ANOMALY DETECTION IN HONEYPOT USING CLUSTERING METHODS

Project work submitted to Avinashilingam institute for Home Science and Higher Education for Women

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

SUBMITTED BY

REVATHI S

UNDER THE GUIDANCE OF

Mrs. S. KARTHIKA M.C.A., M.Phil. NET

Assistant Professor (Dept of IT)

Project Report Submitted

In partial fulfilment of the requirements for the Award of

DEPARTMENT OF INFORMATION TECHNOLOGY

Avinashilingam Institute for Home Science and Higher Education for Women

Coimbatore – 641043

MAY – 2023

DECLARATION

I hereby declare that the project entitled "**ANOMALY DETECTION IN HONEYPOT USING CLUSTERING METHODS**" Mrs. S. KARTHIKA M.C.A., M.Phil. NET Assistant Professor (Dept of IT) **is** a record of the original work done by **Revathi S**(21PIT006) under the guidance of Assistant Professor and Head, Department of Information Technology, School of Physical Sciences and Computational Sciences, Avinashilingam Institute for Home Science and Higher Education for Women inthe partial fulfillment for the award of the degree of Master of Science in Information Technology, and this project work has not formed the basis for any Degree/Diploma/Associates.

Place: Coimbatore

Date: 19.05.2023

S. Revathi

Signature of the candidate

Countersigned By,

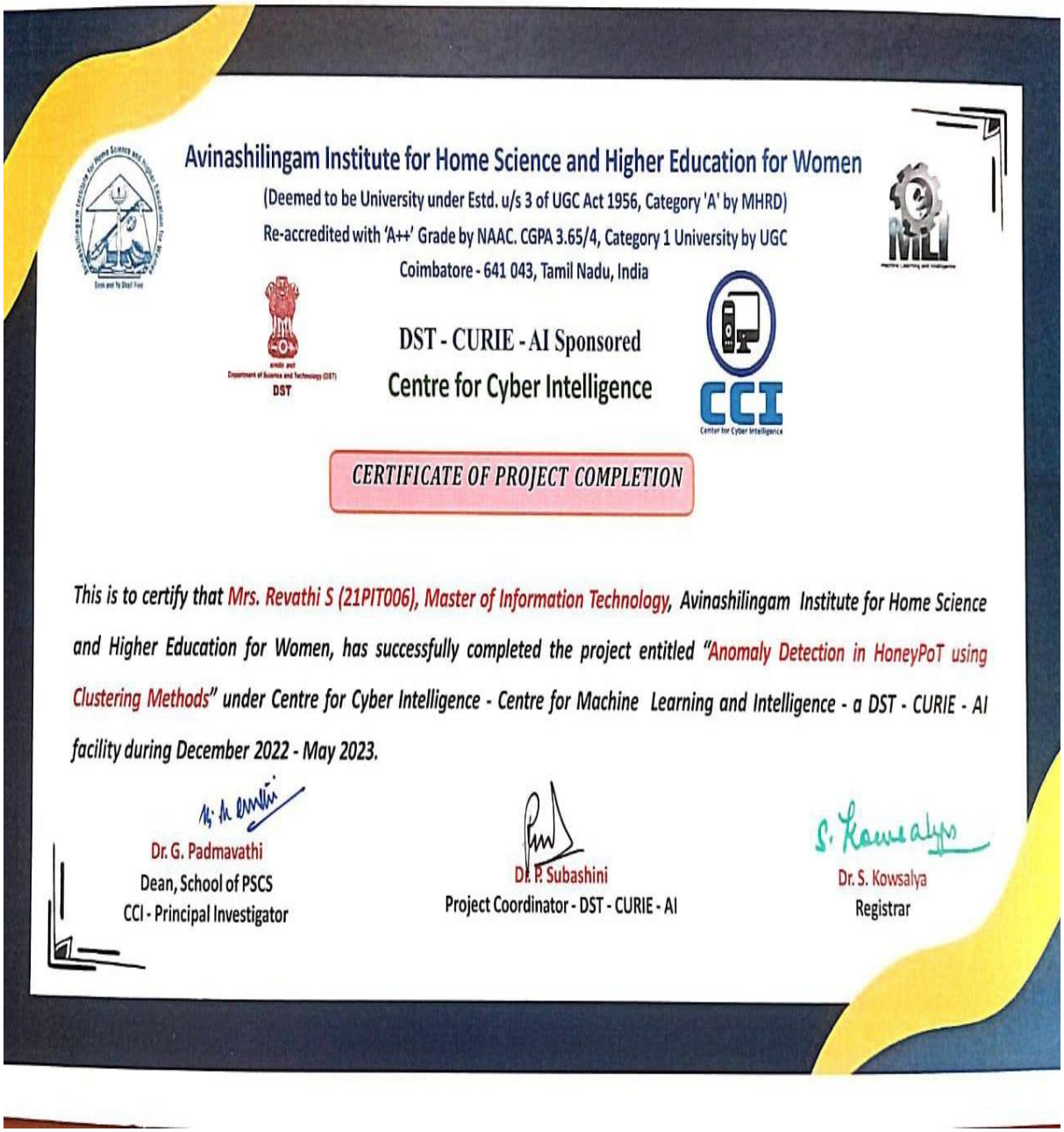
19/5/23

Mrs. S. KARTHIKA M.C.A., M.Phil. NET

Assistant Professor (Dept of IT)

Department of Information Technology,
School of Physical Sciences and Computational Sciences.

CERTIFICATE



Avinashilingam Institute for Home Science and Higher Education for Women

(Deemed to be University under Estd. u/s 3 of UGC Act 1956, Category 'A' by MHRD)

Re-accredited with 'A++' Grade by NAAC. CGPA 3.65/4, Category 1 University by UGC

Coimbatore - 641 043, Tamil Nadu, India

DST - CURIE - AI Sponsored

Centre for Cyber Intelligence

CERTIFICATE OF PROJECT COMPLETION

