Project Plan: AI-Powered Honeypot Threat Intelligence Platform

Document Version: 2.0
Date: 19/10/2021
Author: Darrshan A/L Erettai Muniandy
Status: Initial phase

Contents

1.0 Executive Summary

This project outlines the design and implementation of an advanced **AI-Powered Honeypot Threat Intelligence Platform**. The system will deploy a high-interaction honeypot within a controlled, isolated lab environment to attract and record malicious activity. The core innovation lies in the integration of Machine Learning (ML) models to automatically analyze the captured log data, classifying attacks and identifying behavioral patterns. Processed intelligence will be visualized through a dynamic, real-time web dashboard built with the Flask framework. The final deliverable is a proof-of-concept platform that demonstrates proactive threat detection, moving beyond traditional signature-based methods to provide actionable security insights.

2.0 Project Goals & Objectives

2.1 Primary Goal

To build an integrated system that captures attacker interactions with a honeypot, analyzes them using AI/ML to derive intelligent threat data, and presents the findings on an interactive web dashboard.

2.2 Specific Objectives

1. **Infrastructure Setup:** Establish a virtualized lab network hosting a Kali Linux honeypot, a Windows Server 2019 Active Directory (AD) domain controller, and a Windows 10 client.
2. **Honeypot Deployment & Configuration:** Install and configure a sophisticated honeypot (e.g., Cowrie) on Kali Linux to emulate SSH/Telnet services and log all interactions comprehensively.
3. **Data Collection & Centralization:** Implement a logging pipeline to collect, parse, and forward honeypot logs to a central storage for analysis.
4. **AI/ML Analysis Engine:**
    - **Feature Engineering:** Extract meaningful features from raw logs (e.g., source IP geolocation, command frequency, failed login attempts, session duration).
    - **Model Development & Training:** Develop ML models for:
        - **Attack Classification:** Categorize sessions as brute-force, password spray, command injection, or normal.
        - **Anomaly Detection:** Identify novel or unusual attack patterns using unsupervised learning.
5. **Web Dashboard Development:** Create a secure Flask web application to display:
    - Real-time attack maps.
    - Statistical summaries of attack data.
    - Results from the ML analysis (classifications, confidence scores).

    - Interactive charts and graphs.

6. **Validation & Testing:** Simulate various attack scenarios from the Windows 10 client to validate the entire pipeline, from data capture to AI analysis and dashboard visualization.

3.0 Scope

3.1 In-Scope

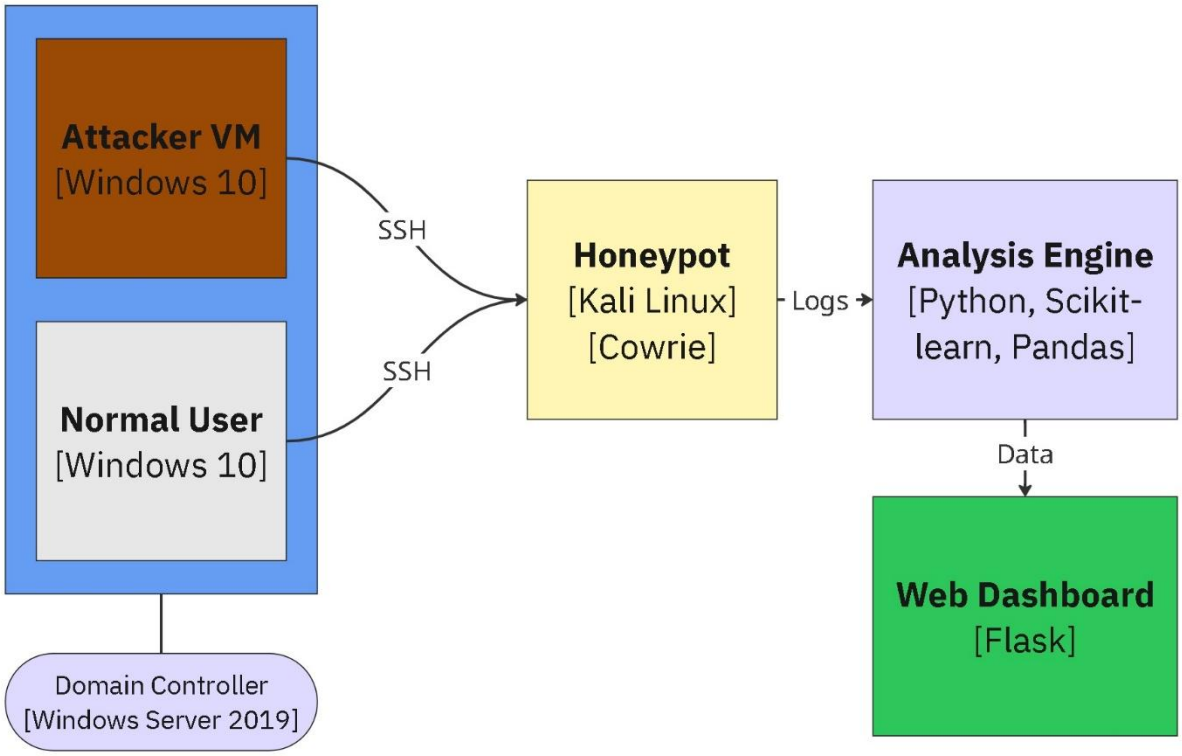
- The project is confined to a VirtualBox virtualized environment.
- Focus on SSH-based attacks targeting the Kali Linux honeypot.
- Analysis of logs generated *only* by the deployed honeypot software.
- Development of a proof-of-concept ML model for classification.
- A functional Flask-based dashboard for internal demonstration.

