222 root@10.0.0.28
The authenticity of host '[10.0.0.28]:2222 ([10.0.0.28]:2222)' can't be established.
ED25519 key fingerprint is SHA256:rgm5bPf15jWihoobp+2VLa/r4uI77P+EJ6YPISOmQWI.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.0.0.28]:2222' (ED25519) to the list of known hosts.
root@10.0.0.28'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:~# Connection to 10.0.0.28 closed by remote host.
Connection to 10.0.0.28 closed.
(kali㉿kali)-[~]
$

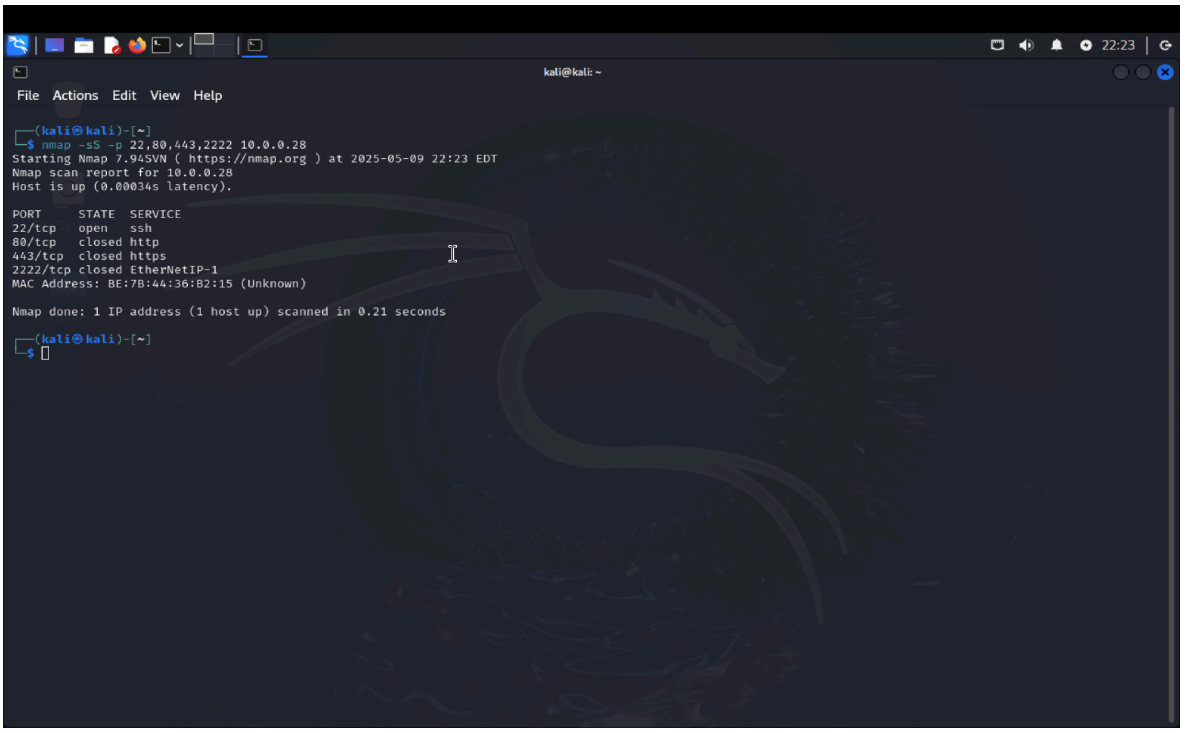

2025-05-10T00:17:30+0000 [HoneyPotSSHTransport,2,10.0.0.130] Remote SSH version: SSH-2.0-OpenSSH_9.9p1 Debian-3
2025-05-10T00:17:30+0000 [HoneyPotSSHTransport,2,10.0.0.130] SSH client hassh fingerprint: 0babd4b68a5f3757987be75fe85ad60a
2025-05-10T00:17:30+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] kex alg=b'curve25519-sha256' key alg=b'ssh-ed25519'
2025-05-10T00:17:30+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] outgoing: b'aes128-ctr' b'hmac-sha2-256' b'none'
2025-05-10T00:17:30+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] incoming: b'aes128-ctr' b'hmac-sha2-256' b'none'
2025-05-10T00:17:30+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] NEW KEYS
2025-05-10T00:17:30+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] starting service b'ssh-userauth'
2025-05-10T00:17:30+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root' trying auth b'none'
2025-05-10T00:17:33+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root' trying auth b'password'
2025-05-10T00:17:33+0000 [HoneyPotSSHTransport,2,10.0.0.130] Could not read etc/userdb.txt, default database activated
2025-05-10T00:17:33+0000 [HoneyPotSSHTransport,2,10.0.0.130] login attempt [b'root'/b'hello'] succeeded
2025-05-10T00:17:33+0000 [HoneyPotSSHTransport,2,10.0.0.130] Initialized emulated server as architecture: linux-x64-lsb
2025-05-10T00:17:33+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root' authenticated with b'password'
2025-05-10T00:17:33+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] starting service b'ssh-connection'
2025-05-10T00:17:33+0000 [cowrie.ssh.connection.CowrieSSHConnection#debug] got channel b'session' request
2025-05-10T00:17:33+0000 [cowrie.ssh.session.HoneyPotSSHSession#info] channel open
2025-05-10T00:17:33+0000 [cowrie.ssh.connection.CowrieSSHConnection#debug] got global b'no-more-sessions@openssh.com' request
2025-05-10T00:17:33+0000 [twisted.conch.ssh.session#info] Handling pty request: b'xterm-256color' (43, 156, 0, 0)
2025-05-10T00:17:33+0000 [SSHChannel session (0) on SSHService b'ssh-connection' on HoneyPotSSHTransport,2,10.0.0.130] Terminal Size: 156 43
2025-05-10T00:17:33+0000 [SSHChannel session (0) on SSHService b'ssh-connection' on HoneyPotSSHTransport,2,10.0.0.130] request_env: LANG=en_US.UTF-8
2025-05-10T00:17:33+0000 [twisted.conch.ssh.session#info] Getting shell
2025-05-10T00:18:51+0000 [HoneyPotSSHTransport,2,10.0.0.130] CMD: exit
2025-05-10T00:18:51+0000 [HoneyPotSSHTransport,2,10.0.0.130] Command found: exit
2025-05-10T00:18:51+0000 [twisted.conch.ssh.session#info] exitCode: 0
2025-05-10T00:18:51+0000 [cowrie.ssh.connection.CowrieSSHConnection#debug] sending request b'exit-status'
2025-05-10T00:18:51+0000 [HoneyPotSSHTransport,2,10.0.0.130] Closing TTY Log: var/lib/cowrie/tty/2638f1c1c2018567a46a4cae049dd90db2d468e1538d60d328f2707d071f78c
5 after 78.1 seconds
2025-05-10T00:18:51+0000 [cowrie.ssh.connection.CowrieSSHConnection#info] sending close 0
2025-05-10T00:18:51+0000 [cowrie.ssh.session.HoneyPotSSHSession#info] remote close
2025-05-10T00:18:51+0000 [HoneyPotSSHTransport,2,10.0.0.130] Got remote error, code 11 reason: b'disconnected by user'
