

# Honeypot Analysis Using Big Data Technologies

Aniruddha Salve<sup>1</sup> Ankit Gawade<sup>2</sup> Akshay Kadam<sup>3</sup> Suraj Kharade<sup>4</sup>

<sup>1,2,3,4</sup>Department of Computer Engineering

<sup>1,2,3,4</sup>SVPMs College of Engineering, Baramati, Pune, India

**Abstract** — In recent years, honeypots have become an essential tool for organizations to understand the malicious activities on the internet. They help collect patterns and activities done by intruders on the system, enabling security managers to prevent potential cyber-attacks. However, as the volume of data collected from honeypots increases, it becomes a significant challenge to analyze the data in a timely manner. It can take an extended period to identify attackers and prevent attacks, leading to a time-consuming task. To address this problem, integrating new methods of analysis or alert systems is necessary. In this work, we propose a new technique that integrates an alert system and predicts future attacks using machine learning time series algorithms such as ARIMA, SARIMAX, and big data. This technique is expected to provide more security to organizations and improve the efficiency of analysis and prevention of cyber-attacks.

**Keywords:** Honeypot, ELK Stack, Hadoop, Spark, Cybersecurity, Machine Learning, SARIMAX, ARIMA, Telegram Alert, Kafka.

## I. INTRODUCTION

Cybersecurity is a rapidly evolving field with ever-increasing threats [1] from malicious actors seeking to exploit vulnerabilities in computer systems. As a result, organizations must stay vigilant and continually adopt new methods and techniques to detect and prevent cyber-attacks. Honeypots are one such technique that organizations can use to identify cyber threats. A honeypot is a system or network designed to appear vulnerable to attackers, with the purpose of luring them in and monitoring their activities. By analyzing the behavior of attackers, security analysts can gain valuable insight into their tactics, techniques, and motivations.

There are various types of honeypots, including low-interaction honeypots and high-interaction honeypots. Low-interaction honeypots simulate vulnerable services and are easier to deploy, while high-interaction honeypots replicate entire operating systems and provide a more accurate representation of a real system. Additionally, organizations can deploy multiple honeypots to cover different areas of their network and gather more data.

One popular honeypot used by organizations is Cowrie, which is an open-source SSH/Telnet honeypot that simulates a vulnerable Linux system. Cowrie allows organizations to capture and analyze the [2] methods used by attackers attempting to exploit SSH or Telnet vulnerabilities. By analyzing the data collected from Cowrie, organizations can improve their cybersecurity posture and identify potential vulnerabilities before they can be exploited.

While honeypots are a valuable tool for detecting and preventing cyber-attacks, they also present certain challenges. One of the biggest challenges is analyzing the large amounts of data generated by honeypots, which can be time-consuming and resource-intensive. To address this challenge, organizations can leverage machine learning and

big data techniques to efficiently analyze and interpret the data collected from honeypots, allowing for more effective and efficient threat detection and response

## II. LITERATURE SURVEY

As cyber-attacks continue to increase, honeypots have become a popular method for identifying and analyzing cyber threats. In recent literature, two surveys have explored the use of honeypots in detecting attacks and designing intrusion detection systems. The first survey, [1] titled "Design of Intrusion Detection Honeypot Using Social Leopard Algorithm to Detect IoT Ransomware Attacks," proposes a honeypot that captures different types of attacks and provides an alarm system using the social leopard algorithm. The second survey, titled "Analysis of Attacks Using a Honeypot," describes the deployment of Dionaea Honeypot on a system to capture attacks and visualize attack patterns and information such as IP addresses and commands using different charts.

The first survey's proposed intrusion detection honeypot using the social leopard algorithm is a promising approach to detecting IoT ransomware attacks. The algorithm uses social network analysis to detect potential attacks by analyzing the behavior of users and groups within the network.

The second survey's [2] deployment of Dionaea Honeypot provides a useful way to capture and visualize attack patterns, allowing for a better understanding of the types of attacks targeting a system.

