

# 1. Introduction

With cyber threats constantly evolving, organizations and individuals must stay proactive in identifying and mitigating potential risks. One widely used technique in cybersecurity is the deployment of **honeypots** trap systems designed to lure attackers by mimicking legitimate network services. Honeypots allow security professionals to observe attacker behavior, analyze intrusion methods, and collect valuable data without risking real systems.

This project involves developing a **basic honeypot using Python**. It operates as a fake TCP service that listens on a specific port, logs incoming connections, and responds with a rejection message. The implementation is conducted in a **virtual Ubuntu environment**, allowing for safe testing without impacting production systems. The honeypot is tested using **Nmap** for scanning and **Netcat** for manual connections, simulating real attacker behavior. This project demonstrates how even a simple script can contribute to early threat detection and basic network monitoring.

## 2. Objectives

- To create a basic TCP honeypot using Python.
- To simulate a fake service that listens on a specific port and logs all incoming connection attempts.
- To detect and observe unauthorized access attempts in a controlled environment.
- To understand how attackers perform network scanning and probing.
- To provide a foundation for more advanced honeypot and intrusion detection development.

### 3. Key Components and Steps

#### Environment Setup

- Ubuntu virtual machine (for safety and isolation).
- Python 3 installed via the terminal using: `sudo apt install python3`.

#### Creating the Honeypot Script

- A file named `honeypot.py` is created using the Nano text editor.
- The script:
  - Listens on **TCP port 2222**.
  - Accepts incoming connections.
  - Logs the attacker's IP address and port.
  - Sends back the message "Access Denied" before closing the connection.

#### Running the Honeypot

- Executed using: `sudo python3 honeypot.py`.
- The script prints: Listening on port 2222... indicating it is active.

#### Testing with Nmap and Netcat

- **Nmap** is used to scan the network and identify live hosts and open ports:
  - `nmap -sn 192.168.1.0/24` for host discovery.
  - `nmap 192.168.1.78` for detailed scanning of the honeypot machine.
- **Netcat (nc)** is used to manually connect to the honeypot:
  - `nc 192.168.1.78 2222` to simulate attacker access.

#### Observations

- The honeypot detects and logs connection attempts.

- **Example:** Connection made from a Kali Linux machine (IP: 192.168.1.80) was successfully logged.
- The system effectively simulates a target, allowing you to observe basic reconnaissance behavior.

## 4. Tools Used

- **Python 3:** Used to write the honeypot script with the built-in socket module.
- **Ubuntu (VM):** Hosted the honeypot in a secure virtual environment.
- **Terminal:** For script execution and basic command-line operations.
- **Nmap:** Performed network and port scanning to simulate attacker behavior.
- **Netcat (nc):** Tested direct TCP connections to the honeypot.
- **Kali Linux:** Acted as the attacker machine for testing intrusion attempts.