This is to certify that Mrs. Revathi S (21PIT006), Master of Information Technology, Avinashilingam Institute for Home Science and Higher Education for Women, has successfully completed the project entitled "Anomaly Detection in HoneyPoT using Clustering Methods" under Centre for Cyber Intelligence - Centre for Machine Learning and Intelligence - a DST - CURIE - AI facility during December 2022 - May 2023.

Dr. G. Padmavathi
Dean, School of PSCS
CCI - Principal Investigator

Dr. P. Subashini
Project Coordinator - DST - CURIE - AI

Dr. S. Kowsalya
Registrar

CERTIFICATE

This is to certify that this project work entitled "**ANOMALY DETECTION IN HONEYPOT USING CLUSTERING METHODS**" done by **REVATHI S (21PIT006)** has been submitted to Avinashilingam Institute for Home science and Higher education for women, Coimbatore-641 043 inpartial fulfilment of the requirement for the award of the degree of **MASTER OF SCIENCE IN INFORMATION TECHNOLOGY.** This Project has not foundthe basis for the award of any Degree/Associate/fellowship or similar title to any Candidate of any University. Certified as a Bonafide record of the work submitted for the Viva voce held on___

Signature of the Head of the Department

4/5/2023

Signature of the Supervisor

Signature of External Examiner(s)

ACKNOWLEDGEMENT

I sincerely thank the **Lord Almighty** and **My lovable parents** for showering their generous blessings upon me in all endeavors.

I wish to express my gratitude to **Prof.S.P.Thyagarajan,** Chancellor, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for providing thefacilities to conduct this study.

I extend my thanks to **Dr. Bharathi Harishankar, Ph.D., FRSA.,** Vice Chancellor, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for providing flamboyant help towards the completion of the study.

I record my deep sense of gratitude and indebtedness to **Dr. S. Kowsalya,** M.Sc., M.Phil., Ph.D., Registrar, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for providing adequate help for the study

I grateful record my sincere thanks to **Dr. G. Padmavathi** M.Sc., M.Phil., Ph.D., Dean and Professor, School of Physical Sciences & Computational of Sciences, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for timely help rendered throughout the course of this work.