These surveys highlight the importance of using honeypots in detecting and analyzing cyber threats. They also suggest the need for more research on the development and implementation of effective honeypots for detecting various types of attacks. Overall, the use of honeypots and other intrusion detection techniques will continue to play a crucial role in ensuring the security of computer networks and systems.

## III. METHODOLOGY

The methodology for our research involves installing the Cowrie [4] honeypot on Ubuntu to capture and analyze attack patterns. We also utilized Hadoop, ELK stack, and Kafka for data collection and analysis. By using these tools, we aim to enhance the security of organizations by detecting and analyzing attack strategies, commands, and patterns to better understand the behavior of attackers and prevent potential threats.

### A. Data Gathering

The data gathering process involves the collection of logs and [1] JSON files generated by the honeypot whenever an intruder attempts to access the system. These logs and JSON files contain crucial information such as timestamps, IP addresses, and attempted usernames and passwords. The data is then stored in Hadoop [6], Kafka-topic, and Elasticsearch

[10] for further analysis and use. By collecting this data, we can gain insight into the patterns and characteristics of malicious activities on the internet, which can help improve the security of the system.

### B. Data Preprocessing

After collecting the data, the next step is to preprocess it. [1] We will be using Hadoop and Spark for big data processing [6]. The data preprocessing will include reading the data, selecting relevant features, and cleaning the dataset. We will transform the log or JSON data into a tabular format to perform machine learning and telegram alert system. This step will improve the quality of the dataset and make it suitable for further analysis.

### C. Model Building

To detect and forecast future attacks, we will be using machine learning algorithms. The first step is to clean the collected dataset and select relevant features. We will be utilizing the log data collected from the honeypot system and transform it into a tabular format suitable for machine learning algorithms.

In particular, we will be focusing on time-series machine learning algorithms such as ARIMA (Autoregressive Integrated Moving Average) and SARIMAX (Seasonal Autoregressive Integrated Moving Average with Exogenous Regressors). These algorithms are effective in identifying patterns and trends in time-series data, which is important for detecting and predicting cyber-attacks.

By utilizing machine learning algorithms, we can build models that can analyze large volumes of log data in real-time, identify patterns and trends, and forecast potential cyber-attacks. These models can also be used to generate alerts to security personnel, allowing them to take proactive measures to prevent cyber-attacks. Therefore, the development of accurate and efficient machine learning models is essential for improving the security of organizations and preventing cyber-attacks.

### D. Algorithms

#### 1) ARIMA

ARIMA (Autoregressive Integrated Moving Average) is a commonly used time series forecasting model that has been shown to be effective in predicting future cyber-attacks. In our research paper, we use ARIMA to predict future cyber-attacks based on the preprocessed dataset containing the date and count of attacks.

The ARIMA model involves three main components: autoregression (AR), differencing (I), and moving average (MA). Autoregression refers to using past values of the variable being forecasted to predict future values. Differencing is used to remove the trend and make the time series stationary. Finally, moving average uses past forecast errors to predict future values.

The ARIMA model is typically denoted as  $ARIMA(p, d, q)$ , where  $p$  is the order of the autoregressive component,  $d$  is the order of differencing, and  $q$  is the order of the moving average component. The ARIMA model can be further extended with seasonal components denoted as  $ARIMA(p, d, q)(P, D, Q)s$ , where  $P, D, Q$  are the seasonal

counterparts of  $p, d, q$ , and  $s$  is the number of periods in each season.

In our research, we use the ARIMA [9] model to predict the number of cyber-attacks that are likely to occur in the future based on historical data. We choose the appropriate values of  $p, d$ , and  $q$  using various statistical techniques such as autocorrelation and partial autocorrelation functions. We also experiment with the seasonal ARIMA model (SARIMA) to capture the seasonal variations in the dataset.

Furthermore, we also explore the potential of using Spark to fetch live data and combine it with the existing dataset to retrain the model for more accurate predictions. This will help to improve the accuracy of our model and make it more robust to real-time cyber-attacks.

