

DECLARATION

I hereby declare that this Project Low Interaction Honeypot System is my own work and has, to the best of my knowledge, not been submitted to any other institution of higher learning.

Student: _____ Registration Number: _____

Signature: Date:

This project Low Interaction Honeypot System has been submitted as a partial fulfillment of requirements for the Bachelor of Science in Software Engineering of Multimedia University of Kenya with my approval as the University supervisor.

Supervisor: _____

Signature: Date:

Abstract

This project presents a low interaction honeypot system. Deploying a cyber-honeypot system serves as a strategic deception mechanism for cyber defenders. By luring attackers into this controlled environment, defenders can gather valuable intelligence. The honeypot system provides insights into various aspects of attacks, including:

Attack Techniques: Defenders learn about the most common attack methods used against the honeypot.

Attacker Behavior: Details such as the attacker's IP allocation, penetration attempts, and executed commands within the honeypot are recorded.

Malware Analysis: Defenders gain knowledge of the malware downloaded by attackers.

Modern honeypots must be deceptive to avoid detection. They mimic real systems, ensuring that attackers invest their efforts while defenders extract critical information. This project explores the implementation of such a honeypot system using open-source tools, enhancing cybersecurity.

Table of Contents

Chapter One: Introduction.....	5
1.1 Background of Study	5
1.2 Problem Statement.....	5
1.3 Aim of the Study	5
1.4 Significance/Justification of Study	7
1.5 Scope.....	7
1.6 Assumptions.....	8
1.7 Limitations.....	8
Chapter 2: Literature review	10
2.1 Introduction	10
2.2 Related Systems.....	10
2.3 Limitations/Weaknesses of These Systems	10
2.4 Handling Weaknesses: Proposed Solution.....	11
Chapter 3: Methodology.....	12
3.1 Introduction	12
3.2 The Methodology.....	12
3.3 Data Collection Methods and Tools.....	12
3.4 Project Resources (Hardware/Software)	13
Project Timeline	14
3.5 Project Budget.....	16
Chapter 4: System Analysis	18
4.1 Detailed Analysis of the System.....	18
4.2 System Requirements	21
Chapter 5: System Design	24
5.1 Architectural Design.....	24
5.2 Database Design.....	24
5.3 User Interface Design.....	25
Chapter 6: Implementation and Testing.....	28
6.1 Development Environment.....	28
6.2 System Components	28
6.3 Test Plan.....	29

Chapter 7: Conclusion	31
7.1 Achievements and Lessons Learned	31
7.2 Conclusions	31
7.3 Recommendations	31

Chapter One: Introduction

1.1 Background of Study

In today's rapidly evolving digital landscape, cybersecurity remains a critical concern for organizations worldwide. As technology advances, so do the tactics employed by malicious actors seeking to compromise networks, steal sensitive data, and disrupt operations. Traditional security measures, while effective, cannot guarantee absolute protection against all threats. Therefore, innovative approaches are necessary to enhance our defenses.

One such approach is the deployment of honeypots which are deceptive systems designed to attract and engage cyber attackers. By strategically placing honeypots alongside production systems, organizations gain valuable insights into attacker behavior, tactics, and vulnerabilities.

1.2 Problem Statement

Despite robust security measures, vulnerabilities persist. Organizations face challenges in identifying and understanding emerging threats. The mechanisms deployed by cyber attackers keep evolving and thus demand that the proponents of cyber security find a way to stay ahead of the curve. As such, there's need to proactively analyze systems and implement appropriate mechanisms to protect them from attacks.

1.3 Aim of the Study

The primary aim of this project is to create a simple honeypot system that complements existing security mechanisms. By this study I should be able to design a simplified honeypot system to aid different organizations bolster their security efforts. We should be able to provide a simplified, cost efficient as well as resource efficient system in terms of computing power to the organizations in the market. Our objectives include:

Diverting Malicious Traffic: Redirecting attacks away from critical systems toward honeypots. The culprits will then interact with the honeypots as though they were interacting with a live system unbeknownst to them.

Early Warning System: Detecting ongoing attacks before they impact vital infrastructure. This way the cyber security experts can deploy appropriate counter measures and always stay ahead of the attackers.

Gathering Attacker Intelligence: Through the study, we'll be able to gain insight into attacker's mindsets, motivations and attack patterns. The intelligence gathered can then be deployed to

design robust attack proof systems as well as coordinate timely response in the wake of an attack.

Research Objectives

Our specific research objectives are as follows:




Investigate Different Types of Honeypots:

Low-Interaction Honeypots: These honeypots simulate minimal services, such as open ports or basic protocols. They are lightweight, easy to deploy, and consume fewer resources. However, their insights into attacker behavior are limited.

High-Interaction Honeypots: These honeypots closely mimic real systems, including full operating systems and services. While more resource-intensive, they provide detailed data on attacker interactions. Our research will explore the trade-offs between low and high interaction.




Assess Honeypot Effectiveness:

We aim to quantify how well honeypots perform in diverting malicious traffic away from critical systems. Metrics include:

-  **Attack Diversion Rate:** What percentage of attacks are directed toward honeypots?
-  **False Positive Rate:** How often do legitimate users inadvertently interact with honeypots?
-  **Early Warning Capability:** How quickly can honeypots detect ongoing attacks?

Analyze Attacker Behavior and Tactics:

By studying honeypot interactions, we can gain insights into:

-  **Attack Patterns:** What methods do attackers employ? Are there common attack vectors?
-  **Exploited Vulnerabilities:** Which vulnerabilities are most frequently targeted?
-  **Attack Motivations:** Are attackers financially motivated, politically driven, or simply curious?

Evaluate Honeypot Deployment Strategies:

We'll explore various deployment scenarios:

- ✚ Network Placement: Should honeypots be placed within the internal network, the DMZ, or both?
- ✚ Service Emulation: Which services (e.g., web servers, SSH, FTP) should honeypots emulate?
- ✚ Geographical Distribution: How does the location of honeypots impact their effectiveness?