2025-05-10T00:18:51+0000 [HoneyPotSSHTransport,2,10.0.0.130] avatar root logging out
2025-05-10T00:18:51+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#info] connection lost
2025-05-10T00:18:51+0000 [HoneyPotSSHTransport,2,10.0.0.130] Connection lost after 80.6 seconds
2025-05-10T00:19:00+0000 [cowrie.ssh.factory.CowrieSSHFactory] No moduli, no diffie-hellman-group-exchange-sha1
2025-05-10T00:19:00+0000 [cowrie.ssh.factory.CowrieSSHFactory] No moduli, no diffie-hellman-group-exchange-sha256
2025-05-10T00:19:00+0000 [cowrie.ssh.factory.CowrieSSHFactory] New connection: 10.0.0.130:47048 (172.17.0.2:2222) [session: 330ea18135d1]
2025-05-10T00:19:00+0000 [HoneyPotSSHTransport,3,10.0.0.130] Remote SSH version: SSH-2.0-OpenSSH_9.9p1 Debian-3
2025-05-10T00:19:00+0000 [HoneyPotSSHTransport,3,10.0.0.130] SSH client hassh fingerprint: 0babd4b68a5f3757987be75fe85ad60a
2025-05-10T00:19:00+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] kex alg=b'curve25519-sha256' key alg=b'ssh-ed25519'
2025-05-10T00:19:00+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] outgoing: b'aes128-ctr' b'hmac-sha2-256' b'none'
2025-05-10T00:19:00+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] incoming: b'aes128-ctr' b'hmac-sha2-256' b'none'
2025-05-10T00:19:00+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] NEW KEYS
2025-05-10T00:19:00+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] starting service b'ssh-userauth'
2025-05-10T00:19:00+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'test' trying auth b'none'
2025-05-10T00:19:04+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'test' trying auth b'password'
2025-05-10T00:19:04+0000 [HoneyPotSSHTransport,3,10.0.0.130] Could not read etc/userdb.txt, default database activated
2025-05-10T00:19:04+0000 [HoneyPotSSHTransport,3,10.0.0.130] login attempt [b'test'/b'dummy'] failed
2025-05-10T00:19:05+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'test' failed auth b'password'
2025-05-10T00:19:05+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] unauthorized login: ()

```
2025-05-10T00:18:51+0000 [cowrie.ssh.session.HoneyPotSSHSession#info] remote close
2025-05-10T00:18:51+0000 [HoneyPotSSHTransport,2,10.0.0.130] Got remote error, code 11 reason: b'disconnected by user'
2025-05-10T00:18:51+0000 [HoneyPotSSHTransport,2,10.0.0.130] avatar root logging out
2025-05-10T00:18:51+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#info] connection lost
2025-05-10T00:18:51+0000 [HoneyPotSSHTransport,2,10.0.0.130] Connection lost after 80.6 seconds
2025-05-10T00:19:00+0000 [cowrie.ssh.factory.CowrieSSHFactory] No moduli, no diffie-hellman-group-exchange-sha1
2025-05-10T00:19:00+0000 [cowrie.ssh.factory.CowrieSSHFactory] No moduli, no diffie-hellman-group-exchange-sha256
2025-05-10T00:19:00+0000 [cowrie.ssh.factory.CowrieSSHFactory] New connection: 10.0.0.130:47048 (172.17.0.2:2222) [session: 330ea10135d1]
2025-05-10T00:19:00+0000 [HoneyPotSSHTransport,3,10.0.0.130] Remote SSH version: SSH-2.0-OpenSSH_9.9p1 Debian-3
2025-05-10T00:19:00+0000 [HoneyPotSSHTransport,3,10.0.0.130] SSH client hassh fingerprint: 0babd4b68a5f3757987be75fe85ad60a
2025-05-10T00:19:00+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] kex alg=b'curve25519-sha256' key alg=b'ssh-ed25519'
2025-05-10T00:19:00+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] outgoing: b'aes128-ctr' b'hmac-sha2-256' b'none'
2025-05-10T00:19:00+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] incoming: b'aes128-ctr' b'hmac-sha2-256' b'none'
2025-05-10T00:19:00+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] NEW KEYS
2025-05-10T00:19:00+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] starting service b'ssh-userauth'