#### Algorithm Steps

- Step 1: Import required libraries and modules
- Step 2: Initialize a Spark Session with a specific configuration
- Step 3: Read data from Hadoop at the specified URL
- Step 4: Filter out successful login counts
- Step 5: Group data by timestamp and aggregate the count of successful logins
- Step 6: Convert the Spark Data Frame to a Pandas Data Frame
- Step 7: Read a previous dataset from a CSV file and union it with the new data
- Step 8: Resample data to get count of attacks by day
- Step 9: Train an ARIMA model with parameters ( $ARIMA(p,d,q)$ )
- Step 10: Assemble the input features using Vector Assembler
- Step 11: Fit the ARIMA model on the assembled Data Frame
- Step 12: Use the ARIMA model to generate predictions for the future dates

#### 2) SARIMAX

SARIMAX (Seasonal Autoregressive Integrated Moving Average with Exogenous Variables) is an extension of the ARIMA model that incorporates exogenous variables in addition to the autoregressive, differencing, and moving average components. The exogenous variables are external factors that may influence the time series being forecasted, such as weather conditions, social media trends, or economic indicators.

In our proposed research paper, we aim to use SARIMAX for predicting future cyber-attacks by incorporating exogenous variables that are relevant to the domain of cyber security. By including these variables, we hope to improve the accuracy of our model and make it more robust to real-world scenarios.

The SARIMAX model is denoted as  $SARIMAX(p, d, q)(P, D, Q)s$ , where  $p, d$ , and  $q$  are the same as in ARIMA,  $P, D$ , and  $Q$  are the seasonal counterparts, and the exogenous variables are denoted by the letter  $X$ . The SARIMAX [9] model allows us to analyze the impact of exogenous variables on the time series being forecasted and to make more accurate predictions.

In addition, we plan to use Spark to fetch live data and incorporate it with the historical dataset to retrain the model in real-time. This will help us to improve the accuracy

of our predictions and enable us to respond quickly to any changes in the cyber security landscape.

Algorithm Steps

- Step 1: Import required libraries and modules.
- Step 2: Initialize a Spark Session with a specific configuration.
- Step 3: Read data from Hadoop at the specified URL.
- Step 4: Filter out successful login counts.
- Step 5: Group data by timestamp and aggregate the count of successful logins.
- Step 6: Convert the Spark Data Frame to a Pandas Data Frame.
- Step 7: Read a previous dataset from a CSV file and union it with the new data.
- Step 8: Resample data to get count of attacks by day.
- Step 9: Train a SARIMAX model with parameters (p, d, q)(P, D, Q, s).
- Step 10: Assemble the input features using Vector Assembler.
- Step 11: Fit the SARIMAX model on the assembled Data Frame.
- Step 12: Use the SARIMAX model to generate predictions for the future dates.

E. Dashboard

To present the insights of the collected data, a Dashboard is considered as the most essential part of the project. Different charts such as pie chart, line plot, and bar plot can be utilized to demonstrate the attack distribution and the use of attacks on a daily basis, as well as to display the count of usernames and password attempts made on the system. There are various ways to implement a Dashboard, but the ELK stack [10] (Elasticsearch, Logstash, and Kibana) was considered for this project. In addition, an alternative approach was considered, where a custom Dashboard was created using Logstash, Hadoop [6], Spark Streaming, and Flask to build a web app that displays the Dashboard in the form of various charts.

F. Telegram Alert

The Telegram Alert system is an essential part of the proposed system, which is designed to notify the security team in real-time about any potential security breaches. The system achieves this by making use of a Kafka-Consumer to pull data from the Kafka topic. The data is then filtered in real-time using pandas data frame to detect any successful events that indicate a security breach.

