Design and Implementation of a Python-Based SSH Honeypot

---

1. Introduction

A honeypot is a cybersecurity mechanism designed to intentionally attract attackers in order to study their behavior and gather threat intelligence. Unlike traditional security tools that block attacks, honeypots allow attackers to interact with a decoy system, enabling security professionals to analyze attack techniques in a controlled environment.

This project focuses on building a **lightweight SSH honeypot using Python**, simulating a vulnerable SSH service to detect and log unauthorized access attempts.

---

2. Objectives

- To design a functional SSH-based honeypot
- To capture and log attacker connection attempts
- To analyze attacker behavior using logged data
- To understand real-world attack patterns in a safe environment

---

3. Tools & Technologies Used

- **Operating System:** Kali Linux
- **Programming Language:** Python 3
- **Network Protocol:** TCP (SSH simulation)
- **Libraries Used:** socket, datetime

---

4. Honeypot Architecture

The honeypot operates by:

1. Listening on a non-standard SSH port (2222)
2. Accepting incoming TCP connections
3. Recording attacker IP address and SSH banner
4. Logging all captured data into a log file
5. Closing the connection without granting access

This ensures attacker interaction without compromising the host system.

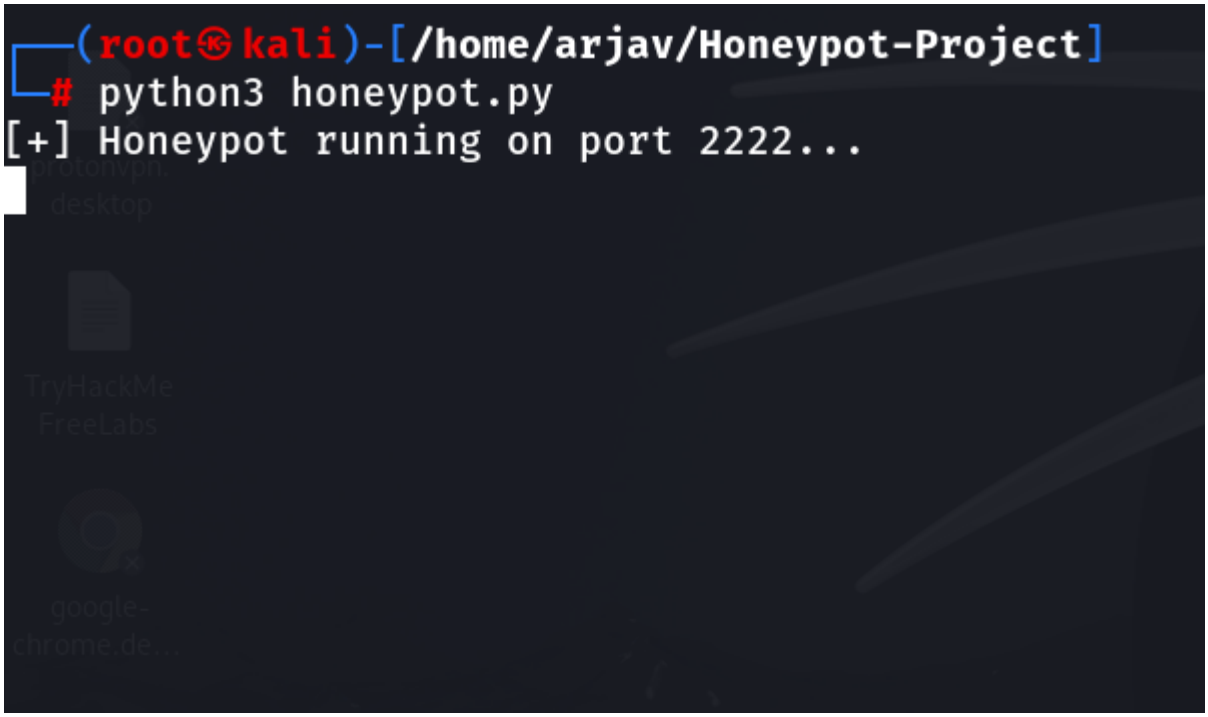

Figure 1: Honeypot Service Initialization

The Python-based honeypot successfully initialized and started listening on port 2222. This confirms that the fake SSH service is active and ready to capture incoming connection attempts.

---

5. Implementation

A Python script was developed to:

- Bind to port 2222
- Accept multiple incoming connections
- Capture attacker metadata
- Store logs in a timestamped format

The honeypot runs continuously until manually stopped by the administrator.

