A Project Report

on

HomeNet-HoneyPot

Submitted by

Divyansh Raj

BTech IT

Sem VI

Dr Akhilesh Das Gupta Institute of Professional Studies

In partial fulfilment for the Summer Internship

Scientific Analysis Group
February - May 2025

ACKNOWLEDGEMENT

This project would not have been completed without the help, support, comments, advice, cooperation, and coordination of various people.

I acknowledge and express my deepest sense of gratitude of my internal supervisor **Mr. Aditya Raj** for his/her constant support, guidance, and continuous engagement. I highly appreciate his technical comments, suggestions, and criticism during the progress of the project.

I owe my profound gratitude to **Ashish Sonkar, Scientist E**, for his valuable guidance and facilitating me during my work. I am truly grateful for his mentorship and encouragement, which have played a pivotal role in shaping my understanding and approach to the subject matter.

Finally, I extend my heartfelt appreciation to other scientists and staff for their precious support and cooperation during the development of this project.

Sincerely,
Divyansh Raj

ABSTRACT

In the evolving landscape of cybersecurity threats, residential networks have become increasingly vulnerable to intrusion attempts, especially with the rise of IoT devices and smart home technologies. This project, **HomeNet Honeypot**, addresses the need for proactive defense mechanisms by deploying a software-based honeypot system tailored for home Wi-Fi environments.

Built using the Cowrie honeypot framework, the system emulates vulnerable services to attract and log unauthorized access attempts, thereby acting as a decoy to study attacker behavior without risking real assets. The captured data is analyzed and visualized through a custom-built web dashboard developed using Flask and Chart.js, enabling real-time monitoring and historical insights into network threats.

HomeNet Honeypot provides a lightweight, scalable, and cost-effective solution for intrusion detection, aiding cybersecurity researchers and enthusiasts in understanding common attack vectors and patterns in home networks. This project contributes to the broader goal of democratizing cybersecurity awareness and defense capabilities at the user level.

INDEX

INTRODUCTION

In an era dominated by ubiquitous internet access and smart home technologies, residential networks have become prime targets for cyberattacks. With the proliferation of Internet of Things (IoT) devices, ranging from smart thermostats to voice assistants and surveillance cameras, the average home now hosts a multitude of endpoints, many of which lack robust security measures. Unlike enterprise systems which often employ layered security protocols and dedicated monitoring teams, home networks are typically undersecured and unmonitored, making them low-hanging fruit for malicious actors.

The motivation behind the "HomeNet Honeypot" project stems from this critical vulnerability. By developing a lightweight, software-based honeypot tailored specifically for home Wi-Fi environments, this project aims to empower everyday users with tools to detect, analyze, and understand unauthorized access attempts on their networks. Honeypots, traditionally used in enterprise and research environments, simulate vulnerable systems to attract attackers, thereby enabling the collection of valuable data on attack methods, tools, and behaviors. This project translates that concept to a residential context, aiming to bridge the gap between professional cybersecurity practices and consumer-level protection.

1.1 Purpose of the Report

The primary purpose of this report is to document the design, implementation, and evaluation of the HomeNet Honeypot project. It serves as both a technical record and a guide for future developers, researchers, and cybersecurity enthusiasts who wish to understand or build upon this work. The report details the motivations, objectives, and practical considerations that informed the project, along with a comprehensive description of the methodology adopted. It also explores the challenges encountered during development, the tools and technologies utilized, and the results obtained from testing the system in a real-world setting.

Furthermore, this report aims to provide insights into the nature and frequency of cyber threats targeting residential networks. By analyzing data captured by the honeypot, we can identify common attack patterns and potentially flag emerging trends. This can contribute not only to personal network defense strategies but also to the broader cybersecurity community's understanding of home network vulnerabilities.

1.2 Scope

The scope of this project is deliberately constrained to the home Wi-Fi environment, focusing on threats that typically target small-scale residential setups. Unlike enterprise or data center security solutions, HomeNet Honeypot is designed to be resource-efficient, easy to deploy, and accessible to users with minimal technical expertise.