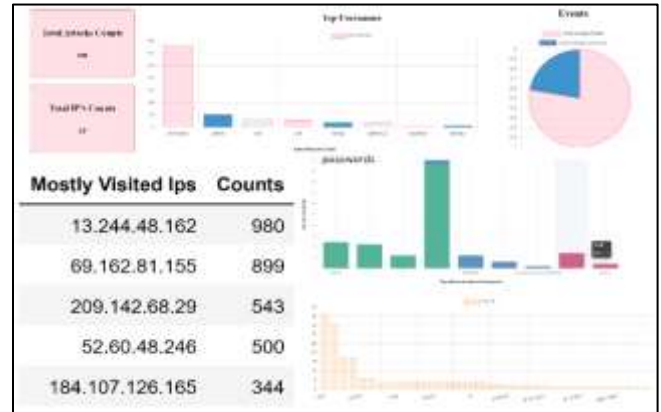
If a successful event is detected, the system sends an API call to Telegram to notify the designated security team member via Telegram. The notification includes information about the intruder such as their username, password, IP address, and timestamp of the intrusion. This ensures that the security team is alerted immediately and can take action to prevent any further damage. The use of Telegram as a notification system ensures that the security team is notified regardless of their physical location, enabling them to respond to any security threats promptly.

Algorithm Steps

- Step 1: Import necessary libraries such as Kafka, Pandas.
- Step 2: Create Kafka Consumer

- Step 3: Iterate the log data from Kafka topic using Kafka consumer
- Step 4: Create a data frame using pandas to perform filtration on the data
- Step 5: If a Success Event is found, call the Telegram API with the chatbot API and chat ID to send the message using Python
- Step 6: End

IV. EXPECTED RESULT



V. PROPOSED SYSTEM

Our proposed system is designed to enhance the security of organizations by detecting and analyzing the strategies used by attackers to intrude their systems. The system consists of several components, including the Cowrie honeypot, the ELK stack, Hadoop, and Spark. The system also incorporates machine learning Time Series algorithms such as ARIMA, and SARIMAX to predict future attacks and identify successful attack events.

The system provides a visual representation of the detected attacks using the ELK stack dashboard and a custom web application. In addition, the system alerts the security team of potential security breaches via telegram alerts, enabling them to take appropriate action to prevent the attack from succeeding.

The proposed system can be implemented in various organizations to enhance their security posture and provide timely alerts to prevent successful attacks.

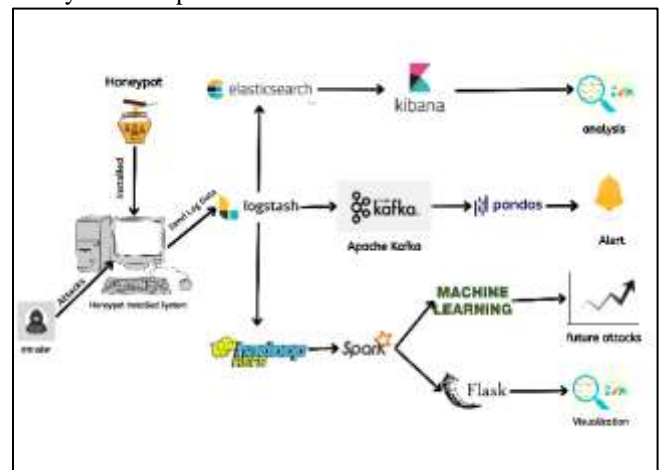


Fig. 1: System Architecture

## VI. REQUIREMENT ANALYSIS

### A. Operating Systems

The project requires Ubuntu 20.04 and Windows 11- 64 bit with 8 GB of RAM.

### B. Languages

The project requires proficiency in Python, HTML, and CSS for building the entire project.

### C. Tools Required

- Hadoop: For distributed storage and processing of large datasets.
- Spark: For fast and distributed data processing.
- Elasticsearch: For indexing and searching large datasets.
- Kibana: For data visualization and exploration.
- Logstash: For collecting, processing, and sending logs and other data.

### D. Technologies

- Big Data: For processing large datasets and extracting valuable insights.
- Machine Learning: For building predictive models and detecting anomalies in the data.
- Data Analytics: For analyzing data and extracting insights from it.

