SJIF Impact Factor (2025): 8.688| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 10 | Issue: 5 | May 2025 - Peer Reviewed Journal

PYPOT – PYTHON BASED MULTILAYERED HONEYPOT

B. Kalaiselvi, C. Sibin, M. Naveen, D. Nithish Kumar
Associate Professor, Department of Computer Science and Engineering, Mahendra Engineering College.
UG Student, Department of Cyber Security, Mahendra Engineering College.
UG Student, Department of Cyber Security, Mahendra Engineering College.
UG Student, Department of Cyber Security, Mahendra Engineering College.

Article DOI: https://doi.org/10.36713/epra21224
DOI No: 10.36713/epra21224

ABSTRACT
Cyberattacks are becoming more advanced and frequent, making it essential to develop proactive defenses that go beyond traditional security systems. Honeypots play a key role in this space by attracting attackers and studying their behavior in a controlled environment. This project introduces Pypot, a lightweight, agent-less, and Python-based honeypot framework that simulates both SSH and web-based attack surfaces. Designed to be portable and easy to deploy, Pypot can run on any platform that supports Python. It integrates a fabricated Linux shell, a fake CMS-style admin login page, and a real-time dashboard to monitor intrusion attempts and generate alerts. A unique feature of Pypot is its ability to track the geolocation of attacking IP addresses and visualize them on an interactive world map, helping analysts understand global threat distribution. Unlike container-based systems like T-Pot, Pypot avoids dependency-heavy environments, offering a standalone solution with modular components. By combining deception, live logging, alerting, and geospatial analysis, Pypot serves as both an educational tool and a practical defense mechanism. Its flexible architecture allows for future extensions, such as FTP, MySQL, and malware trap modules. In an age where threats constantly evolve, Pypot provides an accessible and insightful way to study and defend against attacker behavior.

KEYWORDS: *Honeypot, Python, SSH Emulation, Web Honeypot, IP Tracking, Threat Analysis, Geolocation Mapping, Intrusion Detection, Real-time Monitoring.*

INTRODUCTION

In the modern digital world, cybersecurity threats are more prevalent and dangerous than ever before. Organizations, governments, and individuals face increasing risks from attackers who constantly seek to exploit system vulnerabilities. Traditional defence mechanisms such as firewalls and antivirus software can help reduce these threats, but they are not always effective at detecting new or unknown attack patterns. To bridge this gap, honeypots have emerged as a valuable security solution that not only lures attackers but also helps in studying their behaviour in a controlled environment.

A honeypot is a security mechanism that mimics real services or systems to attract cyber attackers. These traps are used to detect, analyse, and understand unauthorized access attempts without exposing actual resources to danger. Over time, honeypots have become essential tools in threat intelligence, allowing security teams to collect valuable information on attack vectors, techniques, and potential vulnerabilities. However, many existing honeypot systems are complex, rely heavily on Linux or Docker environments, and often require significant system resources and configuration.

To overcome these limitations, we introduce *Pypot*, a flexible, lightweight, and Python-based honeypot system built for easy deployment across different platforms. Unlike traditional container-based honeypots, Pypot does not depend on Unix-based systems or virtualization. Instead, it runs as a standalone script and provides an SSH and web-based emulation layer to simulate vulnerable environments. Pypot includes a fabricated Linux shell and a fake CMS-style admin login page to trick attackers into interacting with the system, while all activity is logged and monitored.

One of the key highlights of this project is the real-time dashboard that displays live logs and attacker data. It also integrates IP geolocation tracking and maps attacker locations across the globe, giving users a clearer picture of threat origins. This geospatial visualization adds a new dimension to honeypot analysis by providing not just *what* happened, but also *where* it came from.