OBJECTIVE

2.1 The key objectives of the project include:

1. **Developing a Custom Honeypot System**: To design and implement a honeypot using the Cowrie framework that mimics commonly targeted services like SSH and Telnet on a home network.
2. **Capturing and Logging Attack Data**: To collect detailed logs of intrusion attempts, including IP addresses, login credentials, command histories, and session durations.
3. **Building a Real-Time Monitoring Dashboard**: To develop a user-friendly web interface using Flask and Chart.js that visualizes honeypot activity and trends over time.
4. **Analyzing Attack Patterns**: To interpret the collected data to identify common attack strategies, source geographies, and tools used by attackers.
5. **Promoting Cyber Security Awareness**: To create documentation and educational resources that help users understand how honeypots work and how they can improve home network security.
6. **Ensuring Scalability and Adaptability**: To ensure the system can be easily modified or extended to accommodate new attack vectors or service emulations.

2.2 Methodology

The development of the HomeNet Honeypot system followed a structured methodology comprising several stages: requirement analysis, design and tool selection, implementation, testing, and evaluation.

1. Requirement Analysis

The first phase involved understanding the unique security challenges associated with home networks. This included:

- Reviewing existing literature and threat reports on residential network intrusions
- Identifying the most commonly exploited services and vulnerabilities in home environments
- Understanding user constraints such as hardware limitations and low technical expertise

Based on this analysis, Cowrie was chosen as the core honeypot platform due to its ability to emulate SSH and Telnet services convincingly while offering extensive logging features.

2. System Design and Tool Selection

The honeypot architecture was designed with modularity and simplicity in mind. Key components include:

- **Cowrie Honeypot**: Simulates vulnerable SSH and Telnet services.
- **Logging Module**: Captures session logs, IP addresses, credentials, and commands.
- **Web Dashboard**: Built with Flask for backend logic and Chart.js for frontend visualization.
- **System Scheduler**: Cron jobs and Python scripts for periodic data extraction and dashboard updates.

Security and resource constraints were central to tool selection, ensuring that the system could run on low-power devices like a Raspberry Pi.

3. Implementation

Implementation began with setting up Cowrie on a virtual machine to simulate a real home network environment. Key configuration steps included:

- Redirecting SSH traffic to Cowrie using iptables rules
- Customizing Cowrie responses to resemble a generic home router
- Enhancing Cowrie's logging features with additional metadata such as geolocation based on IP

Parallelly, the Flask backend was developed to parse Cowrie log files and extract meaningful data points. This backend feeds into a frontend built with HTML, CSS, and Chart.js, providing users with an intuitive dashboard displaying metrics such as:

- Number of login attempts
- Most targeted usernames and passwords
- Geographical distribution of attackers
- Session durations and frequency of commands

4. Testing and Deployment

After implementation, the system was deployed on a Raspberry Pi connected to a home Wi-Fi network. Testing involved:

- Simulating intrusion attempts to validate logging and dashboard accuracy
- Monitoring CPU and memory usage to ensure performance efficiency
- Evaluating responsiveness of the dashboard to real-time data updates

This phase also included functional tests like database integrity checks, interface responsiveness, and log rotation scripts.

5. Data Collection and Analysis

Following deployment, the honeypot was run continuously for a period of four weeks. During this time, the system logged hundreds of intrusion attempts. The collected data was then analyzed to extract insights such as:

- Peak attack times and days
- Most frequently used passwords and usernames
- Tools and command patterns indicating automated scripts
- Regional hotspots of attack origins (via IP geolocation)

This analysis revealed significant trends, including the prevalence of dictionary attacks using commonly leaked credentials and the repeated targeting of default home router login combinations.

6. Documentation and Dissemination

The final stage involved creating comprehensive documentation including:

- User guides for setting up the honeypot
- Technical manuals explaining code structure and deployment steps
- Research briefs summarizing the analytical insights

These materials were intended to facilitate community engagement, allowing others to replicate or extend the project. In addition, the dashboard source code and configuration files were published on a public GitHub repository under an open-source license.