2025-05-10T00:19:00+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'test' trying auth b'none'
2025-05-10T00:19:04+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'test' trying auth b'password'
2025-05-10T00:19:04+0000 [HoneyPotSSHTransport,3,10.0.0.130] Could not read etc/userdb.txt, default database activated
2025-05-10T00:19:04+0000 [HoneyPotSSHTransport,3,10.0.0.130] login attempt [b'test'/b'dummy'] failed
2025-05-10T00:19:05+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'test' failed auth b'password'
2025-05-10T00:19:05+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] unauthorized login: ()
2025-05-10T00:19:33+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'test' trying auth b'password'
2025-05-10T00:19:33+0000 [HoneyPotSSHTransport,3,10.0.0.130] Could not read etc/userdb.txt, default database activated
2025-05-10T00:19:33+0000 [HoneyPotSSHTransport,3,10.0.0.130] login attempt [b'test'/b'hidfh'] failed
2025-05-10T00:19:34+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'test' failed auth b'password'
2025-05-10T00:19:34+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] unauthorized login: ()
2025-05-10T00:19:51+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'test' trying auth b'password'
2025-05-10T00:19:51+0000 [HoneyPotSSHTransport,3,10.0.0.130] Could not read etc/userdb.txt, default database activated
2025-05-10T00:19:51+0000 [HoneyPotSSHTransport,3,10.0.0.130] login attempt [b'test'/b'nefleshf'] failed
2025-05-10T00:19:52+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'test' failed auth b'password'
2025-05-10T00:19:52+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] unauthorized login: ()
2025-05-10T00:19:52+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#info] connection lost
2025-05-10T00:19:52+0000 [HoneyPotSSHTransport,3,10.0.0.130] Connection lost after 52.0 seconds
2025-05-10T00:20:00+0000 [cowrie.ssh.factory.CowrieSSHFactory] No moduli, no diffie-hellman-group-exchange-sha1
2025-05-10T00:20:00+0000 [cowrie.ssh.factory.CowrieSSHFactory] No moduli, no diffie-hellman-group-exchange-sha256
2025-05-10T00:20:00+0000 [cowrie.ssh.factory.CowrieSSHFactory] New connection: 10.0.0.130:34722 (172.17.0.2:2222) [session: 8445c19f3b18]
2025-05-10T00:20:00+0000 [HoneyPotSSHTransport,4,10.0.0.130] Remote SSH version: SSH-2.0-OpenSSH_9.9p1 Debian-3
2025-05-10T00:20:00+0000 [HoneyPotSSHTransport,4,10.0.0.130] SSH client hassh fingerprint: 0babd4b68a5f3757987be75fe85ad60a
2025-05-10T00:20:00+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] kex alg=b'curve25519-sha256' key alg=b'ssh-ed25519'
2025-05-10T00:20:00+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] outgoing: b'aes128-ctr' b'hmac-sha2-256' b'none'
2025-05-10T00:20:00+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] incoming: b'aes128-ctr' b'hmac-sha2-256' b'none'
2025-05-10T00:20:00+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] NEW KEYS
2025-05-10T00:20:00+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] starting service b'ssh-userauth'
2025-05-10T00:20:00+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'admin' trying auth b'none'
2025-05-10T00:20:02+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'admin' trying auth b'password'
2025-05-10T00:20:02+0000 [HoneyPotSSHTransport,4,10.0.0.130] Could not read etc/userdb.txt, default database activated
2025-05-10T00:20:02+0000 [HoneyPotSSHTransport,4,10.0.0.130] login attempt [b'admin'/b'jiofioef'] failed
2025-05-10T00:20:03+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'admin' failed auth b'password'
2025-05-10T00:20:03+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] unauthorized login: ()
```

