Intrusion Detection System based on Raspberry Pi Honeypot and Machine Learning for Home Network

by

Phakonekham Phichit - 001041931

May 9, 2024

Abstract

This study presents a research honeypot intrusion detection system based on a Raspberry Pi 4B device, a cloud-based ELK stack system, and machine learning classification algorithms for SSH and Telnet services in a home network. The system aims to detect unauthorized access attempts by creating a decoy environment that emulates legitimate services. The project aims to improve the implementation by (Jeremiah 2019).

The system collects data on any unauthorized access attempts and processes it using machine learning classification algorithms to identify patterns of malicious activity. The system features discord alerts and data visualization capabilities provided by the ELK stack system, enabling users to quickly respond to potential threats.

In conclusion, the research honeypot intrusion detection system provides a robust and effective solution for detecting and preventing cyber-attacks. By combining honeypot technology, machine learning classification algorithms, and monitoring capabilities, the system enhances the security of computer systems and contributes to the advancement of computer security research. The system's features, including alerts and data visualization capabilities, enable users to make informed decisions to improve the overall security of their systems.

Contents

List of Figures

List of Tables

1 Introduction

Intrusion detection systems (IDS) have become an essential component in ensuring the security of home networks as they provide continuous monitoring of network activities, identify potential threats in real-time, and enable users to take appropriate action. This study presents a research honeypot intrusion detection system that utilizes a Raspberry Pi 4B device, a cloud-based ELK Stack system, and machine learning classification algorithms for SSH and Telnet services. The primary objective of this project is to build upon the implementation proposed by (Jeremiah 2019) and deliver an enhanced solution to detect unauthorized access attempts in home networks more effectively.

Honeypot technology plays a critical role in this research, as it creates a decoy environment that emulates legitimate services to attract potential attackers. This approach facilitates the collection of valuable data on unauthorized access attempts, which can subsequently be processed using machine learning classification algorithms. Employing machine learning techniques significantly improves the intrusion detection capabilities of the system by identifying patterns of malicious activity and adapting to emerging threats more efficiently than traditional rule-based systems.

To further enhance the system's effectiveness, this project incorporates the ELK Stack system, hosted on a cloud-based server(Mohd Ali, Mohd Salleh, and Zulkipli 2021). This integration offers real-time monitoring and data visualization capabilities, enabling users to obtain insights into network activities and potential threats. Furthermore, the system features Discord alerts, allowing users to receive timely notifications of potential security breaches and take prompt action(Joyon 2022).

The research honeypot intrusion detection system seeks to provide a comprehensive solution for detecting and mitigating cyber-attacks by integrating honeypot technology, machine learning classification algorithms, and monitoring capabilities. The system aims to enhance the security of home networks and contribute to the advancement of computer security research. Furthermore, the system's features, including alerts and data visualization capabilities, facilitate informed decision-making and reinforce the overall security of user networks.

This dissertation focuses on the development of a research honeypot intrusion detection system specifically designed for home networks. By combining the capabilities of the Raspberry Pi 4B device, the ELK Stack system, and

machine learning algorithms, the system aims to create a robust and effective security solution. The project encompasses various aspects, including the design, implementation, and evaluation of the proposed system. Throughout the research process, the study also considers the potential challenges and limitations associated with the deployment of the system, as well as the potential implications of its findings for both academic and practical applications.

2 Literature Review

2.1 Existing Intrusion Detection System

This article by (Pashaei et al. 2022) presents a novel approach to early intrusion detection systems (IDS) using honeypots and reinforcement learning for industrial control networks. The proposed method has certain advantages but also faces limitations and critiques. One limitation is the reliance on accurate data collection and analysis, which, if compromised, may lead to false positives or negatives. Additionally, the computational complexity of the SARSA algorithm might hinder scalability for larger networks or real-time applications.

Moreover, the evaluation results, though showing high accuracy and low false positive rates, were conducted on a specific dataset, raising concerns about generalising to other datasets and real-world applications. Honeypots, while helpful in attracting attackers, may also pose security risks if not secured properly.

Khraisat and Alazab 2021 highlights that traditional IDS techniques, such as signature-based detection and anomaly detection, have limitations in detecting new or unknown attacks and adapting to changing attack patterns. Machine learning algorithms can improve IDS accuracy by identifying patterns and behaviours indicative of attacks. These algorithms are more flexible and adaptive, as they do not rely on predefined rules, and can handle large amounts of data. However, challenges remain, such as the need for extensive training data and the potential for false positives and negatives if not trained properly.

(Fadhilah and Marzuki 2020) investigates the performance of IDS Snort and IDS Suricata against DoS/DDoS attacks in a virtual machine environment. The results indicate that both systems effectively detect DoS/DDoS

attacks, with IDS Suricata outperforming IDS Snort in terms of packet capture per second (PCPS) and packet size per second (PSPS). This difference in performance may be attributed to IDS Suricata's support for multithreading. However, the tests were conducted on a single virtual machine, which may not accurately represent real-world scenarios and does not provide information about performance against other types of attacks or combinations of attacks.

(Gassais et al. 2020) proposes a cloud-based approach to host-based intrusion detection for smart devices, arguing that traditional host-based solutions are unsuitable for smart devices due to their limited resources. The proposed architecture sends data to a cloud server for processing and analysis, which then returns alerts and responses to the device. The authors evaluated the efficiency and overhead of their solution in a controlled environment, but the results may not be representative of real-world scenarios.

(Gassais et al. 2020) also discusses lightweight solutions for intruder detection but notes that these tools may not be compatible with smart object hardware such as ARM CPUs. The authors suggest using a honeypot to collect malware samples targeting the monitored device, though further details on the implementation are needed.

In summary, research into honeypots, reinforcement learning, machine learning algorithms, and cloud-based approaches provide insights into improving intrusion detection systems. However, these methods face challenges in data collection, computational complexity, and real-world applicability. Which is appropriate for this project regarding the implementation of honeypot and machine learning.

2.2 Honeypot-Based Intrusion Detection System

The original inspiration for this project was the paper by(Jeremiah 2019). The paper provides a design system which utilised a Raspberry Pi-Honeypot using current existing Kali Linux tools and methods such as Snort, Modern Honeypot Network (MHN), Kippo, Dionaea, and Blastoff. This architecture is widely adopted because of its simplicity and cost-effectiveness deployment in any environment. Due to the limitation of traditional instruction detection systems that are not incapable to protect against sophisticated cyber-attacks, using a honeypot could extra security layer of defence as it creates a decoy system to attract potential attackers. However, the researchers do not provide any evaluation and performance of the system, hardware requirements and

challenges when deploying the system. This could lead to more difficulties when implementing the system. Other similar systems, on paper by (Neranjan Thilakrathne, Samarasinghe, and Priyashan 2021), the study include and evaluates the use of honeypots for analyzing cyber threats targeting IoT devices. The researchers set up a Raspberry Pi honeypot device and kept its SSH connection service open for over six consecutive days to log all the requests coming from the outside internet. During this time, the honeypot received 105,764 login attempts, with the vast majority of requests coming from four sources globally that accounted for around 85% of all requests. The study did not provide an exact success rate for login attempts.

In this paper by (Palša et al. 2022), The paper under consideration provides a comprehensive overview of honeypot configuration and redundancy implementation, two vital components of maximizing uptime and gathering critical information in cybersecurity. Through system simulations, the authors tested honeypot efficacy and redundancy's impact on ensuring maximum uptime, as well as employing attack simulations and network scans to compare honeypots' ability to generate attacker information.

Honeypots offer several advantages, as noted by the authors, including the provision of valuable information on attackers and their tactics, enabling better defence against future attacks. Honeypots are also flexible and can detect new attack tools and methods, making them a powerful tool for cybersecurity. However, implementing redundancy requires careful consideration of the costs and benefits, as it may necessitate additional hardware or software resources.

(Palša et al. 2022), employed Honeyd and Cowrie honeypots to interact with low-level and mid-level attackers, respectively, discussing the honeypots' configuration in detail, including their setup to attract attackers and record their activity. The authors also implemented redundancy through the Heartbeat system, which offers cluster infrastructure services. One limitation of the study is that it only tested two types of honeypots, making it unclear how other honeypots would fare in detecting specific types of attacks. Nonetheless, this paper offers a useful overview of configuring honeypots with redundancy to enhance network security. The authors' findings show that implementing redundancy can significantly improve uptime for honeypot systems, a crucial factor in cybersecurity.

It is well-known that an experienced hacker could identify honeypot (Jeremiah 2019), (Cabral et al. 2021) aimed at augmenting the functionality of the Cowrie honeypot, a widely utilized open-source tool for identifying and an-

alyzing cyber attacks. To achieve this goal, they developed a framework that facilitates the customization of the Cowrie honeypot's configurations, thereby enhancing its ability to deceive and mimic reality in the face of online threats. The researchers accomplished this by modifying the honeypot configurations, a process that entailed analyzing its various artefacts and subsequently altering them to improve its overall deceptive capacity. By comparing default and configured deployments, they were able to demonstrate the efficacy of the modified Cowrie honeypot in gathering crucial information from attackers, including an increase in the number of distinct IP connections, the types of commands executed, files downloaded, and VirusTotal submissions. The primary benefit of this research is its provision of a practical solution for bolstering the effectiveness of Cowrie honeypots, equipping organizations with the means to better detect and analyze cyber-attacks. However, this approach may require additional resources and expertise for its successful implementation. Similar projects, by (Saputro, Purwanto, and Ruriawan 2021) also employed Cowrie honeypot to protect Internet of Things devices from malware attacks or various attack patterns, while also gathering data about the attacker's system. The analysis of the results demonstrates that the honeypot can document all attempts and attack behaviours, with an average CPU load of less than 6.3%.

A study by (Sethi and Mathew 2021) proposes different implementation and model solutions on the honeypots. which goes through the level of interaction, deployment, cost, scalability, solution, and the process of IDS and applications. The author proposes a dynamic honeypot which combines a blockchain system as the basis of the solution with the Kippo honeypot and user authorization. This is a similar honeypot to Cowrie with SSH implementation for a medium-high level honeypot interaction (Başer, Güven, and Aydın 2021).

(Mohd Ali, Mohd Salleh, and Zulkipli 2021) presents an empirical investigation of cloud honeypots and data visualization employing the ELK stack. The authors suggest a strategy for identifying, mitigating, and reporting SSH attacks in the realm of cloud computing. The ELK stack is utilized to visualize data gathered from the honeypot, offering insights into possible security vulnerabilities. The findings indicate that the proposed approach effectively detects and protects against SSH attacks. The proposed design is limited to detecting and preventing SSH attacks only. Other types of attacks may require different approaches. Additionally, the study was conducted using only one honeypot, which may not be representative of all cloud environments.

(Morozov et al. 2023) addresses the implementation of honeypots and cyber deception as methods for identifying cyber attacks targeting critical infrastructure in Ukraine. The authors emphasize the importance of devising efficient strategies to combat emerging threats while offering insights into the application of these instruments. Additionally, the article notes the consistent increase in cyber threats and the repercussions of Russia's 2022 incursion into Ukraine on international cybersecurity. The most common limitations of a traditional honeypot where the system could be resource intensive and when it comes to experiencing attackers, it is easy to be detected. Therefore for a better solution, including machine learning implementation is also a possible solution to minimise false positives. Other similar papers by (Zobal, Kolář, and Fujdiak 2019) state that the challenges with honeypots are detection prevention, deployment and visualisation.