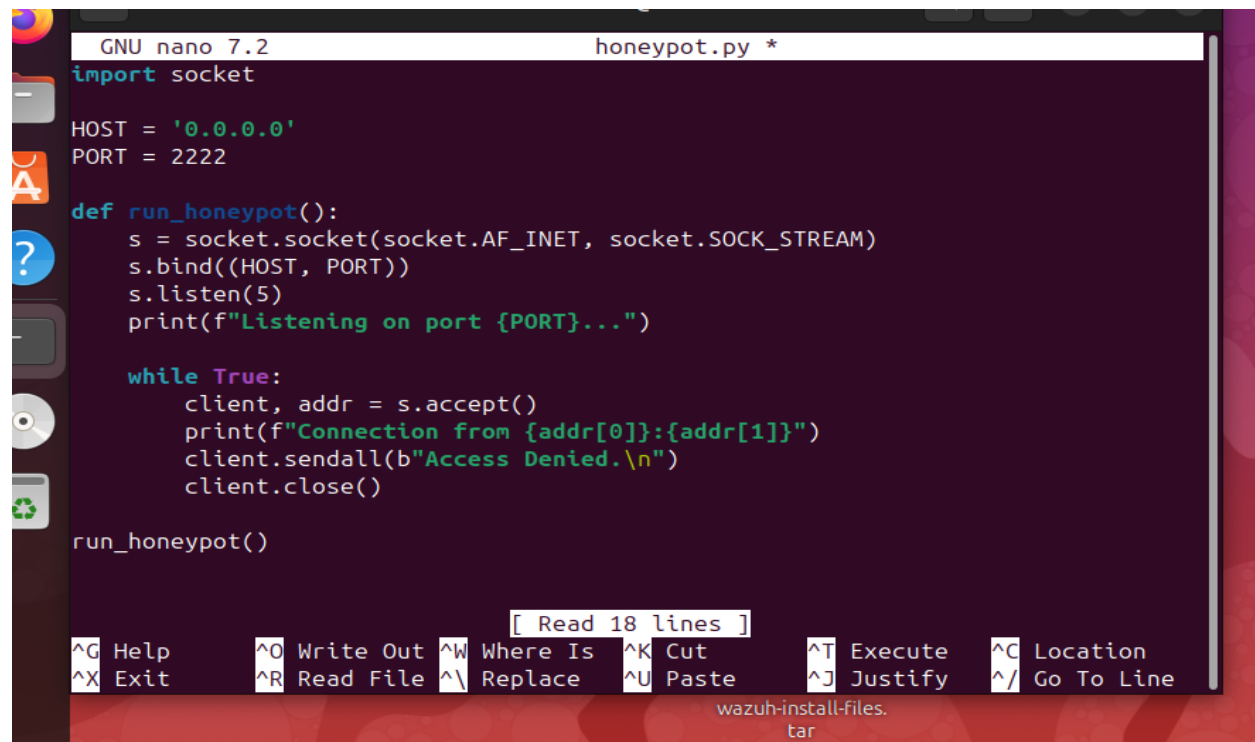
## 5.Demonstration

```
vboxuser@ubuntu:~$ sudo apt install python3
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3 is already the newest version (3.12.3-0ubuntu2).
```

This command is used to install Python 3 on an Ubuntu system using the APT (Advanced Package Tool) package manager.

```
vboxuser@ubuntu:~$ nano honeypot.py
```

Creating a Python file named honeypot.py using the Nano text editor in the terminal.



```
GNU nano 7.2 honeypot.py *
import socket

HOST = '0.0.0.0'
PORT = 2222

def run_honeypot():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind((HOST, PORT))
    s.listen(5)
    print(f"Listening on port {PORT}...")

    while True:
        client, addr = s.accept()
        print(f"Connection from {addr[0]}:{addr[1]}")
        client.sendall(b"Access Denied.\n")
        client.close()

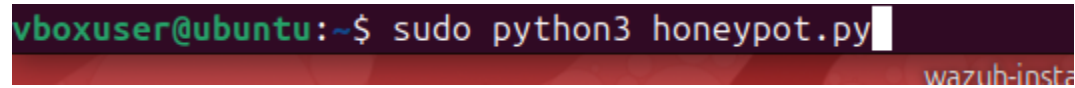
run_honeypot()
```

[ Read 18 lines ]

<b>^G</b> Help	<b>^O</b> Write Out	<b>^W</b> Where Is	<b>^K</b> Cut	<b>^T</b> Execute	<b>^C</b> Location
<b>^X</b> Exit	<b>^R</b> Read File	<b>^V</b> Replace	<b>^U</b> Paste	<b>^J</b> Justify	<b>^_</b> Go To Line

wazuh-install-files.  
tar

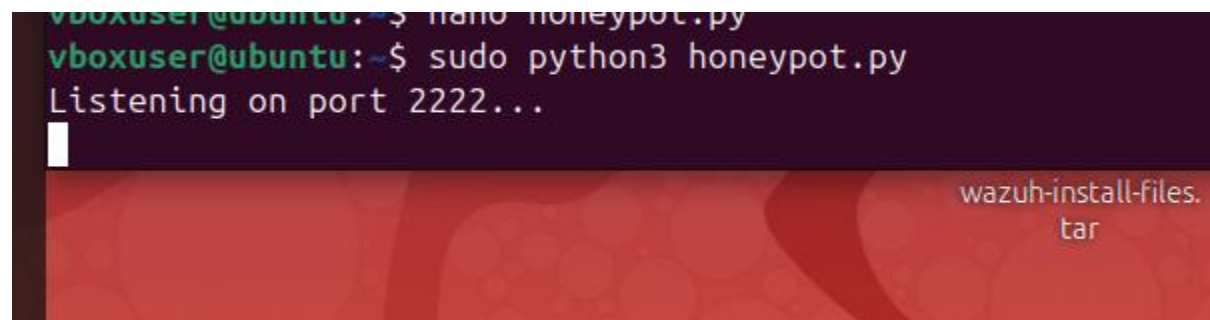
This Python script is a simple TCP honeypot. It listens to incoming connections on a specific port and logs every attempt to connect. It gives no real service but responds with "Access Denied", helping detect potential unauthorized scans or attacks.



```
vboxuser@ubuntu:~$ sudo python3 honeypot.py
```

