Honeypot for Café Wi-Fi: A Decoy-Based Cybersecurity Model

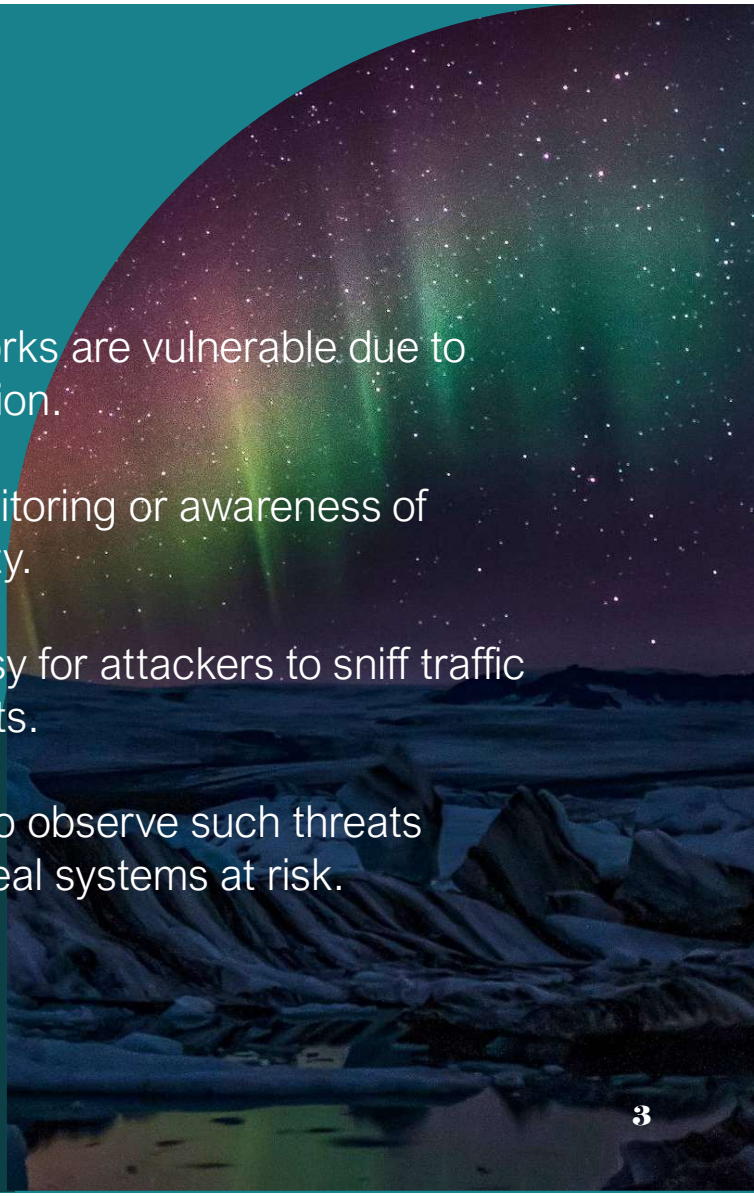
Presented by: Shaikh Muqtasida & Arya Gawit
Internship Program: DigiSuraksha Cybersecurity Internship

Introduction

- Cybersecurity is critical in public spaces like cafés.
- Open Wi-Fi is convenient, but it’s often not secure.
- Attackers can intercept traffic or scan for open devices.
- We explore honeypots as a way to detect and learn from such attacks.

Problem Statement

- Café Wi-Fi networks are vulnerable due to weak/no encryption.
- There's little monitoring or awareness of suspicious activity.
- This makes it easy for attackers to sniff traffic or scan for targets.
- We need a way to observe such threats without putting real systems at risk.



What is a Honeypot?

A honeypot is a fake computer system or service. It is designed to attract attackers. The system logs the attacker's actions for research or defense. It acts as a trap: attackers think it's real, but it’s only for monitoring.