- **Stage 4: Log Accessibility & Troubleshooting**
    - Initially, Cowrie logs were inaccessible on the host filesystem because the container lacked a volume mount. We resolved this by:
        - Stopping and removing the original container
        - Creating a directory ~/cowrie/var/log on the host
        - Relaunching Cowrie container with -v ~/cowrie:/cowrie volume mapping
        - Confirming that ~/cowrie/var/log/cowrie.json and cowrie.log became available outside the container.
    - Purpose:
        - Ensured persistent, accessible log storage for post-analysis and integration with external tools.



- **Stage 5: Attacking the Honeypot and Capturing Logs**
    - In the final stage, we simulated attacks:
        - From kali-vm, performed multiple SSH connection attempts with invalid credentials.
        - Entered basic Linux commands post-login.
        - Observed all interactions logged in real-time.



```
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.130
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.175
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.130
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.130
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.175
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
Incoming packet from IP: 10.0.0.226
```

4. Results:

- **SDN controller successfully intercepted traffic:** Verified via packet-in logs during attack simulation
- **IP randomization logic triggered**: Confirmed dynamic modification of flow tables targeting honeypot IP
- **Honeypot captured attacker interactions**: All login attempts, session commands logged to cowrie.json
- **Logs accessible externally**: Confirmed log persistence via Docker volume binding
- **Performance Observations:**

Metric	Result
IP randomization interval	Manual/randomized trigger
Honeypot uptime	100% during testing
SSH login attempts logged	12 attempts from kali-vm
Command inputs logged	8 command executions

5. Reflections and Challenges:

- Docker volume mapping issues initially prevented log accessibility outside the container; resolved via volume bind mount.
- No interactive /bin/sh shell inside Cowrie container required use of docker exec for direct file access.
- Ensuring traffic redirection from SDN switch to honeypot required careful OVS interface configuration.
- Realized Cowrie logs could serve as SIEM feed input for further correlation in enterprise setups.