## VII. CONCLUSION

The proposed system using Cowrie honeypot, ELK stack, Hadoop, Kafka, and machine learning algorithm has shown great potential in detecting and predicting cyber-attacks. The results of this research indicate that by collecting and storing the data from the Cowrie honeypot, we can effectively visualize and analyze attempted attacks using Kibana and create our own visualization dashboard using Flask, HTML, and CSS. The implementation of Random Forest algorithm in Spark for predicting future attacks based on previous data has also yielded promising results.

Moreover, the integration of a telegram alert system using Kafka and Python to notify security personnel of successful login attempts has improved the efficiency of the overall cybersecurity system. Despite the promising results, this research can be further improved by incorporating more advanced machine learning algorithms and expanding the data sources to enhance the accuracy and coverage of the prediction models.

In conclusion, our proposed system provides a comprehensive and efficient approach to cybersecurity threat detection and prediction. It is a step forward in the development of more effective cybersecurity systems that can help organizations protect themselves from potential cyber threats.

## REFERENCES

- [1] S. Sibi Chakkaravarthy, D. Sangeetha, M. V. Cruz, V. Vaidehi and B. Raman, "Design of Intrusion Detection Honeypot Using Social Leopard Algorithm to Detect IoT Ransomware Attacks," in *IEEE Access*, vol. 8, pp. 169944-169956, 2020, doi: 10.1109/ACCESS.2020.3023764.
- [2] I. Koniaris, G. Papadimitriou and P. Nicopolitidis, "Analysis and visualization of SSH attacks using honeypots," *Eurocon 2013, Zagreb, Croatia, 2013*, pp. 65-72, doi: 10.1109/EUROCON.2013.6624967.
- [3] K. Chin and K. Omote, "Analysis of Attack Activities for Honeypots Installation in Ethereum Network," 2021 IEEE International Conference on Blockchain (Blockchain), Melbourne, Australia, 2021, pp. 440-447, doi: 10.1109/Blockchain53845.2021.00068.
- [4] S. Mehta, D. Pawade, Y. Nayyar, I. Siddavatam, A. Tiwart and A. Dalvi, "Cowrie Honeypot Data Analysis and Predicting the Directory Traverser Pattern during the Attack," 2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICES), Chennai, India, 2021, pp. 1-4, doi: 10.1109/ICES52305.2021.9633881.
- [5] A. A. Egupov, S. V. Zarehin, I. M. Yadikin and D. S. Silnov, "Development and implementation of a Honeypot-trap," 2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EConRus), St. Petersburg and Moscow, Russia, 2017, pp. 382-385, doi: 10.1109/EConRus.2017.7910572.
- [6] M. S. Tok, M. Dener and M. Demirci, "Processing Honeypot Logs with Big Data and Data Visualization via Hadoop- Power BI Integration," 2022 15th International Conference on Information Security and Cryptography (ISCTURKEY), Ankara, Turkey, 2022, pp. 49-54, doi: 10.1109/ISCTURKEY56345.2022.9931797.
- [7] N. Bhagat and B. Arora, "Intrusion Detection Using Honeypots," 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC), Solan, India, 2018, pp. 412-417, doi: 10.1109/PDGC.2018.8745761.
- [8] J. R. Kondra, S. K. Bharti, S. K. Mishra and K. S. Babu, "Honeypot-based intrusion detection system: A performance analysis," 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2016, pp. 2347-2351.
- [9] Duque Anton, Simon & Ahrens, Lia & Fraunholz, Daniel & Schotten, Hans. (2018). Time is of the Essence: Machine Learning-based Intrusion Detection in Industrial Time Series Data.
- [10] H. Almohannadi, I. Awan, J. Al Hamar, A. Cullen, J. P. Disso and L. Armitage, "Cyber Threat Intelligence from Honeypot Data Using Elasticsearch," 2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA), Krakow, Poland, 2018, pp. 900-906, doi: 10.1109/AINA.2018.00132.