(Khan and Abbasi 2020)incorporate honeypots into the network, distributing them both homogeneously and randomly. These honeypots will collaborate with potentially malicious nodes and observe their behaviour. According to simulation results, this technique reduces the error rate in existing IoT IDS; however, it does consume additional energy. This trade-off between energy consumption and detection accuracy is examined by considering standard routing and MAC protocols for the IoT network.

2.3 Application of Raspberry Pi in Network Security

The Raspberry Pi is a small, low-cost, and versatile single-board computer (SBC) that has gained significant popularity among hobbyists, educators, and technology enthusiasts (Hinov and Krastev 2022). Its affordability, accessibility, and ease of use have led to a wide range of applications, including its use in network security as a platform for implementing intrusion detection and prevention systems (IDS/IPS), firewalls, and honeypots.

Several studies have demonstrated the feasibility of using Raspberry Pi as a platform for implementing IDS/IPS systems. For instance, (Ramakrishnan, Gokul, and Nigam 2021) developed a Raspberry Pi-based IDS using Snort, an open-source IDS, and found that the system was effective in detecting various types of network attacks while consuming minimal system resources. Similarly, Hariawan and Sunaringtyas 2021 implemented an IPS on Raspberry Pi using Suricata, another open-source IDS/IPS, and demonstrated that it could successfully prevent several types of network intrusions with minimal impact on network performance.

Raspberry Pi has also been used as a platform for deploying honeypot-based intrusion detection systems. A honeypot is a decoy system designed to attract and analyze cyber-attacks, enabling security researchers to study attackers' tactics, techniques, and procedures (Saputro, Purwanto, and Ruriawan 2021). Raspberry Pi-based honeypots are particularly well-suited for home networks and SMEs due to their low cost and ease of deployment. Numerous studies have employed Raspberry Pi to implement honeypots, such as Cowrie, a popular SSH and Telnet honeypot (Saputro, Purwanto, and Ruriawan 2021), and Honeyd, a more general-purpose honeypot that simulates various services and network devices(Palša et al. 2022).

Despite its advantages, there are some limitations to using Raspberry Pi in network security applications. Firstly, the Raspberry Pi's limited processing power and memory resources may co restrain its ability to handle high volumes of network traffic or complex security tasks, potentially impacting the overall performance and effectiveness of the security system (Martin, Kargaard, and Sutherland 2019).

2.4 Machine Learning Techniques for Honeypot

Several ML techniques have been employed in the context of honeypot-based IDS, including supervised learning algorithms such as Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), and Random Forests (RF), as well as unsupervised learning algorithms like clustering and anomaly detection (Jiang and Zheng 2020)(TAŞÇI et al. 2021). These algorithms have been applied to various aspects of honeypot data analysis, such as classifying attack types, detecting attack patterns, and identifying malicious network traffic.

Support Vector Machines (SVM) have been widely used in the analysis of honeypot data due to their ability to perform well in high-dimensional feature spaces and handle both linear and non-linear relationships between features (Huang et al. 2019). SVM-based approaches have been employed to classify different types of attacks and identify attack patterns in honeypot data (Kristyanto et al. 2022).

k-Nearest Neighbors (k-NN) is another popular supervised learning algorithm used in the context of honeypot-based IDS. k-NN is a simple, non-parametric algorithm that can be used for classification and regression tasks (Li, Hong, and Yu 2020). It has been employed to identify various types of attacks in honeypot data, such as port scans, DDoS attacks, and malware

distribution (Li, Hong, and Yu 2020).

Random Forests (RF) is an ensemble learning method that combines multiple decision trees to improve classification and regression performance. RF has been successfully applied to honeypot data analysis, showing robust performance in detecting and classifying cyber-attacks (Lee et al. 2021). One of the advantages of RF is its ability to handle large datasets and high-dimensional feature spaces, as well as its robustness to noise and overfitting (TAŞÇI et al. 2021). However, RF models can be computationally expensive, particularly when using a large number of trees, which may limit their applicability in resource-constrained environment (TAŞÇI et al. 2021).

Unsupervised learning algorithms, such as clustering and anomaly detection, have also been employed in the analysis of honeypot data. Clustering techniques, like k-means and DBSCAN, have been used to group similar attack patterns or behaviours, allowing security professionals to gain insights into the structure and relationships within the collected data (Sumanth and Bhanu 2020).

Additionally, the quality and representative of the training data can significantly impact the performance of ML models, making it crucial to ensure that the data used for training is diverse and representative of real-world attack scenarios (Ariffin et al. 2022).

Lastly, the interpretability and explainability of ML models can be challenging, particularly for complex models such as deep learning-based approaches(Dowling, Schukat, and Barrett 2020). Developing more interpretable and explainable ML models for honeypot-based IDS remains an active area of research.

2.5 The Role Of Honeypots In Securing Home Networks

Moreover, a key advantage of utilizing Raspberry Pi as a platform lies in the simplicity of implementing sophisticated security tools, such as honeypots. Cyber-attack trends are just emerging, and this project offers an effective method for integrating multiple security tools to reduce attacks and enhance system security(Tripathi and Kumar 2018). In summary, cost-effective network security solutions with straightforward implementation hold significant potential in the current market. A low-powered, portable network security device can be an exceptionally effective tool for ensuring online security.

Moreover, honeypots can serve as an early warning system for home networks, allowing daily users to identify and respond to potential threats before they cause significant damage. By monitoring honeypot activity, security professionals can identify trends and patterns in cyber-attacks, enabling them to proactively update security measures and improve the overall security posture of the home network (Bontchev and Yosifova 2019). In addition, SSH (Secure Shell)remote connections can be dangerous if not properly secured and managed, as they may expose systems to various security risks. The OWASP Top 10 is a list of the most critical web application security risks, and several of these risks are relevant to SSH remote connections(Yadav 2020).

3 Analysis

This analysis section aims to evaluate the existing literature review on honeypot intrusion detection systems (IDS) and identify gaps in knowledge and potential areas for improvement. The literature review revealed several key areas of focus, including the effectiveness of honeypot-based IDS, the application of Raspberry Pi devices, and the role of machine learning techniques in enhancing detection capabilities.

3.1 Evaluation of Previous Honeypot Intrusion Detection Systems

One notable gap in the literature is the limited focus on the specific challenges and requirements of securing home networks. Although several studies have explored the use of honeypots in research environments, the unique challenges associated with home networks, such as limited resources and diverse user behaviours, warrant further investigation. Additionally, the increasing prevalence of Internet of Things (IoT) devices in home networks raises new concerns for security, as these devices often lack robust security measures and can serve as entry points for attackers(Khan and Abbasi 2020). Research on the integration of honeypot-based IDS with IoT devices could provide valuable insights into protecting home networks against a wider range of threats.

Another gap in the literature is the lack of comprehensive evaluation and comparison of different machine learning algorithms for intrusion detection. While some studies have investigated the use of machine learning in honeypot-based IDS, there is still a need for a systematic evaluation of various algorithms to identify the most suitable approach for different scenarios and network conditions. Furthermore, the literature could benefit from a more in-depth analysis of the feature selection and preprocessing techniques used in conjunction with machine learning algorithms, as these can significantly impact the accuracy and efficiency of intrusion detection.

Finally, the research on the integration of various components in a honeypot-based IDS, such as the Raspberry Pi device, machine learning algorithms, and monitoring tools like the ELK Stack, shows a more holistic approach to system design and implementation could help improve the overall performance and effectiveness of honeypot-based IDS. For example, investigating methods for automating the deployment and configuration of honeypots, as well as the efficient sharing of data between system components, could help

streamline the IDS deployment process and minimize potential configuration errors(Hariawan and Sunaringtyas 2021).

Based on the identified gaps in the literature review, several potential areas for improvement can be proposed. First, future research should focus on the unique challenges and requirements of securing home networks using honeypot-based IDS. This could involve investigating the suitability of low-cost, resource-efficient hardware such as Raspberry Pi devices for home network security and exploring ways to optimize system performance in resource-constrained environments. Additionally, research should examine the integration of honeypot-based IDS with IoT devices to better protect home networks from a broader range of threats.

A systematic evaluation of various machine learning algorithms for intrusion detection should be conducted. This would help identify the most suitable algorithms for different network conditions and honeypot configurations, ultimately improving the accuracy and efficiency of honeypot-based IDS. Furthermore, a more comprehensive analysis of feature selection and preprocessing techniques used in conjunction with machine learning algorithms is necessary to optimize the accuracy and efficiency of intrusion detection. The only studies that have implemented a deep analysis of preprocessing are (Haseeb et al. 2020) and (Başer, Güven, and Aydın 2021). This is the same for training time and computation too for IOT devices.

3.2 Machine Learning Algorithms Suitable For Honeypots

In reference to (Kristyanto et al. 2022) and (Li, Hong, and Yu 2020), a segment of the report, covers the machine-learning algorithms, that are suitable for this project. After extensive research, it is believed that Support Vector Machines, K-Nearest Neighbors and Random Forest will be machine learning algorithms for this project.

Support Vector Machines (SVM) is a powerful classification technique that finds the optimal hyperplane separating the classes in the feature space. It is particularly effective in high-dimensional spaces and can be used with different kernel functions to accommodate non-linear relationships between features.

K-Nearest Neighbors (KNN) is a distance-based classification method that assigns a class to an instance based on the majority class of its k near-

est neighbours in the feature space. It is a simple and intuitive method, but its performance may degrade with increasing dimensions due to the curse of dimensionality.

Random Forest (RF) is an ensemble method that combines multiple decision trees to make more accurate and robust predictions. It reduces the risk of overfitting associated with individual decision trees by averaging their predictions.

3.3 Data sets

Based on the datasets used in the literature review, the recommended dataset for this project will be unique. A dataset that is vital to this project is not publicly available. Websites, like Kaggle, reserve datasets that were either too high levelled or too low-level for the level that is required in this project. This means that the dataset attained in this project will have to be seized from the cowrie log files. The cowrie log files record the incoming traffic that occur on the honeypot and will only collect the information that is needed. The desired data collected will be needed for the development of the machine-learning algorithm model which will identify malicious attacks.

4 Requirements Specification

This section is dedicated to listing all the features and testing which will be required to complete the project to a satisfactory level. These are requirements from Design, Implementation, Testing and Integration. (YES = completed and NO = not needed/not implemented.)

- Testing
  - Honeypot effectiveness: Is the Raspberry Pi Honeypot able to detect and collect potential malicious attacks? (YES)
  - Is the honeypot able to verify different types of SSH attacks from the honeypot outside of port scanning, DDOS attacks and Brute-force Attacks? (NO)
  - If not, how high are the hardware requirements optimal enough to run the honeypot? (NO)
  - From the chosen classification techniques, which have the most effective result regarding speed and accuracy? Random Forest
- Features
  - Honeypot deployment: The system must be able to deploy a honeypot on the Raspberry Pi 4B device to emulate legitimate services and attract potential attackers. This will enable the collection of valuable data on intrusion attempts and enhance network security. (YES)
  - Data collection and preprocessing: The system should be capable of collecting data on unauthorized access attempts and preparing it for analysis by machine learning algorithms. Preprocessing may include cleaning, normalization, and feature extraction to ensure high-quality input data for model training. This is through Linux command extraction or STFP file transfer. As for preprocessing, the normalisation is done using a Python script (YES)
  - ELK Stack integration: The system must incorporate ELK Stack for real-time monitoring, data visualization, and log analysis, providing administrators with actionable insights and enhancing security management. (YES)

  - Discord alert notifications: The system should send notifications to users through Discord when potential security breaches are detected, ensuring timely communication and promoting swift responses to potential threats. (YES)

- Additional Features
  - Automate data extraction GUI from Raspberry Pi honeypot (NO)
  - Real-time Machine Learning IDS (NO)

5 Design

This section outlines the system architecture, objectives, features, technologies, and potential challenges of the honeypot system. In this section, the design considerations that guided the development of the honeypot system are described, including the selection of appropriate technologies, the overall system architecture, and the planning of system features. The objective of this section is to provide a comprehensive understanding of how the honeypot system was designed to achieve its intended goals.