Designed with modularity and educational value in mind, Pypot is ideal for students, cybersecurity researchers, and penetration testers. It serves both as a hands-on learning platform and a practical defence tool that can be extended to include additional services like FTP or database honeypots. As cyberattacks continue to rise, projects like Pypot play a crucial role in proactive defence by helping analysts understand the mindset and behaviour of intruders

OBJECTIVES

The primary aim of this project is to design and implement a Python-powered, multi-layered honeypot system capable of emulating both SSH and web-based attack surfaces. The system is intended to actively monitor and log malicious interactions in real time, providing valuable insights into potential threats. Furthermore, it aims to enhance defensive readiness by sending instant alerts to system administrators via Telegram upon detecting suspicious behavior. An additional goal is to integrate IP geolocation tracking to map the origins of attacks globally, offering a broader understanding of emerging threat landscapes.

METHODOLOGY

1. Need for a Deceptive Monitoring System:

Traditional security tools like firewalls and intrusion detection systems are designed to prevent or alert against unauthorized access. However, these tools often fail to provide deep insights into attacker behaviour. Honeypots offer an alternative approach by intentionally exposing fake services to attract malicious actors and silently observe their techniques. This proactive approach helps defenders study real-world attack patterns, collect threat intelligence, and strengthen defensive strategies.

2. Pypot Framework Overview

Pypot is designed as a multi-layered honeypot system that mimics vulnerable SSH terminals and web interfaces. It operates in real-time and captures all forms of interaction initiated by attackers. The framework is broken into the following main components: SSH Honeypot Subsystem Simulates a fake Linux shell that can receive and "respond" to common terminal commands. It mimics directory structures like /etc, /home, /bin, and includes artificial command responses to appear realistic. Web Honeypot Subsystem Hosts a login panel styled like a real CMS (e.g., WordPress or admin dashboard). Credentials entered here are logged for analysis. The interface is designed with HTML, CSS, and Flask to support interaction.Logging Module. Maintains two separate log files: ssh.log and web.log. Each log entry is timestamped and structured with IP, action, and type of event. These logs serve as the data foundation for alerts and dashboard updates.Alert System A Python-based alert module scans the logs for suspicious events (e.g., brute force attempts, execution of known malicious commands). Upon detecting such behaviour, the system triggers instant Telegram notifications. Live Dashboard Interface A Flask web app renders both SSH and web honeypot logs in real-time. It offers log filters, attack summaries, and displays live alerts. Designed for easy accessibility and clarity, the dashboard enables real-time monitoring. IP Geolocation Tracker Captures attacker IP addresses and queries external IP-to-location APIs (e.g., IPinfo or ipapi) to identify the origin country. The system then maps the coordinates onto a world map using JavaScript chart libraries (e.g., Leaflet.js or Chart.js with GeoJSON).

3. Lifecycle Flow of Pypot Operation

a. Initialization: The process begins when the main.py script is executed. It triggers concurrent threads to launch the SSH honeypot, the web honeypot, and supporting modules like the logger and alert engine.

b. Honeypot Engagement

Once deployed:

- Attackers scanning open ports or using automated bots are redirected to the SSH or Web interfaces.
- For SSH, users encounter a responsive fake terminal.
- For Web, users land on a crafted login panel.

All interactions are silently recorded.

c. Log Aggregation

As users (attackers) interact: Terminal input and command history are logged in ssh.log. Login attempts, IPs, and credentials submitted are saved in web.log. The logging module ensures no data loss during simultaneous interactions.

d. Threat Detection and Alerting:

A live Python script reads from both log files and scans for suspicious activity patterns. These include:

- Repeated login failures
- Execution of dangerous shell commands (e.g., wget, rm -rf, chmod, nc)
- Rapid request sequences indicating scanning or Bots

Once a threshold is triggered, the alert module:

- Formats a human-readable message
- Sends it to the registered admin via Telegram bot integration

e. IP Geolocation & Visualization

Every IP found in the logs is sent to a public geolocation API to extract country, region, and coordinates. These details are:

- Logged in a separate geo-log
- Plotted dynamically on an interactive world map within the dashboard This visualization helps identify attacker trends, sources, and global threat landscapes.

f. Real-Time Monitoring via Dashboard

The dashboard renders:

- Live logs with search and filter capability
- Summary stats (e.g., total hits, commands issued, logins tried)
- Alert feed
- Geo-map showing current attacker origins

Users can monitor the system passively or actively without needing to access raw files.