I heartily Thanks to **Dr. Mrs. D. Shanmugapriya** M.Sc., M.Phil., Ph.D., SET Head of Department of Information Technology for the valuable guidance and encouragement during the course of our project.

I heartily Thank my esteemed project guide Assistant **Professor, Mrs.S. Karthika M.C.A., M.Phil.NET** Department of Information Technology, for imparting tremendous assistance and well-timed support for triumph of our project.

I express my honorable thanks to our project coordinator Department of Information Technology, for imparting tremendous assistanceand well-timed support for triumph of our project.

I sincerely thank all **the staff members** of Department of Information Technology Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for their help and support.

I like to extend my gratitude to Ms.A. Roshini, Technical Assistant –Center of cyber intelligence, Department of Computer Science, For providing Project guidelines and always supported me and encouraged me with valuable advice and Profound belief in my work and abilities.

I would like to acknowledge the help rendered by Center for Cyber Intelligence, DST -CURIE – AI Sponsored Phase II for providing the laboratory facilities to execute my project.

I would like to express my special thanks to **my parents, my friends** and all **my well-wishers** for their constant encouragement, support and help in carrying out this work successfully.

ABSTRACT

Aim: To identify and isolate abnormal behavior in network traffic. Clustering methods are machine learning techniques that can be used to group similar network traffic data together, whereas honeypots are decoy systems created to draw in and catch hostile actors. This project deals with the Honeypot data from sensors Dionaea installed at the Information Lab of the University of Muhammadiyah Malang, from January 2019 to December 2019, there were around 6.000,000 more attacks, the average attack reaching 500 to 700 attacks every day. This data is only in the form of attack data (Date, Sensor, Country, Src, IP, etc., port, Protocol, Honeypot Sensor).

Method: The detection of clustering using a unsupervised machine learning approach comprises five phases. Phase 1 is Data Acquisition. In Phase 2 is the Data Preprocessing method, which transforms the dataset and resamples the minority of attackson the datasets. In Phase 3, Outlier analysis feature selection methods are used to select the features in the dataset. In Phase 4, Exploratory data analysis (EDA) is a method for analyzing and comprehending data that places a strong emphasis on using graphical and statistical tools to look for patterns, trends, and relationships within a dataset. To find patterns and links in the data, EDA frequently starts by creating several visualizations, including scatter plots, histograms, box plots, and heat maps. In phase 5, To detect the anomalies the data points are grouped using the machine learning approach of clustering into groups or clusters that share similar traits. K-means and KNN are two popular clustering methods for anomaly identification. Data anomalies can be found using both the K-means and KNN clustering techniques. These algorithms can aid in the detection of potential threats and abnormalities in network traffic data by clustering data points and identifying those that do not fit within the predicted clusters of typical behaviour.

Result: From this proposed system, Comparing to K-means, K-Nearest Neighbors (KNN) score is more effective

Keywords: **Anomaly Detection, Clustering, Honeypots, Unsupervised learning**

TABLE OF CONTENT

CHAPTER 1

1. INTRODUCTION

One of the crucial issues in the digital world today is the problem of security. Security issues in the computer systems and data is very challenging. Any attempt to compromise the integrity, confidentiality and availability of information or resources is called an intrusion. There are two types of intrusion, Technical and non-technical. Technical intrusion involves using technical tools and expertise to perform the intrusion. Non-technical intrusion involves using any non-technical means to perform the intrusion

1.1 INTRUSION DETECTION SYSTEM

Network intrusion detection system is one of the essential parts of infrastructure protection paradigm. Although varieties of approaches are employed to protect essential data in current networked scenario. Providing security for internet is a challenging task; it must be concentrated on the following includes protection, detection, reaction, and recovery. Intrusion detection process adequately carried out the rest of the process get the success; otherwise, it leads to becoming a thread (Weiming Hu et al. 2008). Network intrusions and Host intrusions are the two classifications of Intrusions. In HIDS audit data is used for the analysis, and it's taken from the target host system. The disadvantage of this method is very difficult to prevent the attacks; sometimes, the hackers are modified the audit data (Weiming Hu et al. 2008). Intrusion detection is one of the processes of monitoring computers or data flow in-network for unauthorized access, behavior, or modifying the data.