An overview of the overall system architecture is provided, which includes the Raspberry Pi and the various software components that make up the honeypot system. This encompasses the features and functionalities of the honeypot system, such as the ability to collect and reprocess data, train machine learning models, and send alerts via the Discord app.

Additionally, the selection of technologies used in the development of the honeypot system is discussed, including the ELK Stack, a suite of open-source tools used for log management and data visualization. Finally, potential challenges and limitations of the honeypot system are highlighted, and an explanation of how they were addressed during the design phase is provided. By presenting this information in the Design section, a clear and comprehensive overview of the honeypot system's design is offered, which forms the basis for the system's implementation and evaluation

5.1 Honeypot Network Topology

Due to the hardware limitations of the Raspberry Pi, which is the core component of the honeypot system in this project, incorporating resource-intensive open-source intrusion detection systems (IDS) like Suricata or Snort may not be feasible. As mentioned in the research by (Fadhilah and Marzuki 2020), these IDS tools typically require at least 3GB of memory, making them unsuitable for a Raspberry Pi-based solution. Instead, a more efficient approach for this implementation is to employ ElastAlert, a powerful open-source alerting tool designed for the Elasticsearch platform.

ElastAlert, as described by (Hermawan, Novianto, and Octavianto 2021), is a versatile tool capable of monitoring data streams and applying custom rules to generate notifications based on user-defined thresholds and patterns. This flexibility enables the development of a tailored intrusion detection solution that effectively monitors and responds to events within the Elasticsearch

environment. By integrating ElastAlert with Discord, real-time notifications can be received on both system and mobile devices, ensuring immediate detection and response to suspicious or malicious activities.

The proposed honeypot system is built on a Raspberry Pi device running the Cowrie honeypot, which is then integrated with a cloud-based ELK Stack server for data visualization. The honeypot is deployed within a demilitarized zone (DMZ) on a home router, increasing its attractiveness to potential attackers(Hariawan and Sunaringtyas 2021). The network topology, as illustrated in Figure 1, has been designed to assess the performance of the honeypot in a home network setting.

By utilizing the Raspberry Pi in conjunction with the selected software tools, the honeypot system delivers an efficient and cost-effective solution for home network security. The Cowrie honeypot is a robust and well-established tool capable of emulating SSH and Telnet services, making it an ideal choice for capturing and analyzing malicious activity. Elasticsearch, Logstash, and Kibana (the ELK Stack) provide a comprehensive suite of tools for log management, data processing, and visualization, enabling the monitoring and assessment of security threats in real-time.

Furthermore, the use of Filebeat and Packetbeat ensures the efficient and reliable collection of log and network data, which is essential for the honeypot system's operation. Filebeat functions as a lightweight log shipper, while Packetbeat captures and analyzes network traffic, providing valuable insight into the behaviour of attackers. Nginx serves as a secure and reliable web server, allowing for easy access to the Kibana dashboard and other web-based services.

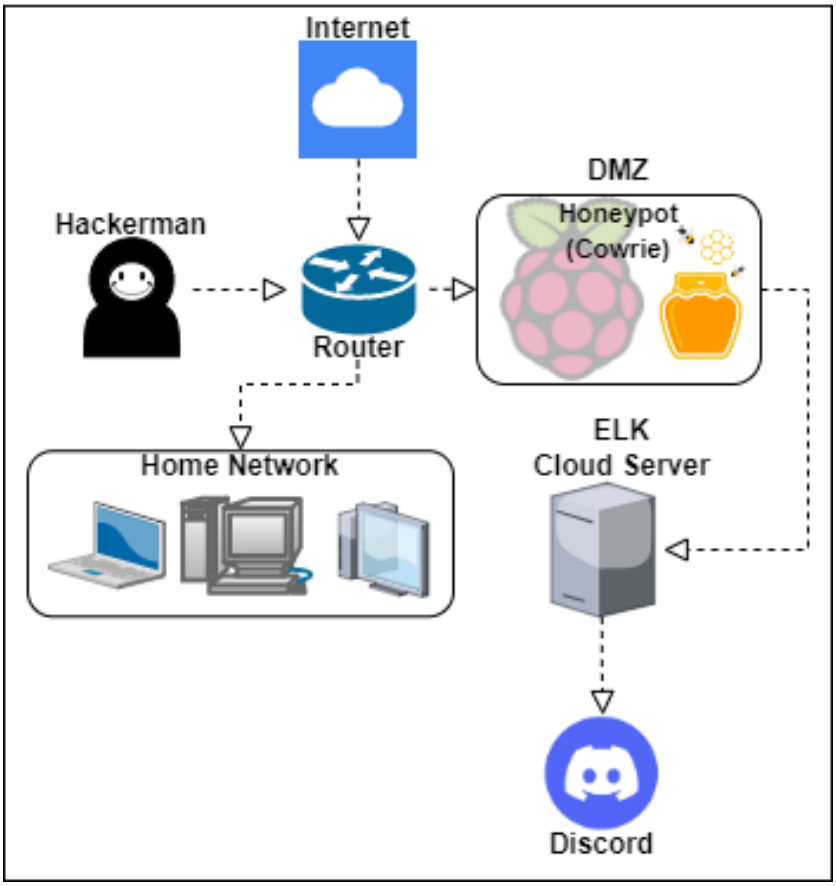

Figure 1: Honeypot Network Topology

5.2 Honeypot Design

The objective of this project is to develop and implement a Raspberry Pi honeypot and analyze the outcomes utilizing the ELK Stack. The proposed design focuses on three critical components: detection, defence, and reporting of attacks, as illustrated in Figure 2. This paper will primarily concentrate on these three sections, which will be discussed in detail in the following sections influenced by (Mohd Ali, Mohd Salleh, and Zulkipli 2021).

The first section is detection, which is crucial for mitigating SSH attacks as they are relatively simple to execute but difficult to identify. The detection system is used to identify any malicious activity on the system. The internal detection mechanism is demonstrated in Figure 2, with the detec-

tion phase concentrating on three main types of attacks: brute forces attack, SSH/TELNET port scans, and SSH and Telnet DDOS Attacks.

To detect SSH and Telnet attacks, such as brute-force, port scans, and DDOS attacks, the following process is utilized. After the user or attacker enters their username and password, the proposed design will verify their legitimacy before granting access to the system. If attackers attempt to log in for a large number of login attempts within a short frame of time, the system will send an alert to Discord containing information about the attacker. If the user is not legitimate, the system will determine the type of SSH attacks based on the attacker's behaviour, whether it is brute-force or port scans, or DDOS attacks.

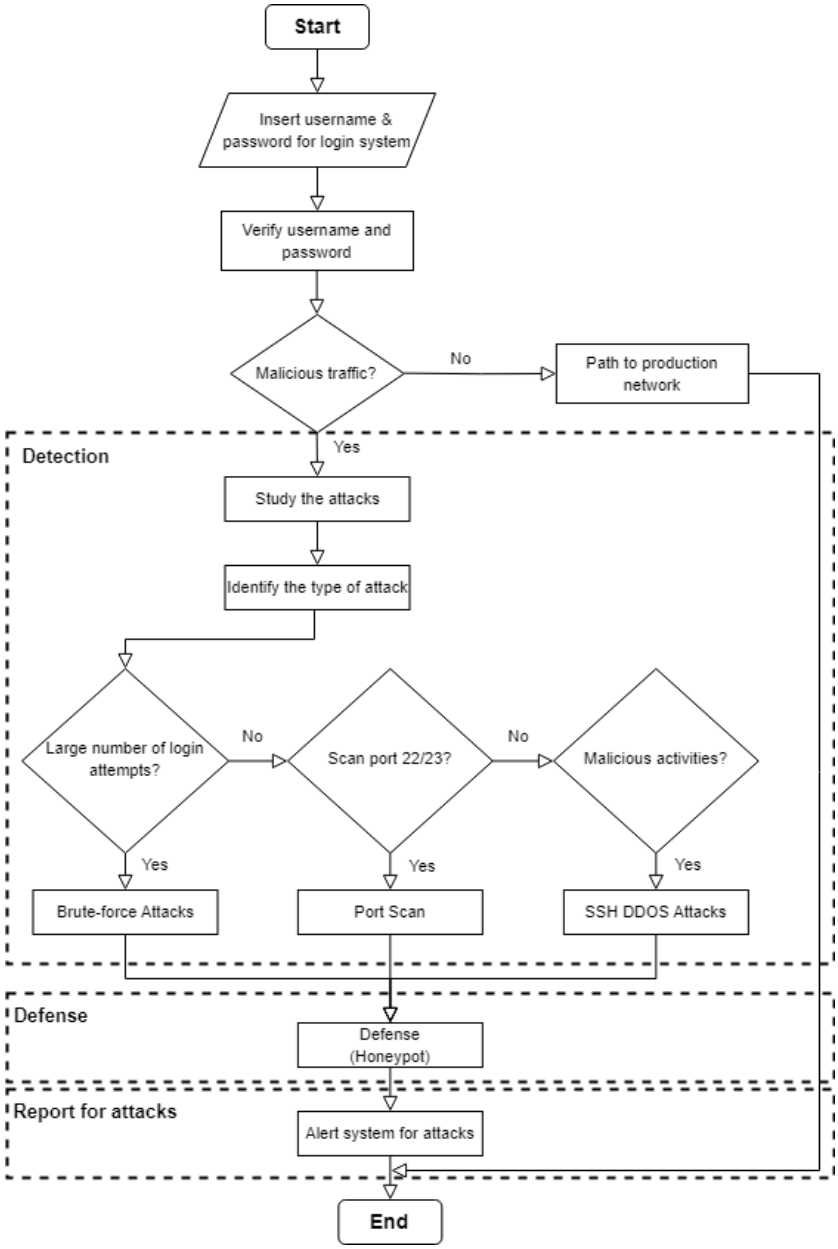

Figure 2: The Inner Design on Detection

The proposed design for the Raspberry Pi honeypot and data visualization aims to prevent SSH and Telnet attacks through the detection, defence, and reporting of malicious activity. The first part of the design focuses on detection, as this is crucial for mitigating SSH and Telnet attacks, which are both easy to execute and difficult to detect. The inner detection design concentrates on Brute force, port scans, and DDOS attacks. The system checks the username and password after the user or attacker inputs this information, to determine if the user is legitimate. If the attackers attempt to log in more than ten times within a minute, the system sends an alert via Discord and provides information about the attacker.

