📄PROJECT REPORT TITLE :

HONEYPOT SERVER TO DETECT ATTACT PATTERNS

🔐 Cyber Security Mini Project
📅 Date: 15/07/2025
👤 Submitted By: Saurabh Rajendra Chavanke
🏫 Institute: Elevated LabS

1️⃣🧭Introduction

🛡️A Honeypot is a security mechanism that simulates vulnerable services to lure attaker and study their behavior.
🎯The goal is to collect intelligence on attack patterns without compromising real system.Goal of high-interaction honeypot is to gain root–or administrator level– access to the server and then monitor the attacker'S activity.

2️⃣🎯 Objective

📎Simulate fake SSH/FTP services
📩 Log attacker attempt and commands
🔍 Analays repeated intrusion patterns
🚫 Block threats using fail2ban
🌏Visualize attacker IP geolocation

3️⃣🧰 Tools & Technologies Used

🔧 **Tool**	🔍 **Purpose**
🐍 Python	Scripting & automation
🐮Cowrie	SSH/FTP honeypot emulation
🚫fail2ban	Auto IP blocking
🌐MaxMind GeoIP	IP Geolocation

4️⃣🧩 System Architecture

🌐Internet

↓

🚨Attacker IPs

↓

🗃️Log files + fail2ban Bans

↓

📈GeoIP + Data Analysis

5️⃣⚙️ Implemantation Steps

A) 🖥️ Deploy Honeypot on VM

— Installed Ubuntu on VirtualBox

— Set up Cowrie or custom python SSH server

— Enabled ports (22/21) for emulated services

B) 📩 Log Connection

—🌐IP Address

—💠Username tried

— 💻Command attempted

C) 🔍 Analyze Log Files

— Parsed logs with python script

— Detected brute-force patterns

— Counted top attacking IPs

D) 🚫 Block with fail2ban

— fail2ban setup to read logs
— Regex filters for Cowrie
— Auto ban via iptables

E) 🌏Visualize IP Geolocation

— Used GeoIP2 with IP logs
— created maps with folium

6️⃣📂 Sample Logs & Analysis

📊 Top 5 Attacking IPs:

🌐IP Address	🔁Attempts
102.22.34.55	48
185.234.123.10	33
182.75.65.20	25
196.52.20.18	19
203.0.113.77	17

7️⃣🗺️ IP Geolocation Map

🌏 Using folium, attacker IPs were plotted
🧭 Red markers = High threat
🟡 Yellow = Medium
🟢 Green = Low

🚀🔗 References

— 🐮 Cowrie: https://github.com/cowrie/cowrie
— 🔐 fail2ban: https://www.fail2ban.org
— 🌏 MaxMind: https://www.maxmind.com
— 📘 Python Docs: https://docs.python.org

# Honeypot Project

This is a python-based honeypot to detect attack patterns.

# Tools :

cowrie or custom python scripts, SSH/FTP emulation.

# Mini Guide :

A) Deploy honeypot on a VM.

B) Log connections, ips, attempted command.

C) Analize log file for repeated attempts.

D) use fail2ban to block real threats.

E) Visualize IP geolocation of attackers.

# Deliverables :

Running honeypot + detailed logs + visual attack reports.

# Features :

- Real-time attack logging

- IP Address blocking

- Log visualization

## How to Run :

1. clone the repo

2. Run: python honeypot.py

# Logs :

Attack logs saved in log/attacks.db

## AUTHOR ##

SAURABH CHAVANKE

Codes:

1. **Logger**

```
    import os
from datetime import datetime

log_dir = "logs"
log_file = os.path.join(log_dir, "honeypot.log")

os.makedirs(log_dir, exist_ok=True)

def log_attempt(ip, username, password):
    with open(log_file, "a") as f:
        log_entry = f"{datetime.now()} | IP: {ip} |
Username: {username} | Password: {password}\n"
        f.write(log_entry)
        print(f"[LOGGED] {log_entry.strip()}")
```

2. Blocker

```
from collections import defaultdict
 MAX_ATTEMPTS = 3
attempts = defaultdict(int)
blocked_ips = "blocked_ips.txt"

def is_blocked(ip):
    try:
        with open(blocked_ips, "r") as f:
            return ip in f.read()
    except FileNotFoundError:
        return False

def register_attempt(ip):
    attempts[ip] += 1
    if attempts[ip] >= MAX_ATTEMPTS:
        with open(blocked_ips, "a") as f:
            f.write(ip + "\n")
        return True
    return False
```

3. Honeypot

```
import socket

import threading

from logger import log_attempt

from blocker import is_blocked, register_attempt


HOST = '0.0.0.0'

PORT = 2222


def handle_client(client_socket, addr):

    ip = addr[0]

    if is_blocked(ip):

        print(f"[BLOCKED] Connection attempt from
blocked IP: {ip}")

        client_socket.close()

        return


    client_socket.send(b"Username: ")

    username =
client_socket.recv(1024).decode().strip()
```

```
    client_socket.send(b"Password: ")
    password =
client_socket.recv(1024).decode().strip()


    log_attempt(ip, username, password)
    blocked = register_attempt(ip)


    if blocked:
        client_socket.send(b"You are blocked!\n")
    else:
        client_socket.send(b"Access Denied.\n")


    client_socket.close()


def start_server():
    server = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
    server.bind((HOST, PORT))
    server.listen(5)
    print(f"[+] Honeypot running on port {PORT}")
    while True:

        client_socket, addr = server.accept()

        thread = 
threading.Thread(target=handle_client, 
args=(client_socket, addr))

        thread.start()



if _name_ == "_main_":

    start_server()
```

4.Geo_visualizer

```
import matplotlib.pyplot as plt

from geopy.geocoders import Nominatim

import time
```

```
def get_location(ip):

    geolocator = Nominatim(user_agent="honeypot")

    try:

        location = geolocator.geocode(ip)

        return location.latitude,
location.longitude

    except:

        return None, None


def visualize_blocked_ips():

    ips = []

    with open("blocked_ips.txt", "r") as f:

        ips = [line.strip() for line in
f.readlines()]


    latitudes, longitudes = [], []


    for ip in ips:

        lat, lon = get_location(ip)

        if lat and lon:
```

```
        latitudes.append(lat)

        longitudes.append(lon)

    time.sleep(1)


if latitudes:

    plt.scatter(longitudes, latitudes)

    plt.title("Blocked IPs - Geolocation Map")

    plt.xlabel("Longitude")

    plt.ylabel("Latitude")

    plt.grid(True)

    plt.show()

else:

    print("No valid IP locations found.")
```