The attackers accessing a system from the internet, by authorized users of the systems who attempt to capture additional privileges, and by authorized users who misuse privileges given to them. Intrusion Detection System (IDS) architecture is shown in Figure 1.1 has been developed to identify any unauthorized attempts or successful attacks on any type of monitored data or resources available as part of a network or host system. Most IDS detects such malicious activity either at the transaction level or at the Operating System (OS) level.

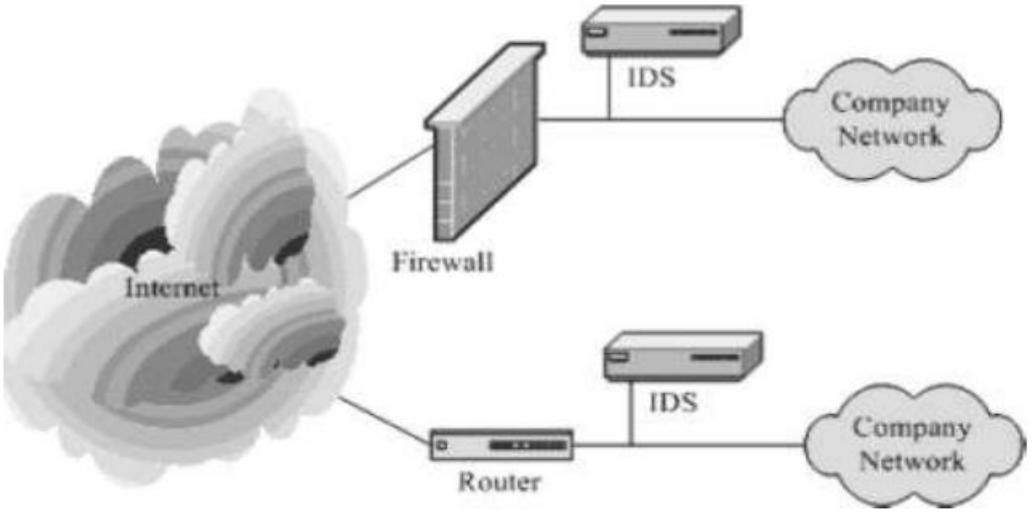


Figure1.1 Intrusion detection system

Intrusion Detection and Prevention System (IDPs) fundamentally focused on identifying possible incidents, logging information about them, and reporting attempts. Many IDPs respond to a detected threat by trying to prevent it from succeeding. They use a variety of response techniques, which involve the IDPS stopping the attack itself, changing the security paradigm, or changing the attack's content.

1.2 INTRUSION DETECTION TECHNIQUES

There are two types of techniques namely

- Misuse Detection Technique
- Anomaly Detection Technique

1.2.1 Misuse Detection

Misuse detection is one of the approaches for detecting attacks. In this approach, first, address the abnormal system behavior, and then define the rest of the activities as normal one. A misuse detection IDS model similarly reflects the system antivirus software. It analyze the performance of system or network scenario and compare the network behavior activity against signatures of a known intrusive system. Intrusion detection systems that follow the misuse detection process mandatorily updated continuously to take the lead against the hackers.

1.2.2 Anomaly Detection

Anomaly detection represents the problem of identifying patterns in data that didn't conform to the expected behavior. With anomaly intrusion detection, the system knows what is normal behavior and attempts to search for other types of behavior that consider suspicious. While in misuse intrusion detection, the system knows what a suspicious behavior is and attempts to search for patterns that match its knowledge of suspicious behaviors. Combining the two methods will yield a better detection rate. Also having reactive intrusion detection as a second line of defense improves security. Finally, one must not forget that intrusion detection is a part of the big security picture and must be combined with other security systems to have as close to a secure environment as possible. Anomaly detection, one method is referred to as outlier detection; it deals with detecting patterns in a given data set that didn't conform to an already known normal behavior. The patterns thus identified are called anomalies and often translate to critical and actionable information in many application domains. Anomalies are also referred to like change, deviation, surprise, aberrant, peculiarity, intrusion, etc.