4. Advantages of Pypot's Design:

- Standalone and Lightweight
- No need for Docker or Linux
- Python-compatible OS.
- Layered Design:Multiple honeypot types (SSH + Web)simulate real-world attack surfaces.
- Live Monitoring and Alerts
- Reduces detection delay and offers immediate threat awareness.

Step-by-Step Process of Pypot Operation:

1. System Initialization

a. Start the main.py script

- This master script initializes all the honeypot components in parallel using Python's threading module.

b. Load Configuration and Resources

- Paths to logs, API keys for geolocation, Telegram tokens, and port settings are loaded at startup.

2. Launch Honeypot Modules

c. Start SSH Honeypot Listener

- A TCP socket is opened on a selected port (e.g., 2222).
- When a connection is received, a mock login prompt is served.
- On successful fake login, a simulated shell environment is provided

d. Start Web Honeypot Interface

- A Flask-based web server runs on port 8080.
- It renders a fake admin login interface styled to mimic a real CMS.
- Any credential entered is accepted and recorded, giving access to a fake dashboard.

3. Logging of Attacker Interactions

e. Capture SSH Commands

- All inputs typed by the attacker in the SSH shell are saved in ssh.log along with:
    - Timestamp
    - IP address
    - Command entered

f. Record Web Login Attempts

- Every login submission is logged in web.log, storing:
    - IP address
    - Username/password submitted
    - Time of login

4. Real-Time Log Monitoring

g. Log Watcher Script Runs Continuously

- A Python script reads the log files in real time (using tail or file-pointer logic).
- It identifies suspicious patterns such as:
    - Multiple failed logins
    - Use of known hacker tools or keywords (e.g., wget, nc, rm, curl)

h. Triggering Alerts

- When such patterns are detected, an alert is formatted and pushed to a preconfigured Telegram bot.

SJIF Impact Factor (2025): 8.688| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 10 | Issue: 5 | May 2025 - Peer Reviewed Journal

- The alert message includes IP, time, and type of activity.

5. IP Geolocation Tracking

   i. Extract Attacker IPs from Logs
   - The monitoring script identifies new IPs from both logs.

   j. Query Geolocation API
   - Each unique IP is sent to a geolocation API (e.g., ipinfo.io, ipapi.co) via HTTPS request.
   - The API returns: Country, City, Latitude & Longitude

   k. Store Location Data
   - The result is saved in a separate geo_log.json file or database for later analysis.