3.2 Out-of-Scope

- Deployment in a production or internet-facing environment.
- Analysis of attacks against the Windows Server or Windows 10 VM (though they provide a realistic network context).
- Real-time blocking or active response in the network.
- A commercially scalable, multi-user web application.

4.0 Technical Architecture

The system will follow a modular, multi-tier architecture:



Data Flow:

1. **Initiation:** An "attacker" (Windows 10 VM) initiates an SSH connection to the honeypot (Kali Linux).
2. **Capture:** The honeypot (Cowrie) emulates a vulnerable system, logs all interactions (login attempts, commands, file downloads), and saves them in JSON format.
3. **Processing:** A Python script periodically ingests these JSON logs.
    - **Data Parsing:** Extracts key fields (timestamp, source_ip, username, password, command, session).
    - **Feature Engineering:** Creates derived features (e.g., login_fail_count, unique_commands_used, is_geo_high_risk).
4. **AI Analysis:** The processed data is fed into a pre-trained ML model.
    - The model classifies the session and calculates an anomaly score.
    - Results are stored in a SQLite database.
5. **Visualization:** The Flask application queries the database and renders the insights on the dashboard via charts, tables, and maps.

5.0 Technology Stack

COMPONENT	TECHNOLOGY CHOICE	JUSTIFICATION
VIRTUALIZATION	Oracle VirtualBox	Robust, free, and supports complex internal networking.
HONEYPOT OS	Kali Linux	Pre-loaded with security tools; ideal for hosting honeypots.
HONEYPOT SOFTWARE	**Cowrie** (Primary)	Advanced, open-source SSH/Telnet honeypot that logs extensively.
DOMAIN ENVIRONMENT	Windows Server 2019, Windows 10	Provides a realistic enterprise network context.
AI/ML FRAMEWORK	Python, Scikit-learn, Pandas, NumPy	Industry-standard for data analysis and machine learning.
DEVELOPMENT ENV.	Jupyter Notebook	Ideal for iterative model development and data exploration.
WEB FRAMEWORK	Flask (Python)	Lightweight, flexible, and easy to integrate with our ML Python backend.
DATABASE	SQLite (Development)	Simple file-based database for the POC. Easily upgradable to PostgreSQL.
FRONTEND	HTML5, CSS3, JavaScript, Chart.js	For creating dynamic and interactive visualizations.
GEOLOCATION	MaxMind GeoLite2	Free IP geolocation database for mapping attacker locations.

6.0 Project Phases & Timeline

This project is structured into five distinct phases over an estimated **6-8 week** timeline.