There are several types of anomaly detection techniques, including:

Statistical methods: Statistical methods use mathematical models to identify data points that are significantly different from the normal distribution of data. These methods may include Gaussian distributions, regression analysis, or hypothesis testing.

Machine learning methods: Machine learning methods use algorithms that are trained on normal data patterns to identify outliers. These algorithms can be supervised, unsupervised, or semi-supervised.

Rule-based methods: Rule-based methods use predefined rules or thresholds to identify anomalous data points. These rules may be based on specific business rules, domain knowledge, or regulatory requirements.

Ensemble methods: Ensemble methods combine multiple anomaly detection algorithms to improve accuracy and reduce false positives. These methods may use a combination of statistical, machine learning, or rule-based techniques.

1.3 TYPES OF INTRUSION DETECTION SYSTEM

Intrusion Detection System classified into two types.

- Network-based Intrusion Detection System (NIDS)
- Host-based Intrusion Detection System (HIDS)

1.3.1 Network-Based Intrusion Detection System

The importance of Network Intrusion Detection System (NIDS) is a system that tries to identify misbehaviour activity like as denial of service attacks, scanning of ports or even trying to crack into computers by Network Security Monitoring (NSM) of network traffic. NIDS scans all the incoming packets and tries to find unauthorized patterns called as signatures or rules.

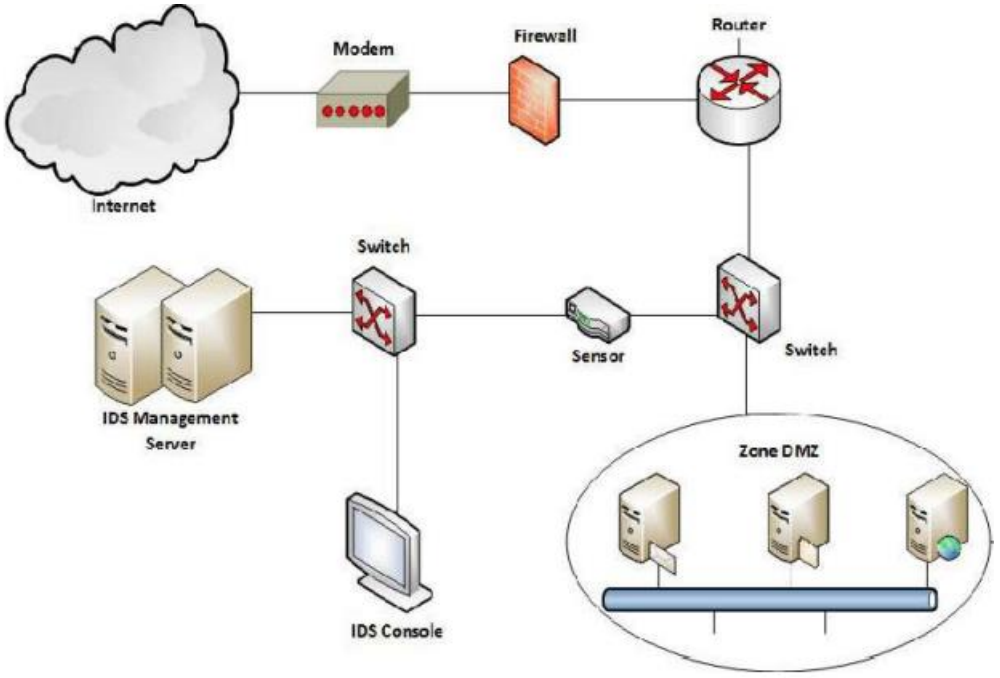


Figure1.3.1 Network-based intrusion detection system

The NIDS then scans any traffic that is flowing over that segment of the network, as shown in Figure 1.3.1 The NIDS function reflects the same way as high-end antivirus applications, and it is comparing each transmitted packet against the signature or pattern file method. The IDS functions in a concrete way to increase packet throughput as inspecting every packet can slow traffic considerably. Further, An IDS uses the firewall methodology while checking the packet by letting through the packets that are not harmful to the system. Preprocessing filters carried out this task for IDS.