Legal and Ethical Considerations:

We'll address legal implications, privacy concerns, and ethical boundaries related to deploying honeypots. This includes obtaining informed consent and ensuring compliance with relevant regulations.

Integration with Incident Response:

How can honeypot data feed into incident response processes? We'll explore ways to correlate honeypot events with real-world incidents.

Honeypot Resilience and Deception Techniques:

Investigate techniques to enhance honeypot resilience, such as dynamic deception (changing the honeypot's appearance) and adaptive responses (altering behavior based on attacker actions).

1.4 Significance/Justification of Study

The significance of this study lies in its potential to enhance network security. By deploying honeypots, organizations can:

- ✚ Learn from Adversaries: Understand how attackers operate and adapt defensive strategies accordingly.
- ✚ Forensic and Legal Insights: Gather evidence for legal proceedings and incident response.
- ✚ Complement Traditional Defenses: Honeypots add an extra layer of defense without replacing existing mechanisms.

1.5 Scope

This project will focus on creating a low-interaction honeypot which is a simple system that emulates vulnerable services. We will explore deployment options, monitoring techniques, and data analysis.

1.6 Assumptions

We assume that:

The organizations have existing cyber security infrastructure in place.

Honeypots will not prevent attacks but provide valuable insights.

The users of the honeypot have at least a basic understanding of the security needs of the systems and the importance of the different security measures.

The security teams understand the attack types tracked by the honeypot due to attack diversity.

1.7 Limitations

a) Limited Attack Surface:

Honeypots only capture attacks directed specifically at them. They do not provide visibility into attacks targeting other systems within the network. Attacks bypassing honeypots remain undetected, potentially leaving critical infrastructure vulnerable.

b) Resource Constraints:

High-interaction honeypots, which closely mimic real systems, consume significant resources (CPU, memory, and network bandwidth). Organizations must balance the deployment of honeypots with resource availability.

c) Deception Effectiveness:

Skilled attackers can identify honeypots by analyzing their behavior or fingerprinting their services. Honeypots must continually evolve to maintain their deceptive effectiveness.

d) Legal and Ethical Challenges:

Deploying honeypots raises legal and ethical questions. Organizations must ensure compliance with privacy laws and obtain informed consent. Honeypots that inadvertently capture legitimate user interactions can lead to privacy concerns.

e) False Positives and Noise:

Honeypots generate noise due to automated scans, probes, and low-level attacks. Distinguishing genuine threats from benign activity is challenging. False positives can overwhelm security teams and divert attention from real incidents.

f) Single Point of Failure:

Relying solely on honeypots as an early warning system creates a single point of failure. If attackers compromise or evade honeypots, critical systems remain exposed.

g) Attack Diversity:

Honeypots may attract specific types of attacks (e.g., web application attacks or port scans), but not all attack vectors. Understanding the limitations of honeypot data is crucial for accurate threat assessment.

Chapter 2: Literature review

1.1 Introduction

In the ever-evolving landscape of cybersecurity, honeypots play a crucial role in enhancing network defense. As organizations grapple with increasing threats, understanding the effectiveness and limitations of honeypot-based solutions becomes paramount. This literature review explores relevant research areas, focusing on botnet detection models within the context of smart factories.

1.2 Related Systems

2.2.1 Deception Technology

Deception technology, an advanced version of honeypots, considers attackers' perceptions and techniques. It is particularly relevant for securing Internet of Things (IoT) devices, medical devices, and embedded operating systems. By mimicking real services, deception technology attracts attackers, allowing close monitoring and behavior analysis.

2.2.2 Malware Honeypots

These honeypots leverage existing knowledge about attackers to prevent attacks. By detecting malware based on historical attack patterns, they enhance security, especially in scenarios like cryptocurrency theft.

2.2.3 Spam Versions and Email Traps

Honeypots effectively counter spam abuse by revealing spammers' IP addresses and other information. These systems serve as traps, capturing spam emails and aiding in backtracking spammers.

1.3 Limitations/Weaknesses of These Systems

While honeypots offer valuable insights, they are not without limitations:

Limited Attack Surface:

Honeypots only capture attacks directed at themselves, leaving other systems vulnerable. Attacks bypassing honeypots remain undetected.

Resource Constraints:

High-interaction honeypots consume significant resources. Balancing deployment with available resources is essential.

Deception Effectiveness:

Skilled attackers can identify honeypots. Continuous evolution is necessary to maintain deception.

Legal and Ethical Challenges:

Privacy concerns arise when capturing legitimate user interactions. Compliance with regulations and informed consent is crucial.

1.4 Handling Weaknesses: Proposed Solution

Our proposed solution aims to address these weaknesses:

Diverse Honeypot Deployment:

We will explore various honeypot types (low-interaction, high-interaction) to maximize coverage. Geographical distribution and service emulation will enhance effectiveness.

Machine Learning Analysis:

Recorded honeypot data will undergo machine learning analysis. Early warning actions will be generated based on detected botnet behaviors.

Legal Compliance:

We will ensure legal compliance and ethical use of honeypots. Privacy considerations will guide our approach.

Chapter 3: Methodology

1.1 Introduction

This chapter outlines the methodology employed to design, develop, and implement a low-interaction honeypot system. The chapter details the research approach, data collection methods, project resources, schedule, and budget. The methodology section serves as a roadmap for conducting the project and ensures a systematic approach to achieving the desired outcomes.

1.2 The Methodology

A low-interaction honeypot system is a cybersecurity tool designed to mimic legitimate systems and attract malicious actors. By mimicking real systems, honeypots can capture valuable data on attacker behavior, techniques, and tools. This project will focus on developing a low-interaction honeypot system, meaning it will provide limited functionality to attackers while still appearing realistic. This approach minimizes the risk of compromising real systems while still collecting valuable intelligence.