The second part of the design is defence, which protects against Bruteforce, Port Scans, and DDOS attacks before they can harm the network. The proposed defence design diverts the attacker to a honeypot system, where they can interact with fake systems. The Cowrie honeypot captures all terminal activity, such as Linux commands, as log files until the attacker logs out. If Cowrie honeypot detects any malicious activity, the system spoofs the attacker's IP address and moves them to the fake system. The honeypot also captures the attacker's interaction details, such as IP address, username, and password. By employing these measures, the design aims to provide a robust and effective defence against SSH and Telnet attacks. This could be seen in the flow diagram of Figure 3.

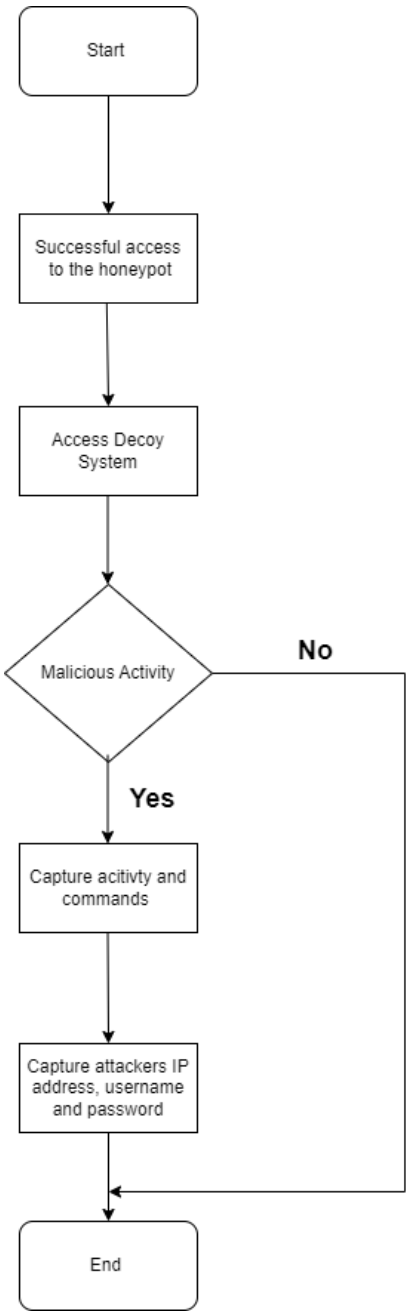

Figure 3: The Inner Design on Defense

The reporting phase is the final stage of the proposed design, and it is essential to record the detected SSH and Telnet attacks. This phase is depicted in Figure 4 and involves providing detailed information about the detected attacks. The reporting medium employed is Discord and ELK stack cloud server, which enables real-time alerts and notifications to be sent to the system and mobile devices. The report of attacks includes specific details such as the type of SSH and Telnet attack, the number of login attempts, the attacker's session, the attacker's source, and the user credentials used to log in. The type of SSH attacks detected are classified into three categories, namely brute-force attacks, port scans and DDOS attacks. The number of login attempts, whether a small or large number of failed login attempts, is recorded to provide insight into the attacker's persistence and intention. The attacker's session is also captured, providing a record of all keystrokes and commands used during the interaction with the honeypot. This would also be recorded in the ELK Stack server too. The attacker's source is another critical piece of information that is recorded, whether the IP address used is real or spoofed. Finally, the user credentials used to log in are recorded to determine the level of access the attacker had to the system. In summary, the reporting phase provides a comprehensive record of detected SSH and Telnet attacks, enabling system administrators to understand the nature and severity of attacks and take appropriate measures to prevent future occurrences. Discord's real-time notifications ensure that any detected attacks are immediately addressed, enhancing the system's security and reliability.

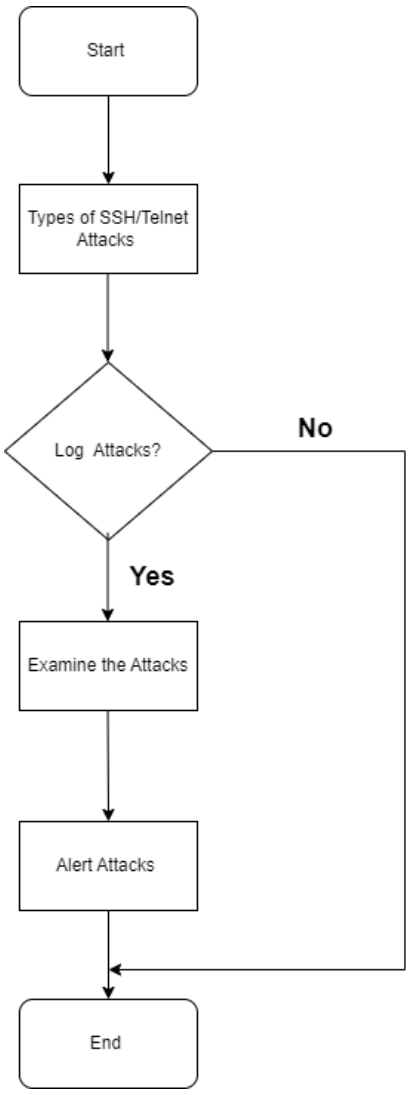

Figure 4: The Inner Design on Report

5.3 Classification Machine Learning Models

5.3.1 Pre-Processing

To ensure that the machine learning classification algorithms are accurate in detecting unauthorized access attempts targeting SSH and Telnet services on a home network, the dataset must participate in pre-processing and go

through various techniques. Pre-processing is essential in maximizing the performance of the machine learning models utilized in this project. This section will cover the pre-processing techniques used in creating machine learning models for the detection system. The pre-processing method would be closely represented by Figure 5

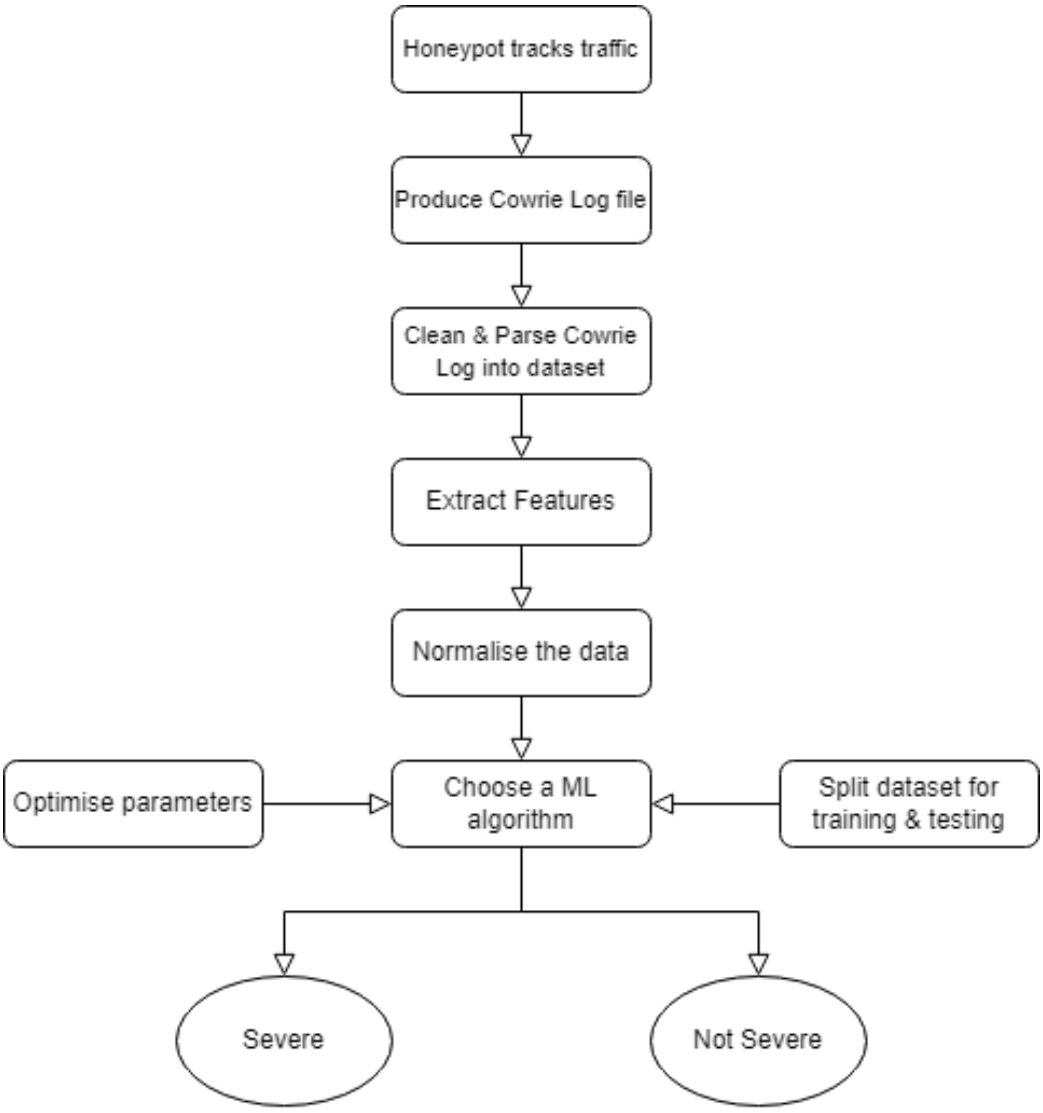

Figure 5: Honeypot Pre-processing Workflow

The first step in the pre-processing pipeline involves data collection. The

honeypot system collects data on unauthorized access attempts, including pertinent features related to SSH and Telnet services such as source IP, timestamps, and protocol-specific attributes. This raw data serves as the foundation for the machine learning models and must be carefully processed to ensure optimal performance.

Once the data has been collected, the cleaning process commences. This step involves inspecting the dataset for inconsistencies, errors, or missing values that could potentially compromise the accuracy and reliability of the machine learning models. Anomalies are corrected or removed, ensuring that the data fed into the models are of high quality and free from any irregularities.

Following data cleaning, feature selection is performed to identify the most relevant features for the machine learning classification algorithms. The chosen features should exhibit a strong ability to discriminate between normal and malicious activities. The selected features will form the basis for the model's decision-making process, thus significantly impacting the overall performance of the intrusion detection system.

In pre-processing, a dataset usually undergoes normalisation. However, due to the nature of the project's dataset, normalisation is not a requirement. As normalisation is normally used on numerical features to bring their values within a common scale. The dataset extracted from the log files, supplies features that are categorical such as `'source ip`, `input`, `timestamp` and `country`.

Next, data encoding is performed to convert categorical features, such as protocol types, into numerical values. This step is necessary to make the data compatible with machine learning algorithms that require numeric input. Various encoding techniques can be utilized, including One-Hot Encoding, Label Encoding, or Binary Encoding, depending on the characteristics of the categorical features and the preferences of the machine learning models.

Lastly, the pre-processed data is partitioned into training and testing sets. This partitioning facilitates model training and evaluation, ensuring that the performance of the models is assessed using unseen data. The partitioning ratio into 75% and 25% should be determined based on the dataset size and the desired balance between model complexity and generalizability.

For our classification targeted feature, followed the model by (Kristyanto et al. 2022), in order to identify potential malicious attacks within SSH and Telnet honeypot, it is classified into two levels, non-severe attacks and severe attacks. Severe attacks occur when the attacker successfully logs in and

subsequently attempts to execute Unix commands, which can be considered dangerous and threatening. Not-so-severe attacks, on the other hand, can be categorized into two types. The first type involves a successful connection, where the attacker manages to log in but does not execute any Unix commands. In this case, the attack is terminated after a successful brute force attempt. The distinction between this type of attack and a severe attack lies in the behaviour following a successful login. The second type of not-so-severe attack is characterized by a failed connection. In this scenario, the attacker is unable to log in, thus classifying it as less severe. This is shown by the following Table 1.