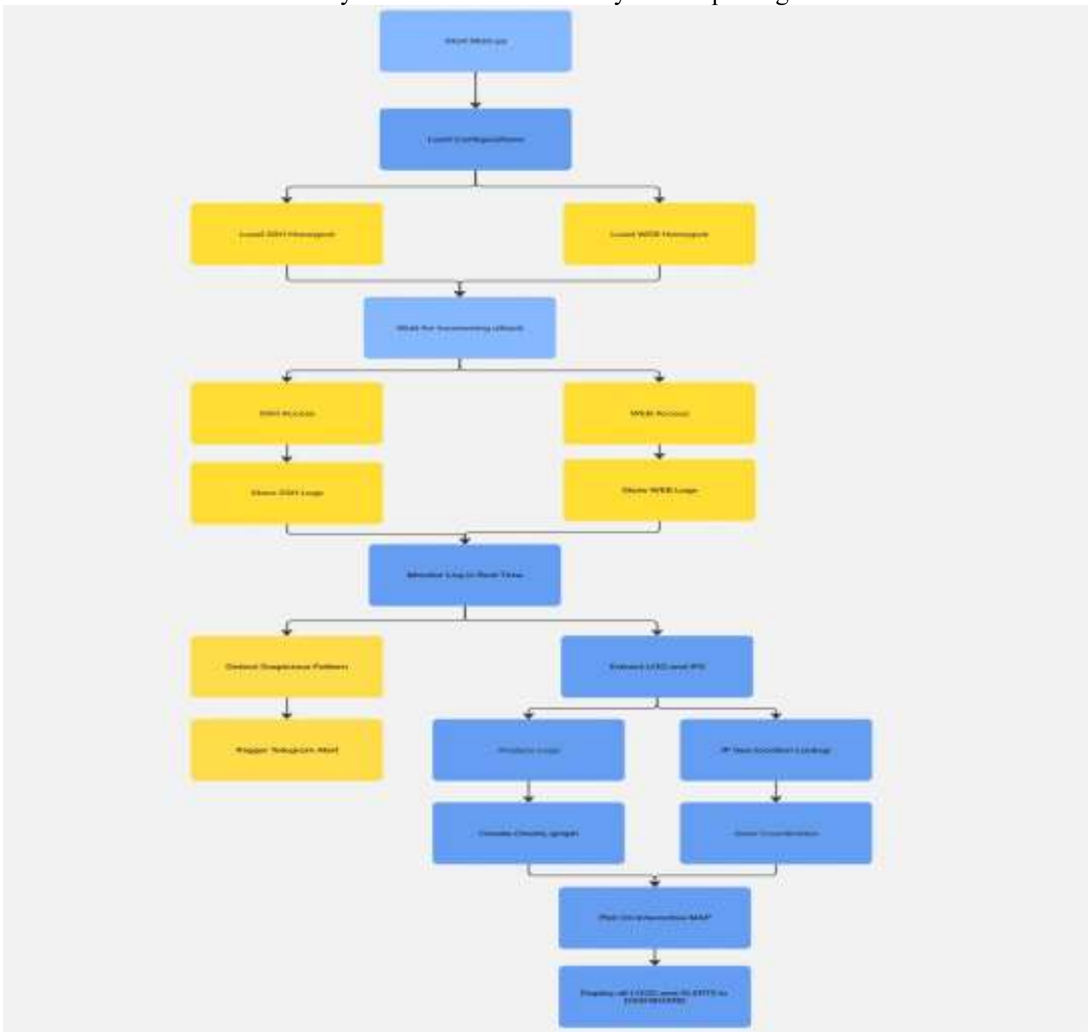
6. Live Dashboard & Visualization

   l. Dashboard Loads Log Data
   - A Flask web dashboard continuously reads the updated logs.
   - Displays:
     - Latest activity (SSH/Web)
     - Total hits
     - Alert history

   m. Display Geo Map of Attacks
   - Using Leaflet.js or Chart.js with GeoJSON, the coordinates are plotted on a world map.
   - Each dot or flag represents an attacker's origin.

7. Termination or Logging Continuation

   n. Manual or Timed Shutdown
   - The system runs indefinitely until manually stopped.
   - All data remains stored locally for further forensic analysis or reporting.



RESULT & DISCUSSION

1. Real-Time Intrusion Capture

The implementation of Pypot successfully captured unauthorized interaction attempts on both the SSH and web honeypot layers. Attackers connected via open ports, believing the services were real, and attempted various commands or credential-based login attacks. The fake shell prompted several brute-force login attempts, while the web interface attracted credential stuffing and common CMS login probes. This confirmed that the honeypot modules are convincingly emulated and functionally deceptive.

2. Accuracy and Responsiveness of Logging System

Both ssh.log and web.log recorded attacker inputs with accurate timestamps, IP addresses, and action details. No input was lost during simultaneous interaction attempts, thanks to the real-time monitoring module running parallel to the honeypot systems. The logs remained consistently formatted and easy to parse, which supports further log analysis, filtering, and forensic investigation.

3. Effectiveness of Alert Mechanism

The alert system successfully identified suspicious behaviors such as:
Use of reconnaissance commands
Suspicious binaries or payload fetch attempts Repeated failed logins within short intervals Whe such patterns were detected, alerts were instantly sent via Telegram with context, including IP addresses and timestamps. This proved valuable for immediate awareness and threat response readiness.

4. IP Geolocation and Map-Based Threat Insight:

Pypot’s IP tracking module accurately fetched location data from incoming attacker Ips using Geolocation API. The coordinates were successfully plotted on a real-time interactive map embedded in the dashboard. This visualization showed attacker clusters from specific countries, providing deeper insight into regional threat trends. It added context to raw logs, helping analysts understand not just *how* an attack occurred, but also *where* it originated.