The chosen research methodology for this project will be the Design, Development, Test, and Implementation (DDTI) approach. This iterative approach involves defining system requirements, designing the honeypot architecture, developing the system, conducting rigorous testing, and finally deploying the honeypot in a controlled environment. The DDTI approach allows for continuous improvement throughout the development lifecycle and ensures a robust and functional honeypot system.

1.3 Data Collection Methods and Tools

Data collection is a crucial aspect of honeypot operation. The honeypot system will be designed to capture various data points on attacker interactions, including:

Attack source: IP address and geographical location of the attacking machine.

Attack type: The method used by the attacker to exploit the honeypot (e.g., port scans, malware deployment attempts).

Attack tools: Identification of specific tools or techniques used by the attacker.

Exploit attempts: Logging attempts to exploit vulnerabilities within the honeypot system.

Time of attack: Recording the date and time of each attack attempt.

To collect this data, the honeypot system will utilize various tools and techniques:

Packet Capture Tools: Software like Wireshark or tcpdump will be used to capture network traffic directed towards the honeypot. These tools provide detailed information on the nature of network communication attempts.

Log Files: The honeypot system will be configured to generate detailed logs recording all interactions with the system. These logs will include timestamps, source information, and details of attempted exploits.

System Monitoring Tools: System monitoring tools will be employed to track system resources, identify suspicious activity, and detect potential compromises within the honeypot environment.

The collected data will be stored securely and analyzed to understand attacker behavior patterns, identify prevalent threats, and inform future security strategies.

1.4 Project Resources (Hardware/Software)

This section details the hardware and software resources required to develop and implement the low-interaction honeypot system.

Hardware:

Server: A dedicated server with sufficient processing power, memory, and storage capacity will be required to run the honeypot system and handle potential attacker interactions. The server specifications will depend on the anticipated volume and complexity of attacks.

Network Switch: A network switch will be needed to connect the honeypot system to the network and isolate it from critical production systems.

Additional Hardware (Optional): Depending on the chosen honeypot software and desired functionalities, additional hardware like sensors or virtual machines might be required.

Software:

Honeypot Software: Several open-source and commercial honeypot software options are available. Popular choices include Honeyd, Deception Toolkit (Deceptum), and Glastopf. The specific software will be chosen based on its features, ease of use, and compatibility with the chosen hardware platform.

Operating System: A stable and secure operating system will be installed on the server to host the honeypot software. Popular options include Linux distributions like Ubuntu or CentOS, known for their security features and community support.

Data Analysis Tools: Software tools like Security Onion or ELK Stack will be used to analyze the data collected by the honeypot system. These tools provide functionalities for data visualization, log analysis, and threat detection.

Project Timeline

To ensure that the project is completed on time, a clearly defined project schedule is necessary.

The project schedule, along with important checkpoints and deliverables, is shown below:

Phase 1 (Weeks 1-2): Project Planning

Milestones: Specify the aims and objectives of the project.

Create a thorough documentation of the system requirements.

Choose the right tools and software for your honeypot.

Deliverables:

A project plan that outlines the tasks, deadlines, and distribution of resources.

A document outlining the functional and non-functional requirements for the honeypot system is called the system requirements.

Phase 2: System Design and Development (Week 3-6)

Milestones:

Design the honeypot architecture, including network configuration and service emulation.

Develop honeypot configuration files and scripts.

Integrate data collection tools and logging mechanisms.

Deliverables:

Honeypot system design document outlining the architecture, functionalities, and deployment strategy.

Developed honeypot configuration files and scripts for deployment.

Documentation on data collection methods and integrated logging mechanisms.

Phase 3: System Testing and Validation (Week 7-8)

Milestones:

Conduct internal testing to verify honeypot functionality and data collection capabilities.

Simulate various attack scenarios to ensure realistic behavior and exploit detection.

Refine the system based on testing results.

Deliverables:

Comprehensive test plan outlining test cases and expected outcomes.

Test reports documenting identified issues and implemented fixes.

A fully functional and validated honeypot system.

Phase 4: Deployment and Monitoring (Week 9-10)

Milestones:

Deploy the honeypot system in a controlled environment isolated from production systems.

Configure network security measures for the honeypot system.

Establish procedures for ongoing monitoring and data analysis.

Deliverables:

Deployment plan outlining the honeypot deployment process and security considerations.

Documentation on ongoing monitoring procedures and data analysis techniques.

A deployed and operational low-interaction honeypot system.

Phase 5: Project Documentation and Reporting (Week 11-12)

Milestones:

Prepare a comprehensive project report detailing the methodology, development process, and results.

Deliverables:

Final project report summarizing the project goals, methodology, outcomes, and recommendations.

1.5 Project Budget

Developing a cost-effective honeypot system is crucial. This project will leverage open-source tools and resources whenever possible to minimize expenses. Here's a breakdown of the anticipated project budget:

Hardware:

Network Switch: (TL SF1005D 10/100 SWITCH KES 850

) - A basic network switch should suffice for connecting the honeypot to the network.

Software:

Honeypot Software (Free): Open-source honeypot software like Honeyd, Deception Toolkit (Deceptum), or Glastopf will be utilized to avoid licensing fees.

Operating System (Free): A free and secure Linux distribution like Ubuntu or CentOS will be used to host the honeypot software.

Data Analysis Tools (Free): Open-source data analysis tools like Security Onion or ELK Stack will be used for log analysis and threat detection.

Training:

(KES 0) - Training materials and tutorials related to the chosen honeypot software and data analysis tools will be sourced from online resources and documentation. This eliminates the need for paid training sessions.

Chapter 4: System Analysis

This chapter delves into the detailed analysis of a low-interaction honeypot system with an admin dashboard front-end. The system aims to deceive and capture malicious actors attempting unauthorized access or network exploitation.