Severity Level	**Traffic Type**
Severe Attacks	SSH includes one or more Unix commands
Not-so-severe Attacks	SSH includes without Unix command SSH brute-force attacks

Table 1: Table of Severity Levels and Traffic Types

The Cowrie honeypot log files were collected and pre-processed to extract valuable information about unauthorized access attempts. The log files, originally in JSON raw data format, contained a wealth of information about each session, including timestamps, source IP addresses, authentication credentials, and executed commands. To facilitate further analysis and the development of machine learning models for intrusion detection, specific features were chosen from the pre-processed data. Table 2 presents the selected features and their respective details.

Features	Details
src_ip	Source IP
username_auth	username
password_auth	password
input_cmd	input command(s) from attackers
timestampt_auth	date and time of the session
severity	level of non-severe and severe attacks
country	Countries from Source IP addresses

Table 2: Features From Honeypot Pre-Processed Data

5.3.2 Model training and validation strategies

The model training and validation strategies for this project are crucial in ensuring the effectiveness of machine learning algorithms in detecting and classifying unauthorized access attempts for SSH and Telnet services. The primary goal is to select the best-performing model(s) with an optimal balance between accuracy, computational efficiency, interpretability, and suitability for the problem at hand. The following sections detail the model training and validation strategies for this project:

The pre-processed dataset is divided into training and testing sets, usually following a 75-25 or 80-20 ratio. This partitioning enables the use of the training set for model training while reserving the testing set for evaluating model performance on previously unseen data. This separation helps ensure that the evaluation results are unbiased, providing a reliable indication of the model's ability to generalize to new data.

To enhance the dependability of the evaluation process, k-fold cross-validation is implemented during the model training phase. With k-fold cross-validation, the training set is subdivided into k equally sized subsets (or folds). The model is trained k times, using k-1 folds for training and the remaining fold for validation in each iteration. Performance metrics are then averaged across the k iterations, yielding a more accurate and stable estimate of the models' performance.

Each model is trained using the training set. Appropriate hyperparameter tuning is applied to optimize the performance of each model. Techniques such as grid search or random search can be employed for hyperparameter tuning, as these methods systematically explore various combinations of hyperparameter values to identify the optimal configuration.

Upon completing the training of each model, their performance is assessed on the validation set (or the held-out fold in cross-validation) utilizing relevant performance metrics, including accuracy, precision, recall, and F1-score. These metrics offer insights into each model's effectiveness in detecting and classifying unauthorized access attempts and their capacity to minimize false-positive rates.

$$\text{Accuracy} = \frac{\text{True Positives (TP)} + \text{True Negatives (TN)}}{\text{Total Number of Samples}} \tag{1}$$

$$\tag{2}$$

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}} \tag{3}$$

$$\tag{4}$$

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}} \tag{5}$$

$$\tag{6}$$

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

Accuracy is the proportion of correct predictions (both true positives and true negatives) out of the total number of predictions made by the model. It is a widely used metric that provides a general overview of the model's performance. However, it may not be suitable in cases where the class distribution is imbalanced, as it could lead to misleading results. In such scenarios, precision, recall, and F1-score become more important.

Precision measures the proportion of true positives (correctly identified malicious activities) out of all the instances predicted as positive (both true positives and false positives). A high precision indicates that the model is good at avoiding false positives, which is essential in intrusion detection systems, as false alarms can lead to unnecessary investigations and increased operational costs.

Recall, also known as sensitivity or true positive rate, calculates the proportion of true positives out of all the actual positive instances (both true positives and false negatives). A high recall value indicates that the model is capable of detecting most of the actual malicious activities. In the context of intrusion detection, it is crucial to have a high recall to minimize the chances of undetected attacks that could cause severe damage to the system or network.

F1-score is the harmonic mean of precision and recall. It balances both precision and recall, providing a single metric that considers both false positives and false negatives. The F1-score is especially useful when there is an uneven class distribution, as it is less sensitive to class imbalance than accuracy. In intrusion detection systems, optimizing the F1-score helps ensure

that the model effectively detects malicious activities while minimizing false alarms.

The evaluation results are compared to select the best-performing model(s) for further refinement and integration into the research honeypot intrusion detection system. The selection process takes into account not only the accuracy of the models but also factors such as computational efficiency and suitability for addressing the specific problem.

6 Implementation

6.1 Honeypot Setup

The honeypot was set up and deployed for 14 days from April 7th to April 20th through DMZ. The Cowrie honeypot is installed on the Raspberry Pi machine including all the packages. In addition, set up the honeypot in a non-root user to prevent your machine if it is breached by the attacker. Make sure that the honeypot listens to SSH and Telnet ports which are port 22 and port 23. Lastly, once that is all set up change the legitimate SSH port connection outside of the honeypot. This means that the SSH connection port should not be the same as the honeypot ports to prevent potential backdoor access on the real machine.

Machine/Specs	Raspberry Pi 4	ELK Cloud Server (VM)	Attack Machine (VM)
Operating System	Ubuntu Server 22.04.2LTS (ARM 64)	Ubuntu Server 22.04.2 LTS (x86-64)	Kali GNU/Linux 2021.1 (x86-64)
Hardware Processor	64-bit BCM2711 quad-core Cortex-A72 1.5GHz	"None"	Intel Core i7-10510U 5Ghz (8 core)
VM Processor	"None"	4 core	2 core
RAM	2 GB	8 GB	16 GB
Storage	128 GB	160 GB	50 GB
Role	IDS honeypot	ELK visualization server	Attack machine

Table 3: System Specifications

6.1.1 Configuration of the Cowrie Honeypot and ELK Stack Server

Table 4 displays the change of the username and password configuration from the default database file into the configured one. According to (Cabral et al. 2021), it allows more deceptiveness and is close to a realistic sever for the honeypot. However, this is not enough to completely mask the server as the research also changes and randomises almost the entire system through a specific Python script. However, due to outdated dependencies and packages, this Python script file implementation was unsuccessful.

Table 4: Default and Configured Honeypot Usernames and Passwords

Default Username	Default Password	Configured Username	Configured Password
root	!root	root	!root
root	!12345	root	!12345
root	(any passwords)	ubuntu	somepassword
tomcat	(any passwords)	admin	admin
oracle	(any passwords)	admin	123456
(any username)	(any passwords)	ubuntu	ubuntu

Additionally, Filebeat and Packetbeat are implemented and installed on the Raspberry Pi system. They are utilized to forward and centralize log data into the ELK stack cloud server by sending it to Elasticsearch. In the configurations for Filebeat and Packetbeat, they are set to port 9200 and the specified IP address of the ELK server, enabling the transmission of logs from the Cowrie Honeypot for data visualization.

Regarding the ELK Stack server, it is deployed on a Linode cloud-hosting virtual machine running an Ubuntu server. This is similar to (Mohd Ali, Mohd Salleh, and Zulkipli 2021). The machine requires the installation of Java, Nginx, Elasticsearch, Logstash, and Kibana. Elasticsearch, an open-source distributed search and analytics engine, facilitates real-time querying, storage, and analysis of extensive data sets. In this research, Elasticsearch is accessed through port 9200 and has a network host address. Kibana serves as the visualization component of the server, organizing and storing data in the Elasticsearch index. With the Nginx proxy server, Kibana can also be accessed remotely from a web server. Kibana is configured for port 5601 access. Depending on the ELK stack's usage, port forwarding and IP table commands may be employed for accessing specific services and data through designated ports.

As discussed in the Design section, ElastAlert is integrated into the ELK stack server to send alerts to Discord when attacks occur. This is achieved by using a webhook from the Discord bot link. Each ElastAlert rule for different SSH/Telnet attacks varies based on the number of events and timeframes from Packetbeat and Filebeat. The brute force rule is determined by the number of events within a one-minute timeframe. If there are more than ten login attempts, ElastAlert will be triggered and send a notification to Discord for reporting. This process involves collecting data from Filebeat,

which filters the Cowrie log files for failed logins. For the port scanning detection rule, Packetbeat detects the number of events based on traffic and connection events. If more than ten requests are sent within 30 seconds, the system will send an alert. Lastly, for a DDoS attack, the rule is similar to port scanning but requires 100 requests within a one-minute duration.

6.2 Machine learning models Implementation

6.2.1 Proposed Models

For this section, this project uses a combination of classification machine learning algorithms such as Support vector machine(SVM), K-Nearest Neighbor(KNN) and Random Forest (RF). Based on the analysis section, these algorithms have shown through different research that are able to perform with the most consistent performance metrics. This is shown by (Kristyanto et al. 2022) and (Jiang and Zheng 2020) where the accuracy score has gained 0.90 respectively and attack classification. This was tested and developed in Juptyer Notebook by Anaconda Environment with Python3.8. In addition, for the data cleaning, a Python notebook is created called: "PrePrecossing_Final.ipynb" to extract all the JSON log files into a CSV file with the features mentioned in Table 2. The Attack Machine is used to train the classification models. However, there is no correlation between the attack tests as it is used for convenience. Python libraries used Panda, Numpy, Scikit-learn, Seaborn and Matlplotlib and GeoIP2 and time. The dataset contains 61674 records to develop each model. The dataset collection is based on the paper (Kristyanto et al. 2022) and (Dowling, Schukat, and Barrett 2020).

6.2.2 Fine-tuning models for optimal performance

The development of an effective intrusion detection system for home networks using a Raspberry Pi-based honeypot required the selection and implementation of suitable machine learning models. Three classification algorithms, namely K-Nearest Neighbors (KNN), Random Forest (RF), and Support Vector Machine (SVM), were chosen for comparison due to their widespread use in machine learning applications and their potential for accurate intrusion detection.

To ensure the integrity of the model evaluation, a rigorous training and validation strategy was employed. The dataset, which comprised various

features such as input commands, country of origin, username and password used in the attack, and attack severity, was first preprocessed to handle missing values and standardize the data. The dataset was then divided into training and testing sets using a 75-25 split, with 75% of the data used for training and the remaining 25% reserved for testing. This approach allowed for an unbiased assessment of each model's performance on unseen data. A preprocessor is defined using the ColumnTransformer class to handle categorical features with different encoding techniques. The 'input_cmd', 'username_auth', and 'password_auth' columns are encoded using the TargetEncoder, while the 'country' column is encoded using the OneHotEncoder.

To further enhance the reliability of the model evaluation, k-fold cross-validation was utilized. This technique involves partitioning the training data into k-equal subsets, training the model on k-1 subsets, and evaluating its performance on the remaining subset. This process was repeated k times, each time using a different subset for validation. The average performance across all k iterations was used as the final performance metric. The use of k-fold cross-validation mitigated the risk of overfitting and provided a more robust evaluation of the model's performance. In this project, a 5-fold cross-validation was implemented.