Types of Honeypots

- **Low-interaction honeypots: Simulate services (e.g., fake web server):** Easier to build. Logs simple interactions. (Used in our project)
- **High-interaction honeypots: Simulate full systems (e.g., real OS):** Harder to build. Captures advanced attacker behavior.

Our focus: Low-interaction honeypot using Python.

Why Café Wi-Fi Needs Honeypots

- Public Wi-Fi is a frequent target for scanning, snooping, and phishing.
  - A honeypot can alert café admins to malicious activity.
- It helps researchers study common attack patterns in a safe way.
  - It is ethical, legal, and doesn't risk real user data.

Our Research Objective

- Create a simple honeypot that mimics a service on café Wi-Fi.
- Log attempted connections from potential attackers.
- Analyze logs to understand how attackers behave in public networks.
- Raise awareness and propose solutions.

Sample Honeypot Tool

honeypot.py

```python
import socket
import datetime

HOST = "0.0.0.0"   # Listen on all network interfaces
PORT = 8000        # Port to listen on

def log_connection(addr, data):
    timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    message = data.decode('utf-8', errors='ignore')[:100]
    log_entry = f"[{timestamp}] Connection from {addr[0]}:{addr[1]} - Data: {message}\n"
    print(log_entry.strip())

    with open("honeypot_log.txt", "a") as log_file:
        log_file.write(log_entry)

def run_honeypot():
    print(f"[INFO] Honeypot listening on port {PORT}...")
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_socket:
        server_socket.bind((HOST, PORT))
        server_socket.listen(5)
        while True:
            conn, addr = server_socket.accept()
            with conn:
                try:
                    data = conn.recv(1024)
                    if data:
                        log_connection(addr, data)
                        conn.sendall(b"Access Denied. This activity has been logged.\n")
                except Exception as e:
                    print(f"[ERROR] {e}")

if __name__ == "__main__":
    run_honeypot()
```

We built a honeypot using Python's socket module. It listens on port 8000 for any connection attempt. Logs include IP address, time, and request data. It's not a real server, just a fake one to attract scans.

Sample Output / Observation

Example log entry:

[2025-05-10 13:04:12] Connection from 127.0.0.1:52342 – Data: GET / HTTP/1.1...

What it tells us:

- Someone tried to connect to our fake service
- Could be a port scanner or curious user on the café Wi-Fi
- Logs help us detect unusual behavior before real damage

Sample Honeypot Tool

- A fake login page is deployed using Python Flask on port 8000.
- The page mimics a public café Wi-Fi login screen, asking for a username and password.
- When a user submits credentials, a fake “Access Denied” message is shown on the screen.
- The attempted username, password, IP address, and timestamp are logged in the terminal for analysis.
- This simulates an attacker or unauthorized user attempting to access network services.
- The system does not authenticate any user — it only captures behavioral data for passive monitoring.

```
aryagawit@Aryas-MacBook-Air honeypot-project % python3 app.py
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8000
 * Running on http://192.168.180.163:8000
Press CTRL+C to quit
127.0.0.1 - - [12/May/2025 15:11:47] "GET / HTTP/1.1" 200 -
[2025-05-12 15:12:20] IP: 127.0.0.1 | Username: arya | Password: abc
127.0.0.1 - - [12/May/2025 15:12:20] "POST / HTTP/1.1" 200 -
```

Market Relevance & Ethics

- Honeypots are used by companies, universities, and governments.
- Legal and ethical if used for detection (not attacking back).
- Can be deployed in small cafés or startups with limited security.
- Helps educate owners about basic cyber threats.

Future Scope

- Add real-time notifications for suspicious behavior
- Visual dashboard for monitoring
- Install honeypot on a Raspberry Pi and keep it running in a café
- Use AI to auto-analyze attack patterns

Thank you

GitHub link with research paper available on request:

Click here to access GitHub link

Tool Demo Video Link:

Click here for Video