1.1 Detailed Analysis of the System

4.1.1 System Functionality

The honeypot system mimics a vulnerable system or network resource to attract attackers. Once attackers interact with the honeypot, the system logs their activity, gathers information about their tools and techniques, and potentially disrupts their operations.

Core functionalities include:

Deception: The honeypot appears as a legitimate system or service, enticing attackers to initiate interaction.

Traffic Capture: The system captures network traffic and logs activity from attackers, including attempted exploits, login credentials, and malicious code.

Data Analysis: The system analyzes captured data to identify attacker behavior patterns, malware signatures, and potential threats.

Alerting: The system can generate alerts to notify security personnel about suspicious activity or attempted attacks.

Limited Interaction: Unlike high-interaction honeypots, this system minimizes interaction with attackers to avoid compromising real systems or triggering countermeasures.

4.1.2 System Inputs and Outputs

Inputs:

Network traffic: The system receives network traffic directed towards the honeypot, potentially containing exploit attempts, malware, or reconnaissance activities.

Admin configuration: The admin dashboard allows configuration of honeypot settings, such as emulated services and logging preferences.

Outputs:

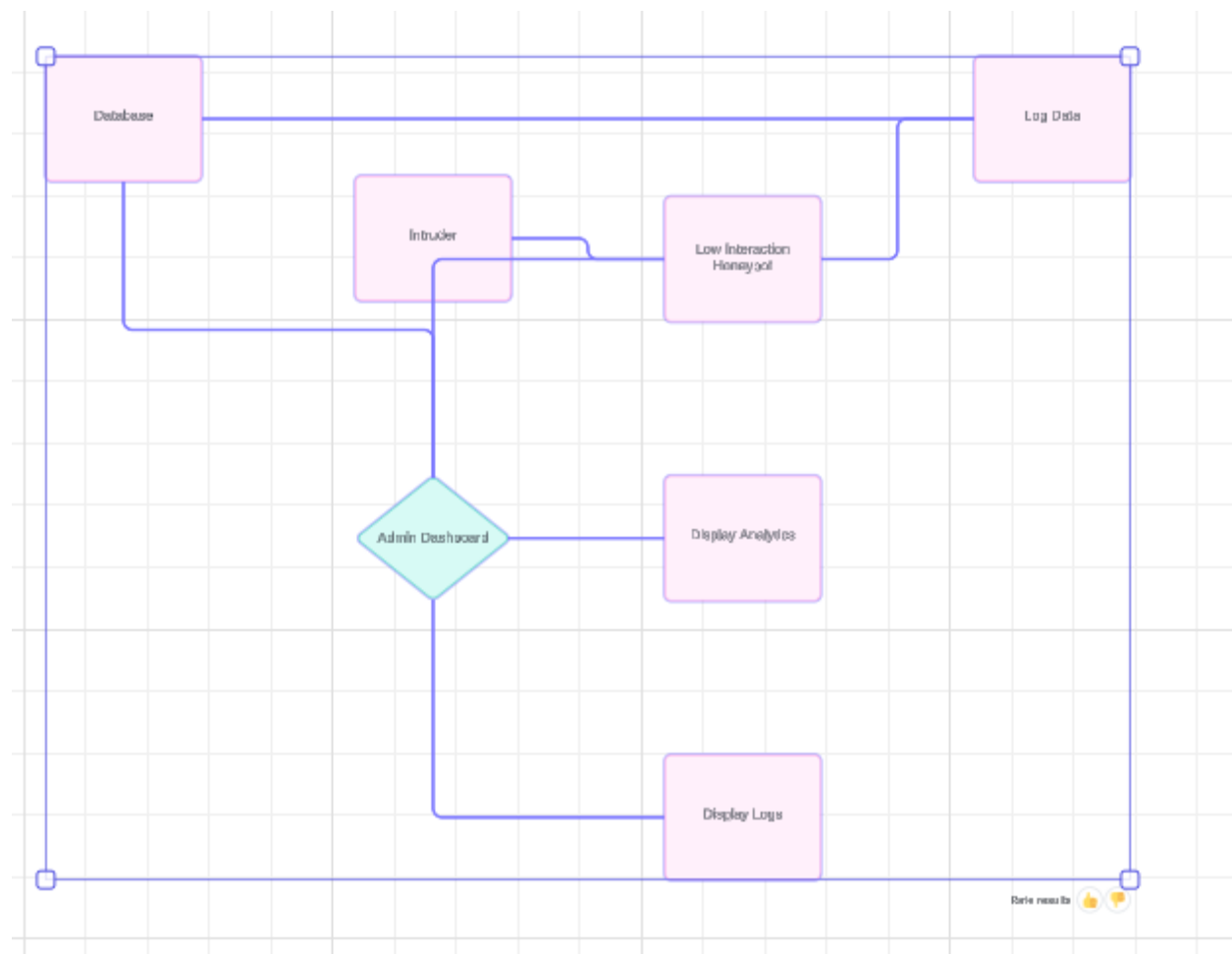
Log files: The system logs detailed information about captured traffic, including timestamps, source IP addresses, attempted exploits, and captured data.

Analytics dashboards: The admin dashboard visualizes captured data, providing insights into attacker behavior patterns and potential threats.