To fine-tune the hyperparameters of each machine learning model, a randomized search, RandomizedSearchCV, was conducted using a pre-defined search space for each model's hyperparameters. This approach facilitated the exploration of the hyperparameter space efficiently and increased the likelihood of finding optimal values. The performance of each model was assessed using various metrics, including accuracy, recall, precision, and F1 score. However, for K-Nearest Neighbors, it is implemented by using GridSearchCV as more reliable results by searching through all possible combinations due to the algorithm's simple and intuitive structure. A scikit-learn Pipeline is created with the preprocessor for each model

During the training phase, each model was trained using the preprocessed training data, and their hyperparameters were optimized to maximize their performance. Once the optimal hyperparameters were identified, the models were retrained using these values to obtain the best possible performance on the training data.

After training and hyperparameter optimization, the models were validated on the testing data. This process involved using the trained models to predict the attack severity based on the features present in the testing dataset. The models' performance was then compared using the chosen eval-

uation metrics. ROC-graph and confusion matrix was also formed through the Python libraries as well as the training duration.

7 Testing and Integration

7.1 Testing Honeypot Attacks

The test cases of SSH and Telnet attacks are done through the Attack Machine based on Table 3

- **Port Scanning Attack:** This attack employs the Nmap tool. The test scenario involves scanning all hosts on the target subnet and conducting additional scans on the most vulnerable hosts.
  - `sudo nmap -A <ip_address>`
  - `sudo nmap -sV -sT -sU <ip_address>`
  - `sudo nmap -Pn -p 22 -sU -sT <ip_address>`
  - `sudo nmap -Pn -p 23 <ip_address>`
- **TCP SYN DOS Attack:** This attack sends a large number of SYN packets to the target Telnet server in an attempt to overwhelm its resources.
  - `hping3 --count 1500000 --data 120 --syn -p 23 --flood --rand-source (ip address of the server)`
- **TCP SYN DOS Attack:** This attack sends a large number of SYN packets to the target SSH server in an attempt to overwhelm its resources.
  - `hping3 --count 1500000 --data 120 --syn -p 22 --flood --rand-source (ip address of the server)`
- **Hydra Brute Force Attack:** This attack attempts to gain unauthorized access to the target server by trying different username and password combinations.
  - `hydra -L username.txt -P passsword.txt ssh://<ip> -s 22 -V -I -F`
  - `hydra -L username.txt -P password.txt telnet://<ip> -s 23 -V -I -F`

- **username.txt contents:**
  - root, admin, user, test, ubuntu, ubnt, support, oracle, guest, supervisor, ftp, pi
- **password.txt contents:** Contains a built-in password combination from Kali Linux based on fasttrack.txt, which includes the most common passwords used for brute force attacks.

These attacks were implemented based on (Hariawan and Sunaringtyas 2021), and (Yadav 2020) for the most common attacks based on OWASP Top 10.

7.2 Evaluating System performance using predefined metrics

The dataset has already been pre-processed and the machine learning algorithm hyperparameters are ready to be set. This section will cover how the optimal hyperparameters, needed for the machine learning models, will be selected. Hyperparameters are parameters that cannot be learned from the training data and control numerous areas of the machine-learning algorithm. They can affect the way the model will learn from the training data. Hyperparameters are very important when it comes to the performance of a model, so finding the optimal values for the hyperparameters is a crucial decision to creating an accurate and efficient machine learning model. Bad hyperparameters would lead to a substandard performing machine learning model. This can lead to low accuracy or high error rates. Attempting to manually find an optimal hyperparameter in each machine-learning model will be a long arduous journey. However, this problem can be easily solved by implementing grid search or randomised search.
Grid Search (Deo et al. 2021) and Randomized Search are techniques that are used for hyperparameter tuning. It finds the optimal hyperparameter by constantly trying to identify the optimum possible combination of hyperparameters on a specified range. It will select the combination that performs best on the validation set. Both techniques also help to avoid overfitting from happening. Overfitting occurs when a model performs well on the training data yet performs poorly on new, unseen data. Using this would allow the machine learning models to be equipped with better accuracy. The tables

below will reveal the hyperparameter settings prepared, for each algorithm, to be tuned and selected.

RF - Hyperparameters	**Settings**
Number of Estimators (n_estimators)	50, 100, 200
Maximum Depth (max_depth)	None, 10, 20, 30
Minimum Samples Split (min_samples_split)	2, 5, 10
Minimum Samples Leaf (min_samples_leaf)	1, 2, 4
Bootstrap (bootstrap)	True, False

Table 5: Random Forest's Hyperparameter Experiment Settings For RandomizedSearch

KNN - Hyperparameters	**Settings**
Number of Neighbors (n_neighbors)	1, 2, ..., 30
Weights (weights)	uniform, distance
Metric (metric)	euclidean, manhattan, minkowski

Table 6: K-Nearest Neighbors' Hyperparameter Experiment Settings For Grid Search

SVM - Hyperparameters	**Settings**
Kernel (kernel)	linear, rbf, poly, sigmoid
C (C)	10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100, 1000
Gamma (gamma)	scale, auto, 0.1, 1, 10

Table 7: Support Vector Machine's Hyperparameter Experiment Settings For Randomised

8 Results

8.1 Performance of the Honeypot

A total of 37,768 recorded login attempts targeting the Raspberry Pi honeypot within a two-weeks time frame of deployment. Out of these attempts, 19,163 were successful in gaining unauthorized access, while 18,605 failed to do so. With 74 countries around the world, trying to breach the honeypot. This highlights the prevalence of unauthorized access attempts and the need for robust intrusion detection systems to protect vulnerable networks.

Furthermore, the login attempts were observed to target two different protocols: Telnet and SSH. The distribution of these attempts reveals that 54.74% of them were aimed at exploiting the Telnet protocol, while the remaining 45.26% focused on the SSH protocol.

Table 5 shows the most common username and password used in attempts to gain access to the Raspberry Pi honeypot.

Table 8: Top Usernames and Passwords

Rank	**Username**	**Count**	**Password**	**Count**
1	root	29,201	admin	578
2	aaliyah	3,013	1234	395
3	admin	1,802	password	380
4	guest	236	123456	372
5	user	183	root	253
6	supervisor	139	12345	232
7	ubnt	105	user	160
8	pi	102	123456789	150
9	support	99	7ujMko0admin	143

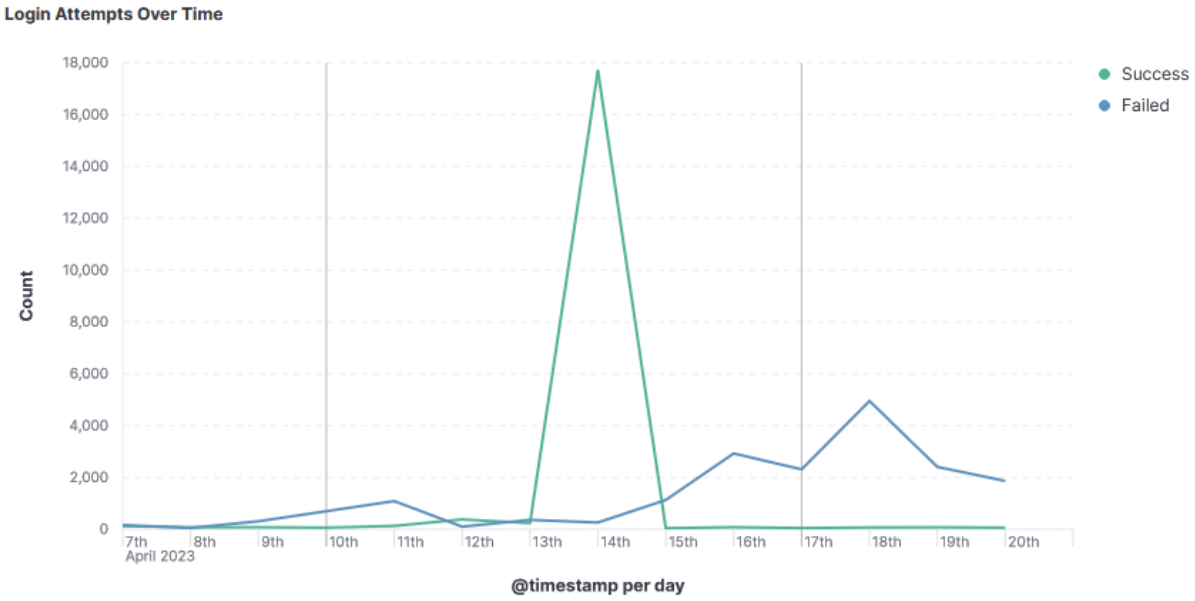

Figure 6: Kibana Login Attempt graph Per Day

Figure 6 shows the Kibana environment from the ELK Stack cloud server, which visualises the number of login attempts over the two-week period the Raspberry Pi honeypot was functional. On day 14/04/2023, Figure 6 presents the day with the highest number of login attempts/interactions, which was almost 18,000 attempts/interactions. The graph shows data between the successful and failed login attempts over time.

The table below, Table 9, presents the four top countries with the most login attempts/interactions. France leads with the rank position one, with a number of 17,567 login attempts/interactions. And China follows behind in position two, with 11,932 login attempts/interactions

Table 9: Top Countries

Rank	Country	Count
1	France	17567
2	China	11932
3	Pakistan	3047
4	Russia	766

Table 10: Top 5 Most Common Commands

Rank	Command	Count
1	`echo -e "\x6F\x6B"`	17559
2	sh	2654
3	shell	2654
4	system	2652
5	rm .s; exit	2564

In Table 10, the most executed commands when the attackers gained access to the decoy system in the honeypot are displayed. The command `"echo -e "\x6F\x6B""` was rank one. It prints the characters `"ok"` on the terminal by enabling the interpretation of escape sequences with `-e` option and uses the hexadecimal values, `"\x6F\x6B`", for the ASCII characters `'o'` and `'k`. The fifth-ranked command `"rm .s; exit"`, is a command to remove files with `.s` in them (which are assembly files) and exit the shell session.

Table 11 demonstrates the honeypot system is able to detect SSH and Telnet attacks. The Discord alert system is able to detect Port scanning from Nmap, and DDOS Attacks from both ports 22 and 23. However, there are delay responses in the Brute Force detection which is also included in both ports. During the Nmap port scanning test, the system experienced a CPU usage of 2.30% and memory utilization of 24.50%, while maintaining a temperature of 58.4°C. As for the Brute Force test, the system experienced a CPU usage of 3.30% and memory utilization of 23.50%, while maintaining a temperature of 58°C for Port 22. Brute Force test for Port 23, the system experienced a CPU usage of 4.7% and memory utilization of 24.1%, while maintaining a temperature of 58°C. For the DDOS attack on port 22, the system experienced a CPU usage of 45.60% and memory utilization of 27.70%, while maintaining a temperature of 60°C for Port 22. Lastly, for the DDOS attack on port 23, the system experienced a CPU usage of 40.60% and memory utilization of 24.50%, while maintaining a temperature of 58°C.

The output of Figure 7 includes the timestamp, attacker IP, the destination IP, and the number of requests sent within the time duration. The output of Figure 8 also includes the timestamp, attacker IP, the destination IP, and the number of requests sent within the time duration. Finally, the output of Figure 9 includes the timestamp, attacker IP, the destination IP,

username, password, country and city of the attacker.