INTERNAL WORKING & DEMONSTRATION

3.1 Architecture Diagram

The architecture diagram illustrates the flow of data from the simulated attack interface (Cowrie) through the logging mechanisms to the data processing backend and visualization dashboard. It shows how different modules like Cowrie, the log parser and Flask-Charts frontend are interconnected.

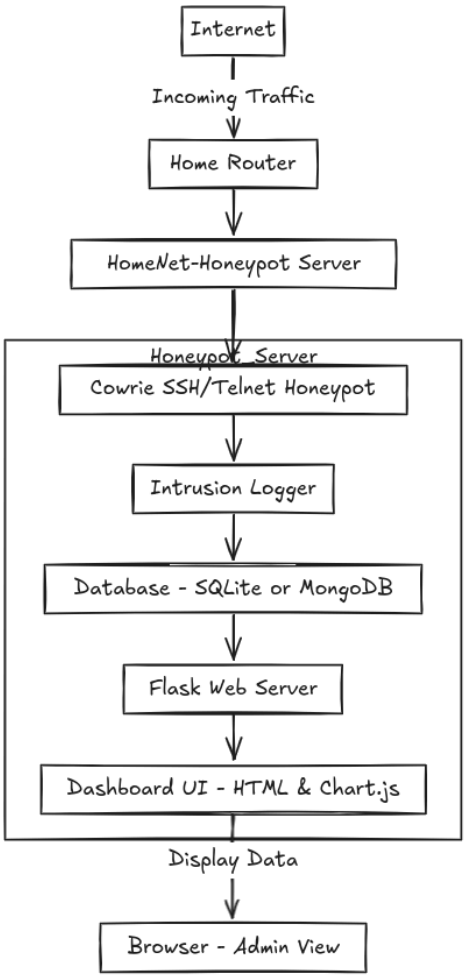


figure 3.1 flowchart of attack simulation

3.2 System Flow and Behavior

The system behavior can be summarized as follows:

- This redirects all SSH traffic to the Cowrie honeypot.
- Cowrie logs every connection attempt and stores detailed interaction logs.
- A Python script parses these logs periodically.
- The Flask backend reads data from the database and formats it for the dashboard.
- Chart.js dynamically renders the graphs on the frontend.

3.3 Project Setup Steps

It is better to set up this project on a linux device with at least 4gb RAM.

(a) Install Cowrie Honeypot on your device

```
sudo apt update && sudo apt install -y git python3-venv
git clone https://github.com/cowrie/cowrie.git

cd cowrie
python3 -m venv cowrie-env
source cowrie-env/bin/activate
pip install -r requirements.txt

cp etc/cowrie.cfg.dist etc/cowrie.cfg
nano etc/cowrie.cfg  # (You can change ports or enable logging
here)
```

(b) Start Cowrie

```
bin/cowrie start
```

This process ensures that:

- Cowrie is isolated and portable (via virtualenv),
- Installed with all required dependencies,
- Configured safely and flexibly,
- And ready to detect fake SSH/Telnet intrusion attempts.
- Log Processing & Storage

Cowrie will start simulating SSH/Telnet services. Any login attempt is captured and stored in `var/log/cowrie/cowrie.log`.

(c) Log Processing & Storage

A Python script parses Cowrie logs (`cowrie.json`) and extracts IP addresses, commands attempted, and timestamps. Data is then stored in a JSON file for easy access by the dashboard.

```
# log_parser.py
import json
def parse_logs(filepath="cowrie/log/cowrie.json"):
      with open(filepath, 'r') as f:
      for line in f:
            log_entry = json.loads(line)
            if log_entry.get('eventid') == 'cowrie.session.connect':
                  print(f"Intrusion attempt from {log_entry['src_ip']}
at {log_entry['timestamp']}")
parse_logs()
```