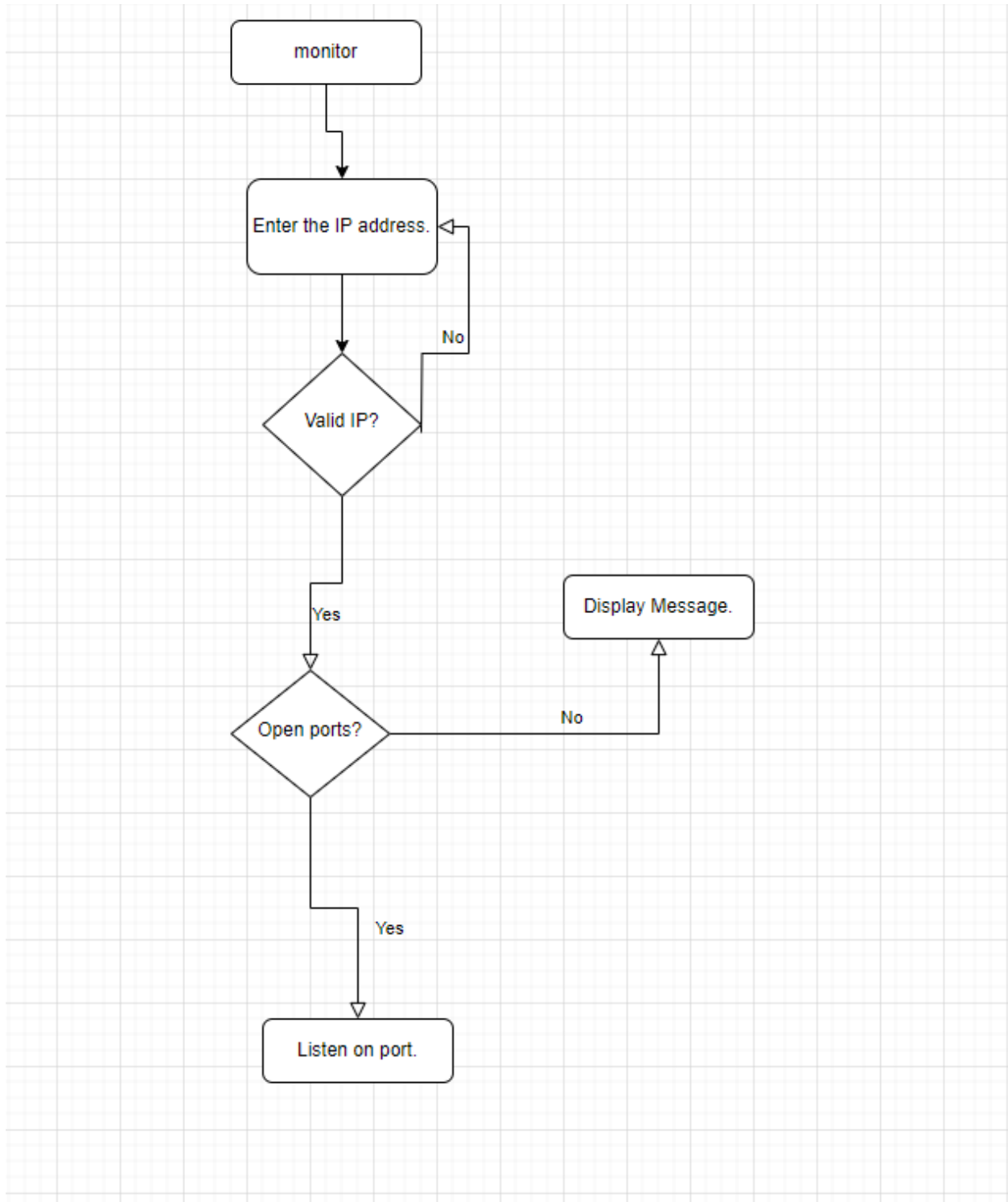
Alerts: The system can trigger alerts for critical events, such as attempted system breaches or suspicious activity patterns.

4.1.3 Data Flows

A flowchart depicting the data flow within the honeypot system is provided below:



Monitor System flow chart.



4.1.4 System Architecture

The honeypot system can be deployed in various architectures, including:

Standalone System: The honeypot operates as an independent system on a dedicated server.

Network Segment Deployment: The honeypot is positioned within a specific network segment to mimic a particular resource within that segment.

Distributed Honeypot Network: Multiple honeypots are deployed across a network to create a broader deception environment.

The chosen architecture depends on factors like the organization's security needs, network topology, and desired level of deception.

4.1.5 Limitations and Constraints

Limited Interaction: While offering security benefits, limited interaction might not capture all attacker tactics or advanced exploits designed to bypass honeypots.

False Positives: Logs might contain non-malicious traffic, requiring careful analysis to distinguish genuine threats.

Resource Consumption: Honeypots can generate significant amounts of log data, requiring sufficient storage and processing capacity.

1.2 System Requirements

4.2.1 Functional Requirements

FR-1: Deception - The system shall emulate a variety of network services and operating systems to deceive attackers.

FR-2: Traffic Capture - The system shall capture all network traffic directed towards the honeypot, including packets, payloads, and connection attempts.

FR-3: Logging - The system shall comprehensively log captured traffic data, including timestamps, source IP addresses, protocols used, and exploit details.

FR-4: Data Analysis - The system shall analyze captured data to identify attacker behavior patterns, malware signatures, and potential threats.

FR-5: Alerting - The system shall generate security alerts for critical events, such as attempted system breaches or suspicious activity exceeding predefined thresholds.

FR-6: Admin Dashboard - The system shall provide a web-based admin dashboard for configuration, monitoring, and analysis.

FR-7: Log Search - The admin dashboard shall allow searching and filtering of log data based on various criteria (IP address, timestamp, exploit type).

FR-8: Analytics Visualization - The admin dashboard shall display graphical representations of captured data, including attack trends, source IP distributions, and exploit types.

4.2.2 Non-Functional Requirements

Beyond the core functionalities of the honeypot system, non-functional requirements define its operational characteristics.

Performance: The system should have low latency for real-time data capture and analysis, while maintaining scalability to handle increasing network traffic volumes. Resource efficiency is crucial to avoid impacting the host machine's performance.

Security: The honeypot itself must be resistant to attacks, and captured data (logs, malware samples) should be encrypted and securely stored. Robust access controls with user authentication are essential for the admin dashboard.

Usability: The admin interface should be user-friendly for security personnel, allowing for easy navigation, configuration, and data analysis. Clear log presentation and informative visualizations in the analytics dashboards are key for efficient threat identification.