Raspberry Pi Attack Cases	Attack Detection	CPU	Memory	Temperature(°C)
Port Scanning Nmap	Yes	2.30%	24.50%	58.4
Hydra Brute Force (Port 22)	Delay	3.3%	23.3%	58
Hydra Brute Force (Port 23)	Delay	4.7%	24.1%	58
TCP SYN DDOS Attack (Port 22)	Yes	45.60%	27.70%	60
TCP SYN DDOS Attack (Port 23)	Yes	40.60%	24.50%	60

Table 11: Raspberry Pi Test Cases & Performance Evaluation Results

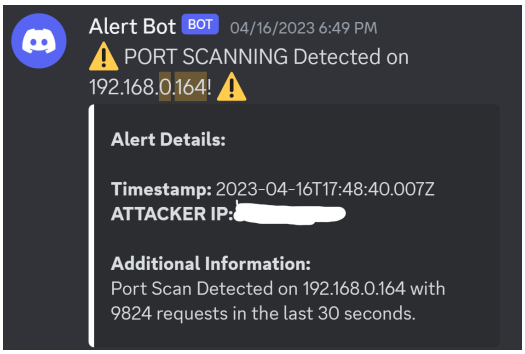

Figure 7: Discord Port Scan Notification Alert

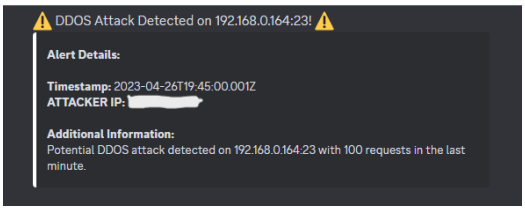

Figure 8: Discord DDOS Attack Notification Alert

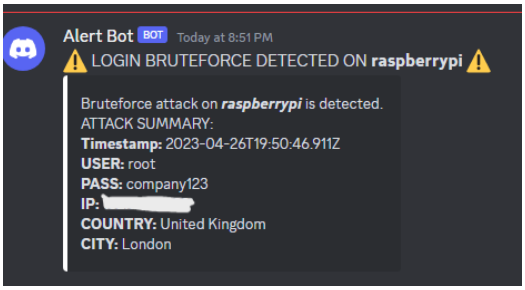

Figure 9: Discord Brute-Force Attack Notification Alert

8.2 Comparison of Machine Learning models

This sections covers the results produced by the chosen machine learning models, displaying the efficiency, accuracy and optimal hyperparameter values of each model.

Model	Hyperparameter	Best Values
SVM	Kernel (kernel)	rbf
	Gamma (gamma)	1
	C (C)	10.0
RF	n_estimators (n_estimators)	50
	min_samples_split (min_samples_split)	10
	min_samples_leaf (min_samples_leaf)	1
	max_depth (max_depth)	None
	bootstrap (bootstrap)	False
KNN	Metric (metric)	manhattan
	n_neighbors (n_neighbors)	4
	Weights (weights)	uniform

Table 12: Best Hyperparameters

Table 13 presents the classification performance metric results for three different machine learning models: K-Nearest Neighbors (KNN), Random Forest (RF), and Support Vector Machine (SVM). The performance of each model is assessed using various metrics, such as accuracy, recall, precision, F1 score, and training duration.

The KNN model achieves an accuracy of 0.9447, a recall of 0.8838, a precision of 0.9678, and an F1 score of 0.9239. The training duration for

this model is relatively short, at 23 minutes. The RF model exhibits an accuracy of 0.9422, a recall of 0.8839, a precision of 0.9606, and an F1 score of 0.9207. The training duration for the RF model is even shorter, at only 3 minutes. Lastly, the SVM model demonstrates an accuracy of 0.9459, a recall of 0.8791, a precision of 0.9760, and an F1 score of 0.9251. However, the training duration for the SVM model is significantly longer, at 2 hours and 20 minutes.

These results indicate that all three models perform well in terms of classification metrics. However, the training duration of each model varies considerably, which may impact the choice of the model based on the specific application and computational resources available. As mentioned before in Implementation Section, this was trained using the Attack Machine without the Kali Linux VM.

Table 13: Classification Performance Metric Results

Model	**Accuracy**	**Recall**	**Precision**	**F1 Score**	**Training Duration**
KNN	0.9447	0.8838	0.9678	0.9239	23 mins
RF	0.9422	0.8839	0.9606	0.9207	3 mins
SVM	0.9459	0.8791	0.9760	0.9251	2h20 mins

The confusion matrix presented in Figure10 malicious and non-malicious attacks can be interpreted in the following manner. There were 9395 cases correctly classified as non-malicious attacks based on severity, which are referred to as True Negatives (TN). In contrast, there were 172 cases that were incorrectly classified as malicious attacks even though they were non-malicious; these cases are known as False Positives (FP).

Furthermore, there were 680 cases incorrectly classified as non-malicious attacks when they were, in fact, malicious attacks. These cases are called False Negatives (FN). Lastly, there were 5172 cases that were correctly classified as malicious attacks, labelled as True Positives (TP).

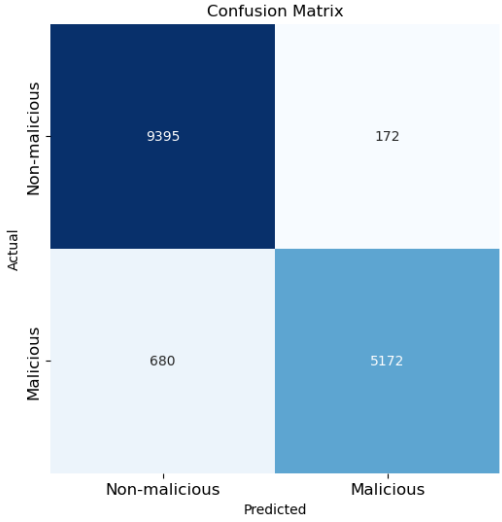

Figure 10: Confusion matrix for KNN

The confusion matrix presented in Figure11 malicious and non-malicious attacks can be interpreted in the following manner. There were 9441 cases correctly classified as non-malicious attacks based on severity, which are referred to as True Negatives (TN). In contrast, there were 126 cases that were incorrectly classified as malicious attacks even though they were non-malicious; these cases are known as False Positives (FP).

Furthermore, there were 707 cases incorrectly classified as non-malicious attacks when they were, in fact, malicious attacks. These cases are called False Negatives (FN). Lastly, there were 5145 cases that were correctly classified as malicious attacks, labelled as True Positives (TP).

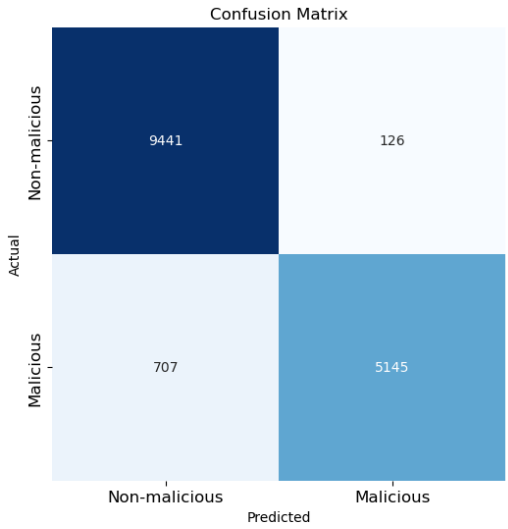

Figure 11: Confusion matrix for SVM

The confusion matrix presented in Figure12 analyses malicious and non-malicious attacks that can be interpreted in the following manner. There were 9355 cases correctly classified as non-malicious attacks based on severity, which are referred to as True Negatives (TN). In contrast, there were 212 cases that were incorrectly classified as malicious attacks even though they were non-malicious; these cases are known as False Positives (FP).

Furthermore, there were 679 cases incorrectly classified as non-malicious attacks when they were, in fact, malicious attacks. These cases are called False Negatives (FN). Lastly, there were 5173 cases that were correctly classified as malicious attacks, labelled as True Positives (TP)

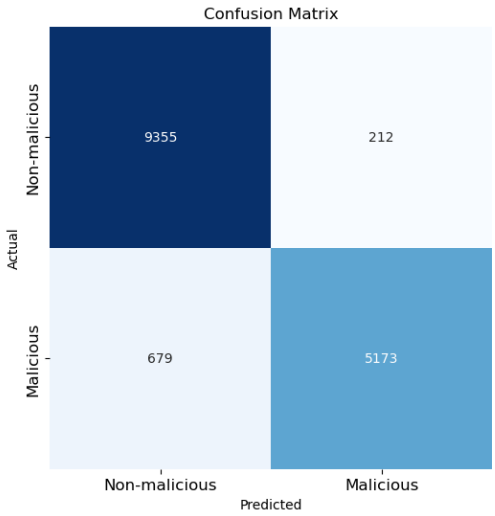

Figure 12: Confusion matrix for RF

In Figure 13, the ROC graphs for all models are presented along with their respective AUC values. The graph illustrates the performance of each model in terms of its ability to correctly classify malicious and non-malicious attacks. The ROC curves for the various models are plotted by displaying the True Positive Rate (TPR) on the y-axis against the False Positive Rate (FPR) on the x-axis. The ROC curve with an area of 0.93 demonstrates a strong ability to differentiate between the two classes, as a larger AUC value indicates better overall performance. A threshold of 0.5 is used as the decision boundary for classifying the attacks. If the probability of an attack being malicious is greater than or equal to the threshold, it is classified as malicious; otherwise, it is classified as non-malicious. By adjusting the threshold value, we can modify the trade-off between TPR and FPR to meet the specific requirements of the application.

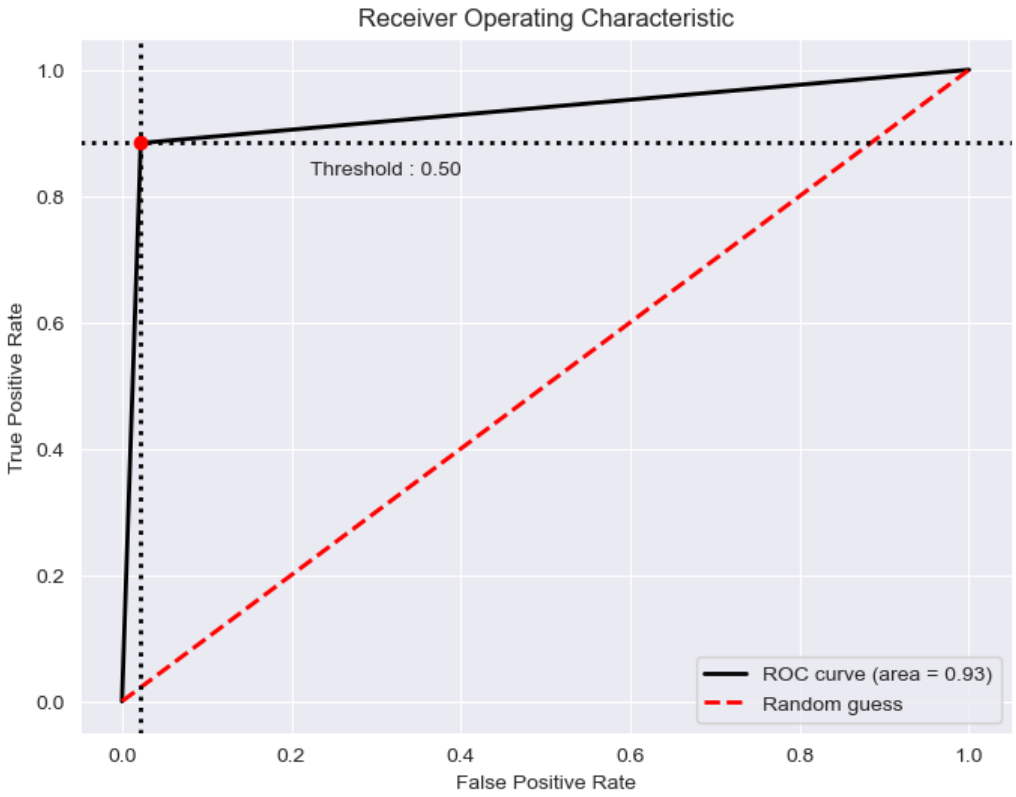

Figure 13: ROC - Graph for All Models

9 Discussion and Evaluation

The final product of the project met the base functional level of the requirements within the time frame of the development. The core features have been implemented successfully. However, there were still inconsistent delays time regarding the brute force notification from the discord alert system. This could be due to issues within the configuration within Filebeat when trying to identify the specific elements from the Cowrie JSON file regarding the number of failed login attempts in a given time duration. The data displays fine in Kibana when refresh but this could be a novice error by the implementation part. Regardless, the goal was to improve the work of (Jeremiah 2019) which was achieved successfully and includes the goal of real-time data visualization, data collection and extraction which is evaluated into machine learning models to classify SSH and Telnet attacks. Whereas Traditional honeypots cause a higher rate of false positives. The models developed are not implemented within real-time which could be used for future works with other open source IDS, but due to the limited resources using ElastAlert is one of the options which is influenced by (Hermawan, Novianto, and Octavianto 2021). Another limitation of this project is as it only implements one type of honeypot which limits the varieties of protocols and diversities of attacks to create a more coherent research honeypot.