5. Performance and Stability:

Despite running multiple threads and handling real-time updates, the Pypot system demonstrated excellent performance and remained lightweight. The Python threading model ensured non-blocking behaviour, and CPU/memory usage stayed within acceptable limits even during simulated bot attacks. The dashboard loaded without delay, and log updates were visible in real-time with minimal lag.

6. Educational and Practical Value

From an academic and research perspective, Pypot serves as an excellent platform to:
Teach log analysis and threat detection Simulate real-world attacks in a sandboxed environment demonstrate the use of alerting, geolocation, and visualization Its modular and transparent architecture makes it ideal for cybersecurity coursework, demonstrations, and penetration testing

SUGGESTIONS

To enhance the capabilities of the Pypot framework, several improvements can be considered for future development. One recommendation is to incorporate additional emulation layers that mimic FTP, SQL databases, and IoT protocols, thereby expanding the range of bait services for potential attackers. Another area of improvement involves upgrading the threat detection logic to include behavioral anomaly analysis, allowing for more nuanced and intelligent identification of malicious activities. Integrating the honeypot system with centralized Security Information and Event Management (SIEM) platforms can also enable real-time threat correlation and response at a larger scale. Finally, implementing persistent log storage and secure remote backup mechanisms will help preserve crucial forensic data and support extended analysis

CONCLUSION

In an era of rapidly evolving cyber threats, deception-based security mechanisms like honeypots have become increasingly important. This project, *Pypot*, presents a lightweight, Python-powered honeypot system designed to simulate SSH and web-based attack surfaces. It enables the safe observation and analysis of malicious behaviour without exposing any real systems to risk.

The success of Pypot lies in its ability to capture real-time interactions, generate instant alerts, and visualize attacker origins through IP geolocation mapping — all within a standalone and platform-independent environment. Its modular structure, built entirely in Python, makes it highly adaptable and easy to extend. Whether deployed for education, research, or operational cybersecurity, Pypot provides meaningful insights into attacker tactics and patterns.

Beyond its technical effectiveness, Pypot also serves as a valuable learning tool. It offers students, security enthusiasts, and analysts a hands-on opportunity to understand intrusion behaviour, practice log analysis, and build situational awareness through live monitoring.

Looking ahead, Pypot can be enhanced with additional honeypot layers such as FTP, MySQL, or IoT emulation. Integration with centralized SIEM platforms, email alerting, or malware behaviour analysis modules can further increase its value in real-world environments.
By combining simplicity, deception, and visibility, Pypot stands as a practical and insightful approach to modern cyber defence.

CONFLICT OF INTEREST

The *Pypot* project is intended solely for educational and cybersecurity research purposes. It is designed to simulate vulnerable services in a safe, isolated environment for the purpose of studying attacker behavior. There are no commercial interests, and the authors affirm that this project was conducted without any conflict of interest. Care has been taken to ensure that the tool does not engage in any unauthorized network interaction or violate ethical standards.

REFERENCES

1. *Spitzner, L. (2003). Honeypots: Tracking Hackers. Addison-Wesley.*
2. *Seifert, C. (2009). Honeypots for Windows Systems: Creating a Realistic Virtual Honeynet. SANS Institute.*
3. *Provos, N., & Holz, T. (2007). Virtual Honeypots: From Botnet Tracking to Intrusion Detection. Addison-Wesley.*
4. *Conti, G. (2007). Security Data Visualization: Graphical Techniques for Network Analysis. No Starch Press.*
5. *Rash, M. (2005). Intrusion Detection with Snort, Apache, MySQL, PHP and ACID. No Starch Press.*
6. *Orebaugh, A., Ramirez, G., & Beale, J. (2006). Wireshark & Ethereal Network Protocol Analyzer Toolkit. Syngress.*
7. *IPinfo.io. "Free IP Geolocation A Python Software Foundation. (2024). Python 3 Documentation.*
8. *Flask Documentation. (2024).*
9. *Telegram Bot API.*