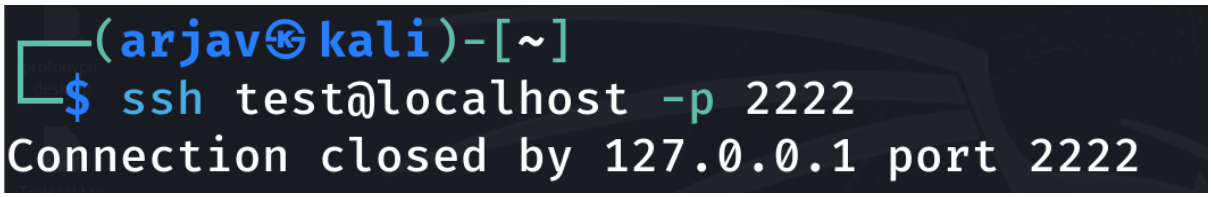

Figure 2: Simulated Attacker Attempting SSH Connection

A simulated attacker attempts to establish an SSH connection to the honeypot on port 2222. The connection is intentionally closed, demonstrating that no real system access is provided while still allowing activity to be logged.

---

6. Attack Simulation

To test the honeypot:

- An SSH connection was initiated using:
- ssh test@localhost -p 2222
- The honeypot successfully detected the connection
- The attacker IP and SSH banner were logged

This confirmed that the honeypot functions as intended.

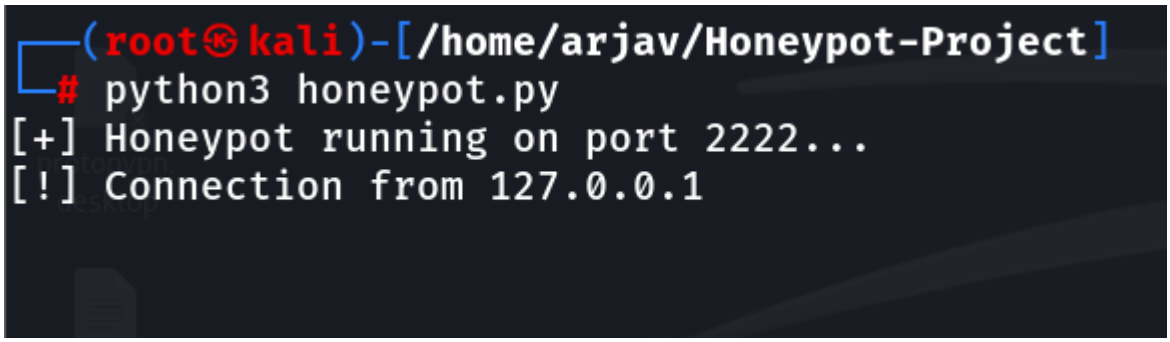

Figure 3: Honeypot Detecting Incoming Connection

The honeypot detects and reports an incoming connection attempt, capturing the attacker's IP address in real time. This confirms the honeypot's ability to identify suspicious network activity.

---

7. Logging & Analysis

Captured logs include:

- Date and time of attack
- Source IP address
- SSH banner information

Example:

[2026-02-06 13:11:59] IP: 127.0.0.1 | Data: SSH-2.0-OpenSSH_10.0p2

Such logs can help identify attacker tools and techniques..

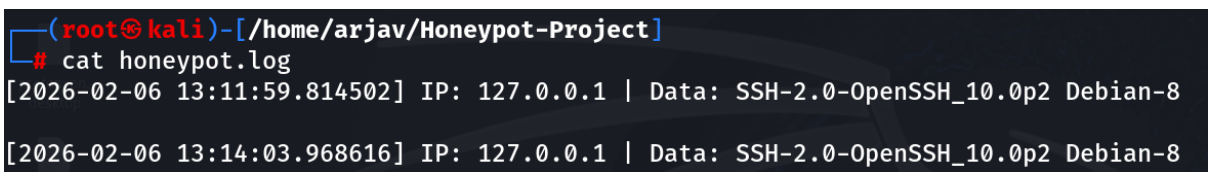

Figure 4: Logged Attacker Activity Captured by Honeypot

The honeypot log file displaying captured attacker details, including timestamp, source IP address, and SSH banner information. This demonstrates how honeypots can be used to gather threat intelligence and analyze attack patterns.

---

8. Results

- Honeypot successfully captured connection attempts
- Logs stored attacker metadata accurately
- No system compromise occurred
- Demonstrated effective threat monitoring

---

9. Conclusion

This project demonstrates how honeypots can be used as a powerful cybersecurity tool to monitor malicious activity. The Python-based honeypot provides a simple yet effective method to study attack behavior while maintaining system safety.

---

10. Ethical Considerations

This honeypot was deployed strictly in a controlled lab environment for educational purposes. No real users or external systems were targeted.

---

11. Future Enhancements

- Credential capture simulation
- Geo-location of attacker IPs
- Web dashboard for log visualization
- Multi-protocol honeypot support