Example logs include anonymized entries like:

```
10  [-] Generating new RSA keypair...
11  [-] Generating new ECDSA keypair...
12  [-] Generating new ed25519 keypair...
13  [-] Ready to accept SSH connections
14  [cowrie.ssh.factory.CowrieSSHFactory] No moduli, no diffie-hellman-group-exchange-sha1
15  [cowrie.ssh.factory.CowrieSSHFactory] No moduli, no diffie-hellman-group-exchange-sha256
16  [cowrie.ssh.factory.CowrieSSHFactory] New connection: 127.0.0.1:36936 (127.0.0.1:2222) [session: 3dfa1e
17  [HoneyPotSSHTransport,0,127.0.0.1] Remote SSH version: SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.1
18  [HoneyPotSSHTransport,0,127.0.0.1] SSH client hassh fingerprint: 78c05d999799066a2b4554ce7b1585a6
19  [cowrie.ssh.transport.HoneyPotSSHTransport#debug] kex alg=b'curve25519-sha256' key alg=b'ssh-ed25519'
20  [cowrie.ssh.transport.HoneyPotSSHTransport#debug] outgoing: b'aes128-ctr' b'hmac-sha2-256' b'none'
21  [cowrie.ssh.transport.HoneyPotSSHTransport#debug] incoming: b'aes128-ctr' b'hmac-sha2-256' b'none'
22  [cowrie.ssh.transport.HoneyPotSSHTransport#info] connection lost
23  [HoneyPotSSHTransport,0,127.0.0.1] Connection lost after 1.9 seconds
24  [cowrie.ssh.factory.CowrieSSHFactory] No moduli, no diffie-hellman-group-exchange-sha1
25  [cowrie.ssh.factory.CowrieSSHFactory] No moduli, no diffie-hellman-group-exchange-sha256
26  [cowrie.ssh.factory.CowrieSSHFactory] New connection: 127.0.0.1:42036 (127.0.0.1:2222) [session: fd3a3a
```

example for json logs:


```
{"eventid":"cowrie.client.version","version":"SSH-2.0-OpenSSH_9.0p1
Ubuntu-1ubuntu7.1","message":"Remote SSH version: SSH-2.0-OpenSSH_9.0p1
Ubuntu-1ubuntu7.1","sensor":"divyansh-VivoBook-ASUSLaptop","timestamp":"2
025-04-23T01:02:20.595907Z","src_ip":"127.0.0.1","session":"3dfa1e16c4e2"
}
```

(d) Flask Dashboard Setup (Web Interface)

The dashboard uses Flask as the backend and Chart.js on the frontend to plot intrusion trends.

Flask App:

```
# app.py
from flask import Flask, render_template, jsonify
import json
import os

app = Flask(__name__)

COWRIE_LOG_PATH = "../honeypots/cowrie/var/log/cowrie/cowrie.json"

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/logs")
def logs():
    if not os.path.exists(COWRIE_LOG_PATH):
        print("Loaded logs:")
        return jsonify([])
    with open(COWRIE_LOG_PATH) as f:
        data = [json.loads(line) for line in f if line.strip()]
    return jsonify(data)

if __name__ == "__main__":
    app.run(debug=True)
```

Frontend page:

```
fetch("/logs")
 .then((res) => res.json())
 .then((data) => {
   const ipCounts = {};
   data.forEach((log) => {
     const ip = log.src_ip || log.src_host;
     if (ip) ipCounts[ip] = (ipCounts[ip] || 0) + 1;
   });

   const labels = Object.keys(ipCounts);
   const values = Object.values(ipCounts);

   new Chart(document.getElementById("ipChart"), {
     type: "bar",
     data: {
       labels: labels,
       datasets: [
         {
           label: "Attack Count",
           data: values,
           backgroundColor: "rgba(75, 192, 192, 0.6)",
           borderColor: "rgb(6, 71, 71)",
         },
       ],
     },
   });
 });
```

(e) Demonstration

- Start Cowrie: `bin/cowrie start`
- Simulate an SSH login from same machine: `ssh root@localhost -p 2222`
- Cowrie logs the attempt.
- Run the Flask app: `python3 app.py`
- Open the dashboard on browser: `http://localhost:5000`
- Intrusion attempts appear on real-time bar charts.