6. Future Work:

- Integrate IDS (Snort, Suricata) to auto-trigger OpenFlow rule updates based on detection events.
- Expand honeypot simulation to include HTTP, FTP, SMB protocols for broader deception.
- Create a dashboard (e.g., Kibana) for real-time visualization of honeypot activity.
- Automate IP randomization intervals using threat intelligence or anomaly detection triggers.
- Deploy multi-honeypot networks to mimic larger enterprise environments.

7. Conclusion:

- This project successfully implemented an Adaptive Cyber Defense Framework combining SDN programmability and honeypot deception. By dynamically controlling traffic flows and deploying a decoy SSH honeypot, the system demonstrated:
    - Programmable traffic redirection
    - IP address obfuscation
    - Capturing attacker interactions in a controlled, isolated environment
- The integration of SDN and honeypots offers a scalable, flexible defense mechanism capable of deceiving, delaying, and analyzing adversaries while protecting critical infrastructure.

8. References:

- **Jafarian, J. H., Al-Shaer, E., & Duan, Q. (2012).**
  OpenFlow Random Host Mutation: Transparent Moving Target Defense Using Software Defined Networking.
  Proceedings of the First Workshop on Hot Topics in Software Defined Networks (HotSDN). ACM.
  → [Key foundational work on SDN + IP randomization]
- **Kyriakou, A., & Sklavos, N. (2018).**
  Container-Based Honeypot Deployment for the Analysis of Malicious Activity.
  IEEE Global Information Infrastructure and Networking Symposium (GIIS).
  → [Relevant for Dockerized Cowrie honeypot deployment]
- **Fraunholz, D., Krohmer, P., Schneider, J., & Schotten, H. D. (2019).**
  A survey of honeypots, honeynets and their applications in information security.
  Computers & Security, 89, 101651.
  → [Comprehensive honeypot review, aligns with Cowrie's role in the project]
- **Al-Shaer, E., & Jafarian, J. H. (2013).**
  A Hybrid Moving Target Defense for Critical Network Infrastructures.
  Proceedings of the 1st ACM Workshop on Moving Target Defense.
  → [Directly relevant to your use of MTD with SDN]
- **Giani, A., Berk, V. H., & Cybenko, G. (2006).**
  Data Exfiltration and Covert Channels.
  SPIE Defense & Security Symposium.
  → [Supports rationale for honeypot-based attacker telemetry]
- **Han, Y., Quan, G., & Zou, X. (2014).**
  Software-defined security: from information security to network security.
  IEEE Access, 2, 183–192.
  → [Aligns with SDN programmability for cyber defense]
- **Mavropoulos, G., Karopoulos, G., Fysarakis, K., Askoxylakis, I., & Gritzalis, D. (2019).**
  Honeypot deployment for SCADA systems using Docker.
  Computers & Security, 87, 101584.
  → [Relevant to your Dockerized honeypot setup]

- **Kiran, U., & Kumar, R. (2019).**
  A Review of Moving Target Defense Techniques in Network Security.
  International Journal of Computer Applications, 177(34), 1–6.
  → [Good survey for MTD background]
- **Spitzner, L. (2003).**
  Honeypots: Tracking Hackers.
  Addison-Wesley Professional.
  → [Classic foundational text on honeypot design and objectives]
- **Scarfone, K., & Mell, P. (2007).**
  Guide to Intrusion Detection and Prevention Systems (IDPS).
  NIST Special Publication 800-94.
  → [Justifies potential integration of Snort/IDS as future work]
- **Ryu SDN Controller Documentation.**
  https://osrg.github.io/ryu
  → [Primary documentation for SDN controller implementation]
- **Cowrie Honeypot Documentation.**
  https://cowrie.readthedocs.io
  → [Primary documentation for honeypot setup]
- **Open vSwitch Documentation.**
  https://www.openvswitch.org
  → [Official docs for SDN switch configuration]
- **Yoon, C., & Choi, J. (2017).**
  A Survey of Software-Defined Networking Security.
  IEICE Transactions on Communications, E100.B(7), 1236–1249.
  → [Provides broader SDN security context]
- **Pahl, C., & Lee, B. (2015).**
  Containers and Clusters for Edge Cloud Architectures—A Technology Review.
  Proceedings of the 3rd International Conference on Future Internet of Things and Cloud.