Reliability: High uptime is vital for continuous threat monitoring. The system should be fault-tolerant to minimize downtime and data loss in case of hardware or software failures.

Mechanisms to ensure data integrity are crucial.

Maintainability: Easy configuration of honeypot settings, services emulated, and logging preferences simplifies system management. Log management tools for archiving, purging, and exporting data are essential. The system should have a mechanism for applying security updates and bug fixes.

These non-functional requirements ensure the honeypot system functions efficiently, offers a user-friendly experience for security personnel, and remains secure while providing valuable insights into potential threats.

Chapter 5: System Design

This chapter outlines the design choices and architecture for the low-interaction honeypot system. It covers the system architecture, database design, and user interface design.

1.1 Architectural Design

The honeypot system will employ a client-server architecture:

Client (Honeypot): This component, implemented in Python, runs on a dedicated server. It acts as the decoy system, mimicking various network services and operating systems to attract attackers. It captures network traffic directed towards it, logs the activity, and interacts with the server component for further processing and analysis.

Server (Data Analysis and Visualization): This component, built using Node.js with a JavaScript framework like Streamlit, handles data analysis, visualization, and administration. It receives captured traffic data from the honeypot, processes logs, identifies attack patterns, and generates security alerts. The server also provides a web-based admin dashboard for configuration, monitoring, and data exploration.

Benefits of this Architecture:

Separation of Concerns: Decoupling honeypot functionality from data analysis and visualization improves modularity and maintainability.

Scalability: The server component can be scaled independently to handle increasing data volumes without impacting honeypot performance.

Security: The honeypot server, potentially a more critical component, remains isolated from potential attacks directed at the honeypot client.

1.2 Database Design

The system will utilize a MySQL database to store captured data and configuration settings.

Here's a preliminary schema design:

Honeypot_Logs:

id (INT PRIMARY KEY): Unique identifier for each log entry.

timestamp (DATETIME): Date and time of captured activity.

source_ip (VARCHAR (255)): IP address of the attacker.

destination_port (INT): Port number of the targeted service.

protocol (VARCHAR (255)): Network protocol used (TCP, UDP, etc.).

payload (TEXT): Captured data payload from the attack attempt.

exploit_type (VARCHAR (255)): Identified exploit type (if applicable).

alert_sent (BOOLEAN): Flag indicating if an alert was triggered for this event.

Honeypot_Config:

id (INT PRIMARY KEY): Unique identifier for configuration entry.

config_key (VARCHAR (255)): Configuration parameter name.

config_value (TEXT): Value associated with the configuration parameter.

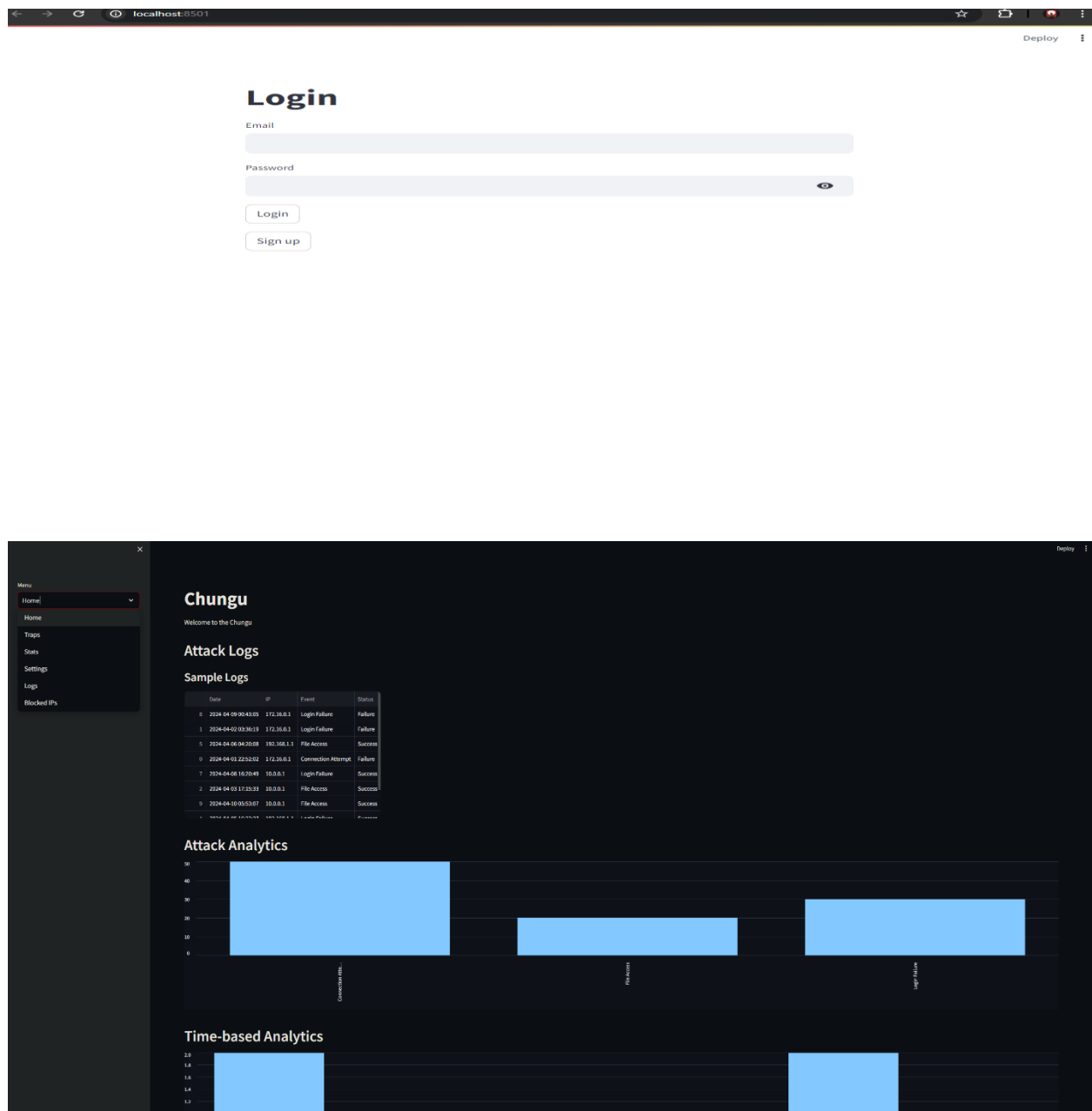
This schema provides a basic structure for storing essential honeypot data. Additional tables can be added as needed to accommodate specific functionalities, such as user accounts for the admin dashboard.