3.4 Handling and Response

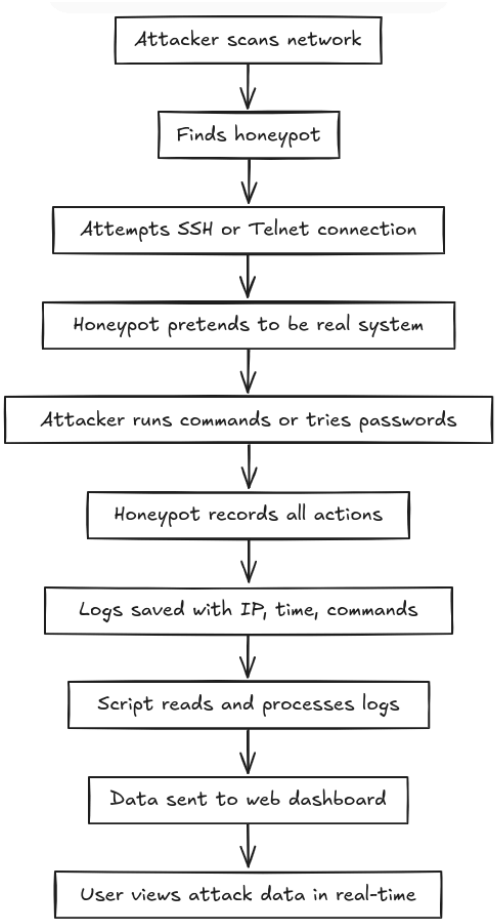


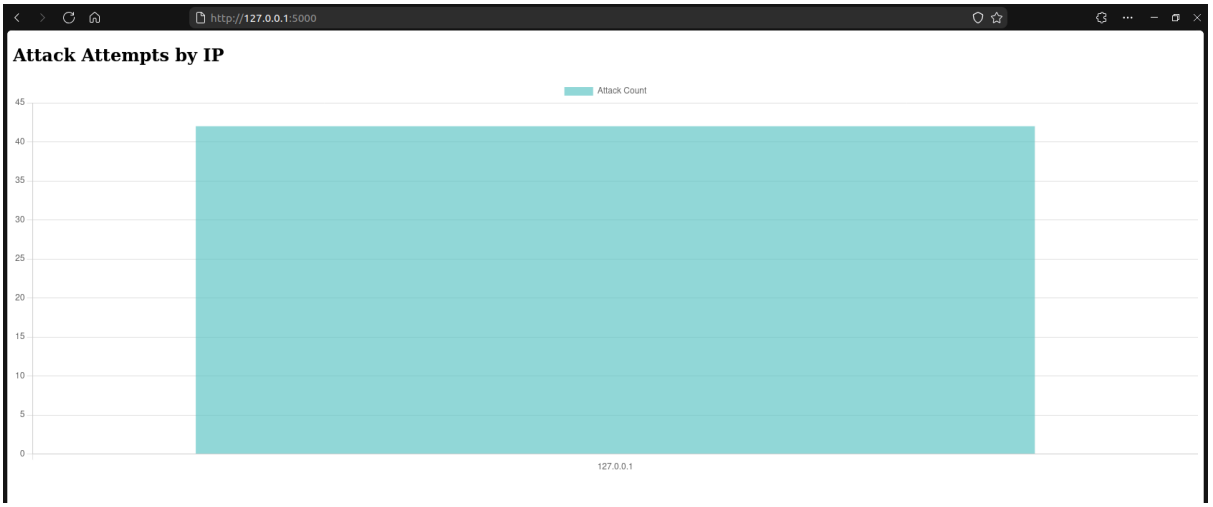
figure 3.2 Internal Working of project

Internal working when the attack happens:

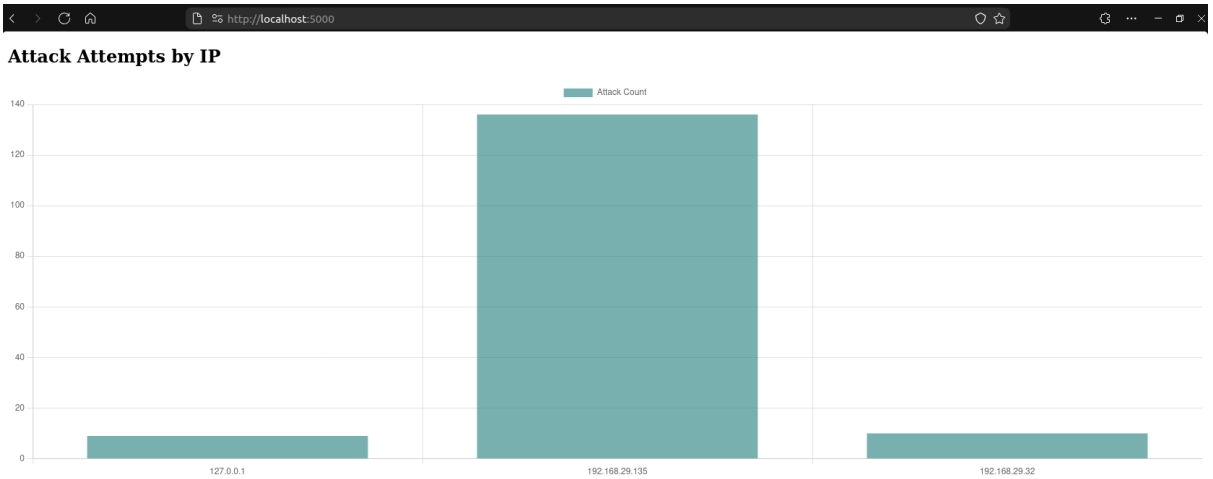
1. **Attacker finds your fake device** (the honeypot) on the network.
2. They try to **connect**—usually through SSH or Telnet.
3. The honeypot (e.g., Cowrie) **pretends to be a real system** and accepts the connection.
4. The attacker **tries passwords, runs commands, or uploads files**.
5. Every single action is **secretly recorded**—including:
    a. IP address
    b. Time of attack
    c. Commands they tried
6. This data is **saved in log files**.
7. A script **reads and processes** the logs.
8. The dashboard **displays this attack data** as charts and tables.
9. You can **monitor all of this in real time** through the dashboard.

PROJECT SNAPSHOTS

4.1 Bar Graph of one attacker:



4.2 Bar Graph of multiple attackers



CONCLUSION AND SCOPE

5.1 Conclusion

The HomeNet Honeypot project demonstrates the feasibility and value of deploying lightweight honeypots in residential environments. By mimicking commonly exploited services and capturing detailed logs, the system provides homeowners and researchers with crucial insights into the evolving threat landscape of home networks.

Beyond technical efficacy, the project also contributes to cybersecurity awareness by making advanced monitoring techniques accessible to non-expert users. The modular architecture and open-source nature of the project ensure adaptability and community-driven improvements.

5.2 Future Scope

Looking ahead, future iterations of HomeNet Honeypot could incorporate:

- **Email and SMS alerts** for real-time threat notifications
- **Support for additional protocols** such as HTTP, FTP, or SMB
- **Machine learning models** to classify attack behavior and flag anomalies
- **Multi-device synchronization** for homes with multiple access points

5.3 Raspberry Pi Implementation

To make HomeNet Honeypot more accessible and deployable, it can be implemented on a **Raspberry Pi**, providing a low-cost, portable, and plug-and-play intrusion detection solution. The steps include:

- Installing a lightweight OS like **Raspberry Pi OS**.
- Deploying honeypot software (e.g., **Cowrie** or a **custom Flask-based service**) as a systemd service or daemon.
- Configuring the honeypot to auto-start on boot for seamless operation.
- Connecting to the home network via Wi-Fi or Ethernet for continuous monitoring.
- Optionally adding a **status display** (e.g., OLED screen or LEDs) for quick threat-level indicators.

This approach enhances usability for non-technical users and allows easy deployment across different residential environments.

Integrating these features could further enhance the tool's usability and effectiveness, encouraging broader adoption and contributing to the collective cybersecurity defense of residential users.