PHASE	KEY TASKS	DELIVERABLE	DURATION
PHASE 1: INFRASTRUCTURE & HONEYPOT	1. Configure VirtualBox NAT/Host-Only network. 2. Install & configure Win Server 2019 AD, join Win 10 to domain. 3. On Kali: Install & harden OS, install & configure Cowrie.	A fully operational lab network with a logging honeypot.	1-2 Weeks

PHASE 2: DATA ENGINEERING	1. Develop Python log parser for Cowrie JSON logs. 2. Design feature extraction logic. 3. Set up SQLite DB schema for storing raw and processed data.	A script that automatically parses logs and populates the database.	1 Week
PHASE 3: AI/ML MODEL DEVELOPMENT	1. **Exploratory Data Analysis (EDA)** on initial captured data. 2. **Feature Selection:** Define the final feature set for the model. 3. **Model Training:** Train and evaluate classifiers (e.g., Random Forest, SVM). 4. **Model Export:** Serialize the final model (e.g., using pickle).	A trained and validated ML model file ready for integration.	2-3 Weeks
PHASE 4: WEB DASHBOARD & INTEGRATION	1. Develop Flask application with routes and templates. 2. Integrate the ML model into a Flask route for analysis. 3. Create dashboard views with Chart.js for visualizations. 4. Implement the IP geolocation mapping feature.	A fully functional web dashboard displaying AI-analyzed honeypot data.	1-2 Weeks
PHASE 5: TESTING & DOCUMENTATION	1. Execute controlled attack simulations. 2. Validate end-to-end data flow and dashboard accuracy. 3. Create final project documentation and presentation.	A tested, documented, and presentation-ready project.	1 Week

7.0 Risk Assessment & Mitigation

RISK	PROBABILITY	IMPACT	MITIGATION STRATEGY
LAB ENVIRONMENT ESCAPE	Low	High	Use isolated Host-Only/NAT networking in VirtualBox. Do not use Bridged mode. Ensure honeypot VM has no critical host data.
ML MODEL INACCURACY	Medium	Medium	Start with a simple, interpretable model (Random Forest). Manually label a small dataset for validation. Use confidence thresholds.
HONEYPOT DETECTION	Medium	Low	Use a high-interaction honeypot like Cowrie which is harder to fingerprint. Customize its configuration to appear more realistic.
PROJECT SCOPE CREEP	High	Medium	Adhere strictly to the defined in-scope items. The primary focus is the POC, not a production system.
DATA VOLUME & PERFORMANCE	Low	Low	For the POC scale, SQLite and a single Flask process are sufficient. The architecture allows for scaling components later.

8.0 Success Metrics

The project will be deemed successful if:

- The honeypot successfully captures and logs SSH interaction data from the Windows 10 VM.
- The ML model can classify simulated attack types with **>85% accuracy** on a validation set.
- The Flask dashboard loads without errors and displays at least:
  - A live feed of recent attacks.
  - A pie chart of attack classifications.
  - A world map plotting attacker IP locations.
- The entire pipeline operates autonomously from log generation to dashboard update.

9.0 Conclusion

This AI-Powered Honeypot Threat Intelligence Platform project represents a significant step beyond traditional security monitoring. By leveraging machine learning to analyze attacker behavior automatically, it provides a foundation for proactive cybersecurity defense. This proof-of-concept will not only demonstrate technical proficiency in multiple domains but also serve as a powerful showcase of how AI can be operationalized to derive actionable intelligence from security data. We are confident in the proposed plan and are prepared to move forward with its execution.

Appendix A: Preliminary Feature Set for ML Model

- session_duration
- number_of_failed_logins
- number_of_unique_commands
- presence_of_destructive_commands (e.g., rm, dd, wget)
- country_of_origin (encoded)
- username_entropy

Appendix B: Potential Future Enhancements

- Integrate multiple honeypots (e.g., a Windows-based honeypot).
- Implement a real-time alerting system (e.g., via Telegram/Slack).
- Containerize the application using Docker for easy deployment.
- Upgrade to a deep learning model for analyzing command sequences (using LSTMs).