This project was spent on creating features and training and experimenting with the models. There were issues implementing a home network system as external factors such as ISP and electrical shortages are also issues that needed to be considered as well as maintaining good security practices such as strong passwords for the honeypot server and updates.

The classification models could have been improved using more features from the raw data. However, there were issues parsing more useful data due to the different layers of JSON. These features could be protocols, OS versions, destination IP, and destination port. Moreover, other comparison models such as (Ramakrishnan, Gokul, and Nigam 2021) and (Saputro, Purwanto, and Ruriawan 2021), do not have a comprehensive data visualization and alert system.

10 Conclusion

The goal of this project was to create an intrusion detection system on a Raspberry Pi honeypot and identify malicious attacks using machine learning classification. Most of the goals made in this project have been achieved, making the project almost a success. Unfortunately, the alert system creates nuances when reporting brute attacks on Discord.

In conclusion, this project has demonstrated the effectiveness of a Raspberry Pi-based honeypot in detecting and analyzing various cyber-attacks targeting home networks. The implementation of machine learning algorithms, such as Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Random Forest (RF), has shown promising results in terms of classification accuracy, precision, recall, and F1-score. These findings contribute to the growing body of knowledge on honeypot intrusion detection systems and highlight their potential for enhancing the security of home networks.

By understanding the techniques and origins of these attacks, users can better protect their home networks, and developers can design more effective intrusion detection systems tailored to the unique needs of residential environments. A lightweight Raspberry Pi SSH honeypot platform and ELK stack data visualization cloud platform.

Future work on this project could explore several areas. One possibility is the integration of additional machine learning algorithms to investigate the performance of other classification algorithms, such as deep learning models, and compare their effectiveness with the current approaches. Another area of focus could be the expansion of the feature set to improve the classification performance and enable the detection of a wider range of attack types.

Moreover, implementing real-time intrusion detection and response mechanisms could further enhance the security of home networks. Testing the proposed system using larger and more diverse datasets would validate its effectiveness in different network environments. Finally, designing and implementing user-friendly interfaces could facilitate the management and monitoring of the honeypot system, making it more accessible to non-expert users.

References

Ariffin, Tajul Azhar Mohd Tajul et al. (2022). "IoT attacks and mitigation plan: A preliminary study with Machine Learning Algorithms". In: *2022 International Conference on Business Analytics for Technology and Security (ICBATS)*. IEEE, pp. 1–6.

Başer, Melike, Ebu Yusuf Güven, and Muhammed Ali Aydın (2021). "SSH and Telnet Protocols Attack Analysis Using Honeypot Technique: Analysis of SSH AND TELNET Honeypot". In: *2021 6th International Conference on Computer Science and Engineering (UBMK)*. IEEE, pp. 806–811.

Bontchev, Vesselin and Veneta Yosifova (2019). "Analysis of the global attack landscape using data from a telnet honeypot". In: *Information & Security: An International Journal* 43.2.

Cabral, Warren Z et al. (2021). "Advanced cowrie configuration to increase honeypot deceptiveness". In: *ICT Systems Security and Privacy Protection: 36th IFIP TC 11 International Conference, SEC 2021, Oslo, Norway, June 22–24, 2021, Proceedings*. Springer, pp. 317–331.

Deo, Tejas Y et al. (2021). "A white-box SVM framework and its swarm-based optimization for supervision of toothed milling cutter through characterization of spindle vibrations". In: *arXiv preprint arXiv:2112.08421*.

Dowling, Seamus, Michael Schukat, and Enda Barrett (2020). "New framework for adaptive and agile honeypots". In: *Etri Journal* 42.6, pp. 965–975.

Fadhilah, Dede and Marza Ihsan Marzuki (2020). "Performance analysis of ids snort and ids suricata with many-core processor in virtual machines against dos/ddos attacks". In: *2020 2nd International Conference on Broadband Communications, Wireless Sensors and Powering (BCWSP)*. IEEE, pp. 157–162.

Gassais, Robin et al. (2020). "Multi-level host-based intrusion detection system for Internet of things". In: *Journal of Cloud Computing* 9, pp. 1–16.

Hariawan, Febrian Rachmad and Septia Ulfa Sunaringtyas (2021). "Design an Intrusion Detection System, Multiple Honeypot and Packet Analyzer Using Raspberry Pi 4 for Home Network". In: *2021 17th International Conference on Quality in Research (QIR): International Symposium on Electrical and Computer Engineering*. IEEE, pp. 43–48.

Haseeb, Junaid et al. (2020). "IoT attacks: features identification and clustering". In: *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, pp. 353–360.

Hermawan, Denny, Nugroho Ganda Novianto, and Digit Octavianto (2021). "Development of Open Source-based Threat Hunting Platform". In: *2021 2nd International Conference on Artificial Intelligence and Data Sciences (AiDAS)*. IEEE, pp. 1–6.

Hinov, Nikolay and Filip Krastev (2022). "Identification, Vulnerability Research and Cybersecurity of Raspberry Pi Devices". In: *2022 8th International Conference on Energy Efficiency and Agricultural Engineering (EE&AE)*. IEEE, pp. 1–6.

Huang, Cheng et al. (2019). "Automatic identification of honeypot server using machine learning techniques". In: *Security and Communication Networks* 2019, pp. 1–8.

Jeremiah, June (2019). "Intrusion detection system to enhance network security using raspberry pi honeypot in kali linux". In: *2019 International Conference on Cybersecurity (ICoCSec)*. IEEE, pp. 91–95.

Jiang, Kui and Haocheng Zheng (2020). "Design and implementation of a machine learning enhanced web honeypot system". In: *2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. IEEE, pp. 957–961.

Joyon, Ahmed Ruhul Quddos (2022). "Author's declaration of originality". In.

Khan, Zeeshan Ali and Ubaid Abbasi (2020). "Reputation management using honeypots for intrusion detection in the internet of things". In: *Electronics* 9.3, p. 415.

Khraisat, Ansam and Ammar Alazab (2021). "A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges". In: *Cybersecurity* 4, pp. 1–27.

Kristyanto, Marco Ariano et al. (2022). "SSH Bruteforce Attack Classification using Machine Learning". In: *2022 10th International Conference on Information and Communication Technology (ICoICT)*. IEEE, pp. 116–119.

Lee, Seungjin et al. (2021). "Classification of botnet attacks in IoT smart factory using honeypot combined with machine learning". In: *PeerJ Computer Science* 7, e350.

Li, Taotao, Zhen Hong, and Li Yu (2020). "Machine learning-based intrusion detection for iot devices in smart home". In: *2020 IEEE 16th International Conference on Control & Automation (ICCA)*. IEEE, pp. 277–282.

Martin, Erik David, Joakim Kargaard, and Iain Sutherland (2019). "Raspberry Pi Malware: An analysis of cyberattacks towards IoT devices". In: *2019 10th International Conference on Dependable Systems, Services and Technologies (DESSERT)*. IEEE, pp. 161–166.

Mohd Ali, Fakariah Hani, Muhammad Fadhli Mohd Salleh, and Nurul Huda Nik Zulkipli (2021). "An experimental study on cloud honeypot and data visualization using ELK stack". In: *International Journal of Nonlinear Analysis and Applications* 12.Special Issue, pp. 1117–1132.

Morozov, Dmytro S et al. (2023). "Honeypot and cyber deception as a tool for detecting cyber attacks on critical infrastructure". In.

Neranjan Thilakrathne, Navod, Rohan Samarasinghe, and Madhuka Priyashan (2021). "Evaluation of Cyber Attacks Targeting Internet Facing IoT: An Experimental Evaluation". In: *arXiv e-prints*, arXiv–2201.

Palša, Jakub et al. (2022). "Configuration Honeypots with an Emphasis on Logging of the Attacks and Redundancy". In: *2022 IEEE 20th Jubilee World Symposium on Applied Machine Intelligence and Informatics (SAMI)*. IEEE, pp. 000073–000076.

Pashaei, Abbasgholi et al. (2022). "Early Intrusion Detection System using honeypot for industrial control networks". In: *Results in Engineering* 16, p. 100576.

Ramakrishnan, Karthik, P Gokul, and Rohan Nigam (2021). "Pandora: An IOT based Intrusion Detection Honeypot with Real-time Monitoring". In: *2021 International Conference on Forensics, Analytics, Big Data, Security (FABS)*. Vol. 1. IEEE, pp. 1–7.

Saputro, Elang Dwi, Yudha Purwanto, and Muhammad Faris Ruriawan (2021). "Medium interaction honeypot infrastructure on the internet of things". In: *2020 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS)*. IEEE, pp. 98–102.

Sethi, Tanmay and Rejo Mathew (2021). "A Study on Advancement in Honeypot based Network Security Model". In: *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*. IEEE, pp. 94–97.

Sumanth, R and KN Bhanu (2020). "Raspberry Pi based intrusion detection system using k-means clustering algorithm". In: *2020 Second In-*

ternational Conference on Inventive Research in Computing Applications (ICIRCA). IEEE, pp. 221–229.

TAŞÇI, Hatice et al. (2021). "Password Attack Analysis Over Honeypot Using Machine Learning Password Attack Analysis". In: *Turkish Journal of Mathematics and Computer Science* 13.2, pp. 388–402.

Tripathi, Shyava and Rishi Kumar (2018). "Raspberry pi as an intrusion detection system, a honeypot and a packet analyzer". In: *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*. IEEE, pp. 80–85.

Yadav, Vijay (2020). "A Study of Threats, Detection and Prevention in Cybersecurity". In: *International Research Journal of Engineering and Technology (IRJET), May*, pp. 1150–1153.

Zobal, Lukáš, Dušan Kolář, and Radek Fujdiak (2019). "Current state of honeypots and deception strategies in cybersecurity". In: *2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. IEEE, pp. 1–9.