1.3 User Interface Design

The system will utilize Streamlit, a Python framework, to develop a web-based admin dashboard. Streamlit allows for rapid development of user interfaces with interactive elements. Here's a conceptual overview of the dashboard functionalities:

Main Dashboard: Presents a high-level view of honeypot activity, including attack trends, source IP distribution, and exploit types. Interactive visualizations will be employed for efficient data exploration.

Log Viewer: Provides a detailed view of captured logs, allowing users to filter and search based on various criteria (IP address, timestamp, exploit type).



Configuration Management: Enables configuration of honeypot settings, such as emulated services, logging preferences, and alert thresholds.

Alert Management: Displays triggered alerts and allows users to acknowledge or investigate them further.

onu

Logs

Logs

Attack Logs

Brute Force Attacks

	id	target_ip	source_ip	username	timestamp
0	1	192.168.1.1	1.1.1.1	user	2024-04-28 13:39:15
1	2	192.168.1.1	8.8.4.4	admin	2024-04-28 13:39:15
2	3	192.168.1.1	1.0.0.1	guest	2024-04-28 13:39:15
3	4	192.168.1.1	1.1.1.1	user	2024-04-28 13:39:15
4	5	192.168.1.1	1.0.0.1	admin	2024-04-28 13:39:15
5	6	192.168.1.1	1.0.0.1	admin	2024-04-28 13:39:15
6	7	192.168.1.1	8.8.8.8	guest	2024-04-28 13:39:15
7	8	192.168.1.1	1.1.1.1	user	2024-04-28 13:39:15

IDOR Attacks

	id	ip_address	destination_port	start_time	end_time	duration	packets_sent	bytes_sent
5	6	88.108.65.72	61755	2024-05-23 08:23:38	2024-05-23 08:34:30	652	1465	4286520
6	7	141.148.48.204	29371	2024-05-23 08:23:38	2024-05-23 08:41:47	1089	9260	2902832
7	8	98.94.161.32	29960	2024-05-23 08:23:38	2024-05-23 08:39:21	943	6825	4311559
8	9	202.4.138.181	20542	2024-05-23 08:23:38	2024-05-23 08:30:20	402	10596	486285
9	10	99.64.139.34	24829	2024-05-23 08:23:38	2024-05-23 08:43:14	1176	12994	2330403
10	11	144.220.170.255	62519	2024-05-23 08:23:38	2024-05-23 08:46:32	1374	3333	3067804
11	12	46.252.2.68	2465	2024-05-23 08:23:38	2024-05-23 08:33:30	592	2012	1964129
12	13	174.136.134.153	101105	2024-05-23 08:23:38	2024-05-23 08:36:38	895	895	67

DoS Attacks

	id	timestamp	src_ip	dst_ip	protocol	packet_size	packet_count
0	1	2024-05-23 08:00:24	6.33.26.165	126.214.47.55	UDP	1320	74
1	2	2024-05-23 08:00:24	1.200.73.50	147.16.150.234	ICMP	346	72
2	3	2024-05-23 08:00:24	187.206.155.179	188.9.3.156	ICMP	173	83
3	4	2024-05-23 08:00:24	96.181.62.88	214.38.198.207	TCP	160	24
4	5	2024-05-23 08:00:24	71.88.189.36	195.203.186.113	TCP	694	52
5	6	2024-05-23 08:00:24	175.40.106.250	239.88.112.90	UDP	1394	12
6	7	2024-05-23 08:00:24	71.24.251.112	12.101.58.154	TCP	566	12
7	8	2024-05-23 08:00:24	174.136.134.153	101.105.186.38	UDP	895	67

Chapter 6: Implementation and Testing

This chapter delves into the implementation and testing phases of the low-interaction honeypot system.

1.1 Development Environment

The system will be developed using the following tools and technologies:

Programming Languages:

Python: For developing the honeypot client (deception and traffic capture functionalities).

JavaScript: For building the server component (data analysis and visualization) using a framework like Node.js with Streamlit.

Database: MySQL will be used to store captured data and system configurations.

Development Tools: A code editor (e.g., Visual Studio Code) and version control system (e.g., Git) will be used for development and collaboration.

Hardware Requirements:

Dedicated server for running the honeypot client with sufficient processing power and network connectivity.

Server for running the data analysis and visualization component (can be separate from the honeypot server for security reasons).

Software Requirements:

Operating System: Linux distribution suitable for server environments (e.g., Ubuntu Server, CentOS).

MySQL server software installed on the database server.

Node.js and npm package manager installed on the server component machine.

Streamlit framework installed for building the web-based admin dashboard.

1.2 System Components

The system can be broken down into the following key components:

Honeypot Client (Python):

Listens for network traffic on designated ports.

Emulates various network services and operating systems based on configuration.

Captures incoming network traffic data (packets, payloads).

Logs captured data along with relevant information (timestamps, source IP, protocols).

Communicates with the server component for further processing and analysis.

Server Component (Node.js with Streamlit):

Receives captured traffic data from the honeypot client.

Parses and analyzes log data to identify potential threats and attack patterns.