A terminal window with a dark background. The prompt is 'vboxuser@ubuntu:~\$'. The command 'sudo python3 honeypot.py' is entered. A white cursor is at the end of the command. In the bottom right corner, there is a faint watermark that says 'wazuh-install-files.tar'.

The honeypot script is written in Python and is executed using the command `sudo python3 honeypot.py` on an Ubuntu virtual machine. When the script runs, it opens a fake service that listens on TCP port 2222 using a socket. This makes the system appear as if a real service is running on that port, which might attract attackers. The honeypot waits for any incoming connection. When a connection attempt is made, it captures the IP address and port of the source and prints it on the terminal. The script then responds to the connecting client with a simple message saying, "Access Denied" and closes the connection. This allows the system administrator or analyst to observe potential unauthorized access attempts without risking real system compromise.



```
vboxuser@ubuntu:~$ nano honeypot.py
vboxuser@ubuntu:~$ sudo python3 honeypot.py
Listening on port 2222...
```

A terminal window showing the execution of the script. The first line shows 'vboxuser@ubuntu:~\$ nano honeypot.py'. The second line shows 'vboxuser@ubuntu:~\$ sudo python3 honeypot.py'. The third line shows the output 'Listening on port 2222...'. A white cursor is at the end of the output line. In the bottom right corner, there is a faint watermark that says 'wazuh-install-files.tar'.

When the user runs the command `sudo python3 honeypot.py`, it starts the honeypot script with administrative privileges. The script then activates and begins listening for incoming connections on port 2222. The message `Listening on port 2222...` confirms that the honeypot is now active and waiting for any system or attacker to try and connect. This helps monitor and capture unauthorized access attempts in a safe and controlled environment.

```

(root@vbox)-[/home/kali]
# nmap -sn 192.168.1.0/24
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-06 03:26 EDT
Nmap scan report for 192.168.1.1
Host is up (0.00057s latency).
MAC Address: 28:C5:D2:EC:7F:AC (Intel Corporate)
Nmap scan report for 192.168.1.64
Host is up (0.35s latency).
MAC Address: 7A:4D:44:EA:7C:32 (Unknown)
Nmap scan report for 192.168.1.68
Host is up (0.14s latency).
MAC Address: A2:E0:D8:BB:57:90 (Unknown)
Nmap scan report for 192.168.1.69
Host is up (0.18s latency).
MAC Address: 36:62:11:95:97:40 (Unknown)
Nmap scan report for 192.168.1.74
Host is up (0.15s latency).
MAC Address: 86:5A:4B:F7:2C:DD (Unknown)
Nmap scan report for 192.168.1.78
Host is up (0.00031s latency).
MAC Address: 08:00:27:7A:A9:8C (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.1.104
Host is up (0.000095s latency).
MAC Address: 28:C5:D2:EC:7F:AC (Intel Corporate)
Nmap scan report for 192.168.1.110
Host is up (0.14s latency).
MAC Address: D2:F2:EF:1A:C0:81 (Unknown)
Nmap scan report for 192.168.1.130
Host is up (0.23s latency).
MAC Address: E8:BF:B8:D1:6B:68 (Intel Corporate)
Nmap scan report for 192.168.1.212
Host is up (0.19s latency).
MAC Address: 34:60:F9:22:9F:F2 (TP-Link Limited)
Nmap scan report for dsldevice.lan (192.168.1.254)
Host is up (0.20s latency).
MAC Address: 54:37:BB:BA:57:40 (Taicang T&W Electronics)
Nmap scan report for 192.168.1.80
Host is up.
Nmap done: 256 IP addresses (12 hosts up) scanned in 8.89 seconds

```

The command `nmap -sn 192.168.1.0/24` is used to perform a network discovery scan on the subnet 192.168.1.0/24, which includes all IP addresses from 192.168.1.1 to 192.168.1.254. The `-sn` option tells Nmap to only discover which hosts are up (i.e., live) on the network, without performing a port scan.

```

Nmap done: 1 IP address (1 host up) scanned in 0.32 seconds

(root@vbox)-[/home/kali]
# nmap 192.168.1.78
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-06 03:29 EDT
Nmap scan report for 192.168.1.78
Host is up (0.00047s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
443/tcp   open  https
2222/tcp  open  EtherNetIP-1
MAC Address: 08:00:27:7A:A9:8C (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.32 seconds

(root@vbox)-[/home/kali]

```

The result of running the `nmap 192.168.1.78` command, which scans the specific IP address 192.168.1.78 to check for open ports and services.

```
SyntaxError: unexpected character after line continuation character
vboxuser@ubuntu:~$ nano honeypot.py
vboxuser@ubuntu:~$ sudo python3 honeypot.py
Listening on port 2222...
Connection from 192.168.1.80:43182
Connection from 192.168.1.80:43194
Connection from 192.168.1.80:43204
Connection from 192.168.1.80:43206
Connection from 192.168.1.80:43210
Connection from 192.168.1.80:43216
Connection from 192.168.1.80:43224
Connection from 192.168.1.80:43232
Connection from 192.168.1.80:43246
Connection from 192.168.1.80:43254
Connection from 192.168.1.80:43270
Connection from 192.168.1.80:43274
Connection from 192.168.1.80:43282
Connection from 192.168.1.80:43298
Connection from 192.168.1.80:43310
Connection from 192.168.1.80:43316
Connection from 192.168.1.80:43326
Connection from 192.168.1.80:43336
Connection from 192.168.1.80:43344
Connection from 192.168.1.80:43356
Connection from 192.168.1.80:43358
Connection from 192.168.1.80:43364
Connection from 192.168.1.80:43376
Connection from 192.168.1.80:43390
Connection from 192.168.1.80:43406
Connection from 192.168.1.80:43416
Connection from 192.168.1.80:43418
Connection from 192.168.1.80:43424
Connection from 192.168.1.80:43432
Connection from 192.168.1.80:43444
Connection from 192.168.1.80:557
```

This log demonstrates our Python-based TCP honeypot successfully detecting and recording multiple connection attempts from what appears to be a Kali Linux machine (IP: 192.168.1.80) conducting network reconnaissance.

```
(root@vbox)-[/home/kali]
# nc 192.168.1.78 2222
Access Denied.
```

When executing the command `nc 192.168.1.78 2222` from a Kali Linux, a network connection is initiated to the IP address 192.168.1.78 on port 2222. This port is being monitored by the honeypot script, which listens for incoming connections. Upon accepting the connection, the honeypot script is programmed to send an "Access Denied" message as a response to any connection attempt.

## 6. Conclusion

This Python-based honeypot demonstrates that even simple scripts can help detect unauthorized access attempts and provide insight into attacker behavior. It serves as a foundational step for implementing more advanced intrusion detection or deception systems in real-world environments.

## 7. Future Enhancements

- Add file logging to store connection attempts.
- Integrate with email or SMS alerts for real-time notifications.
- Use a more complex service emulation to attract deeper attacker engagement.
- Visualize attack data using dashboards like Kibana or Grafana.
- Deploy in a cloud environment to test real-world exposure.

## 8. Appendix

### How the Attacker Identified the Ubuntu Honeypot IP

While scanning the network from a Kali Linux machine using Nmap, the attacker detected several active IP addresses. One of them was 192.168.1.78. Although the device did not reveal a hostname or detailed service information, several clues helped the attacker identify it as the Ubuntu honeypot:

#### 1. MAC Address Vendor Match



The MAC address associated with 192.168.1.78 began with 08:00:27, which corresponds to **Oracle VirtualBox** network adapters. This confirmed that the device was a **virtual machine**, most likely one of the VMs the attacker knew were running.

## **2. OS Fingerprinting**

Using Nmap with OS detection (nmap -O 192.168.1.78), the attacker found that the device was running a **Linux operating system**. Since other known machines on the network were Windows-based or networking equipment, this Linux machine was likely the **Ubuntu VM**.

## **3. Custom Open Port (2222)**

A deeper scan using nmap -sV 192.168.1.78 showed that the device had **port 2222 open**, which is a **non-standard port** often used to disguise services or mislead attackers. This unusual port choice raised suspicion.