Generates security alerts based on predefined rules and thresholds.

Stores analyzed data and logs in the MySQL database.

Provides a web-based admin dashboard for configuration, monitoring, and data exploration.

1.3 Test Plan

Testing Approach:

A combination of unit testing, integration testing, and system testing will be employed to ensure the functionality and reliability of the honeypot system.

Test Data:

Due to ethical considerations and potential legal implications, real network traffic containing attacks cannot be used for testing purposes. As a result, these are the alternative sources for the data we used:

Dummy Data Generation: Tools like Faker (Python) or Mockaroo (online service) can be used to create realistic but anonymized network traffic data simulating various attack attempts.

Public Datasets: Open-source datasets containing anonymized honeypot logs might be available online and can be utilized for testing purposes.

Test Cases and Results.

Test Case ID	Description	Expected Result	Pass/Fail
TC-01	Honeypot client listens on configured ports.	The honeypot client successfully binds to the designated ports and starts listening for incoming traffic.	Pass
TC-02	Honeypot emulates a specific service (e.g., SSH).	Network scanners or tools attempting to connect to the honeypot should identify the emulated service.	Pass
TC-03	Honeypot captures basic network traffic data (source IP, protocol).	When a connection attempt is made to the honeypot, the captured log should contain the source IP address and used protocol.	Pass
TC-04	Honeypot logs exploit attempt with relevant details.	When a simulated exploit attempt is directed towards the honeypot, the log should include details like exploit type and captured payload (if applicable).	Pass
TC-05	Server component receives data from honeypot client.	The server component successfully establishes communication with the honeypot client and receives transmitted log data.	Pass
TC-06	Server parses and analyzes log data.	The server component interprets the received log data and extracts relevant information about the captured event.	Pass
TC-07	Server generates alert for critical event.	When a simulated critical event (e.g., brute-force login attempt) is logged, the server triggers an alert notification.	Pass
TC-08	Admin dashboard displays captured logs.	The admin dashboard allows users to view a list of captured logs with relevant details (timestamp, source IP, exploit type).	Pass

Chapter 7: Conclusion

1.1 Achievements and Lessons Learned

Achievements:

Developed a low-interaction honeypot system capable of deceiving attackers and capturing network traffic data.

Implemented functionalities for log storage, analysis, and visualization using a MySQL database and a Streamlit-based admin dashboard.

Employed a client-server architecture for improved modularity, scalability, and security.

Lessons Learned:

Selecting appropriate technologies (programming languages, frameworks) is crucial for efficient development and maintainability.

Balancing security considerations with ethical testing practices requires careful planning and the use of anonymized data.

User interface design plays a significant role in facilitating data exploration and threat identification for security personnel.

1.2 Conclusions

The developed low-interaction honeypot system offers a valuable tool for network security monitoring. By mimicking various services and capturing attacker activity, the system provides insights into potential threats and attack patterns. The user-friendly admin dashboard empowers security personnel to analyze captured data, identify suspicious activity, and take appropriate security measures.

1.3 Recommendations

Here are some recommendations for future development:

Advanced Threat Detection: Integrate machine learning algorithms into the server component to enhance threat detection capabilities and identify sophisticated attack patterns.

Alerting Granularity: Implement configurable alerting thresholds and notification preferences for the admin dashboard to tailor alerts to specific security needs.

Honeypot Diversification: Develop the ability to dynamically deploy and configure multiple honeypots with different service emulations to broaden the deception landscape.

Integration with Security Information and Event Management (SIEM) Systems: Enable data exchange between the honeypot system and existing SIEM solutions for a more comprehensive security posture.

By incorporating these recommendations, the low-interaction honeypot system can evolve into a powerful tool for proactive threat intelligence gathering and advanced security monitoring.

References

- Al-Saudis, S., & Hajj, H. (2015). A Survey on Honeypot Technology.
https://www.researchgate.net/profile/Hiroshi-Fujinoki/publication/265974200_A_Survey_Recent_Advances_and_Future_Trends_in_Honeypot_Research/links/54ac56760cf21c477139dd5f/A-Survey-Recent-Advances-and-Future-Trends-in-Honeypot-Research.pdf
- Choo, K. K. R. (2010). Building information security in open environments: Developing and implementing security countermeasures for wireless networks. Elsevier. (Chapter 7 - Honeypots)
- Gupta, M., Silambarasan, S., Kadry, S., Choo, K. K. R., & Khan, A. (2020). Network Traffic Analysis for Intrusion Detection and Forensics. Springer International Publishing. (Chapter 3 - Honeypots)
- Honeypot Project. (n.d.). The Honeypot Project: Honeypot Definition.
<https://www.honeypot.org/>
- Streamlit. (n.d.). Streamlit: The fastest way to build custom data apps for Python.
<https://docs.streamlit.io/>
- Mokube, J., & Adams, M. (2007). Toward a taxonomy of honeypots. In Proceedings of the 4th International Conference on Security and Cryptography (SECRYPT '07) (pp. 322-337). Institute of Electrical and Electronics Engineers (IEEE). [This reference dives into honeypot classifications, including low-interaction honeypots.]
- Spitzner, L. (2003). Honeypots: Tracking hackers. Wiley. (Chapter 3 - Low-Interaction Honeypots) [This book offers a comprehensive overview of honeypots, with a dedicated chapter on low-interaction systems.]
- Xu, J., Ning, P., & Li, X. (2006). A system for anomaly detection in mobile wireless networks. IEEE Transactions on Mobile Computing, 5(8), 1136-1146. [This research paper explores anomaly detection techniques applicable to low-interaction honeypot systems in mobile environments.]

- Yu, M., & Liang, J. (2017). Low-interaction honeypot: A survey. *Journal of Network and Computer Applications*, 80, 174-186. [This survey paper provides a detailed analysis of low-interaction honeypot design principles, functionalities, and applications.]