1.3.2 Host-Based Intrusion Detection System (HIDS)

Much as a NIDS will dynamically verify the network packets, HIDS are run on individual hosts or machine on the network. A HIDS tracks the inbound and outbound packets from the device only and alerts the user or administrator of abnormal activity is detected.

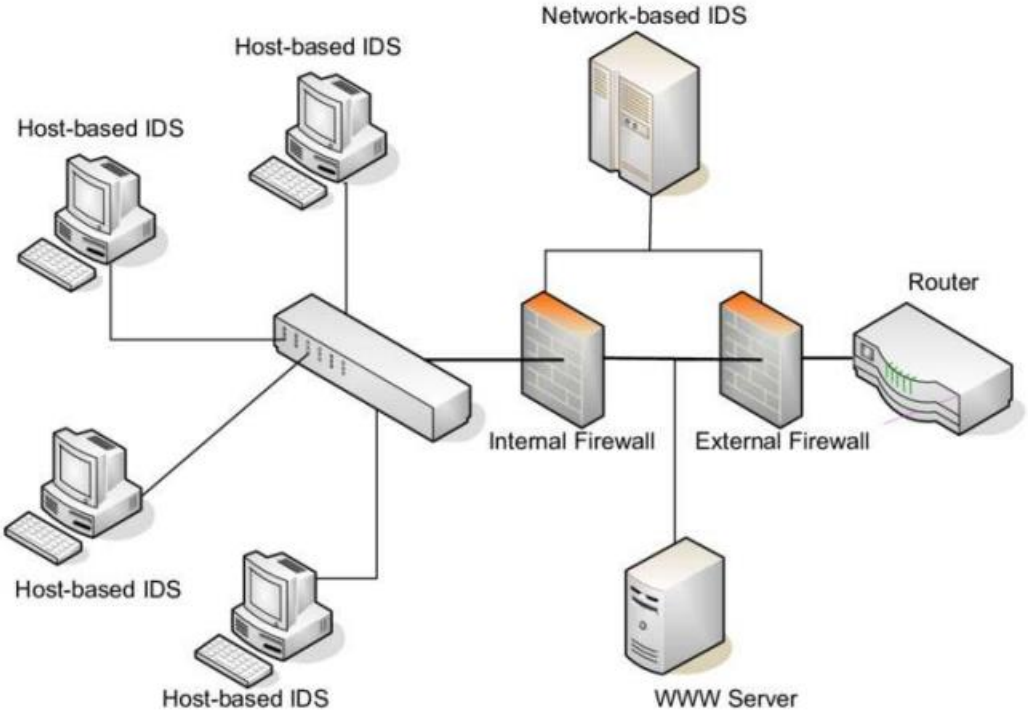


Figure 1.3.2 Host-based intrusion detection system

HIDS can be installed on various types of machines named servers, workstations, and notebook computers, as shown in Figure 1.3.2 Traffic transmitted to the host is taken into account and analyzed, passed onto the host if there are no potentially malicious packets during the data transmission. HIDS are fundamentally more focused on the local machines rather than compared to the NIDS. NIDS focus mostly on the network those specific hosts themselves.

1.3.3 Collaborative Intrusion Detection

An effective Collaborative Intrusion Detection Network (CIDN) allows distributed Intrusion Detection Systems to enhance the intrusion detection capability, and share the knowledge about intrusion for detecting new type of attacks.

1.4 INTRUSION PREVENTION SYSTEM

IPS also knew as intrusion detection and prevention systems (IDPS) are network security processes that continuously monitor network and system activities for detecting malicious behavior. The major classification of IPS are Host based IPS and Network based IPS.

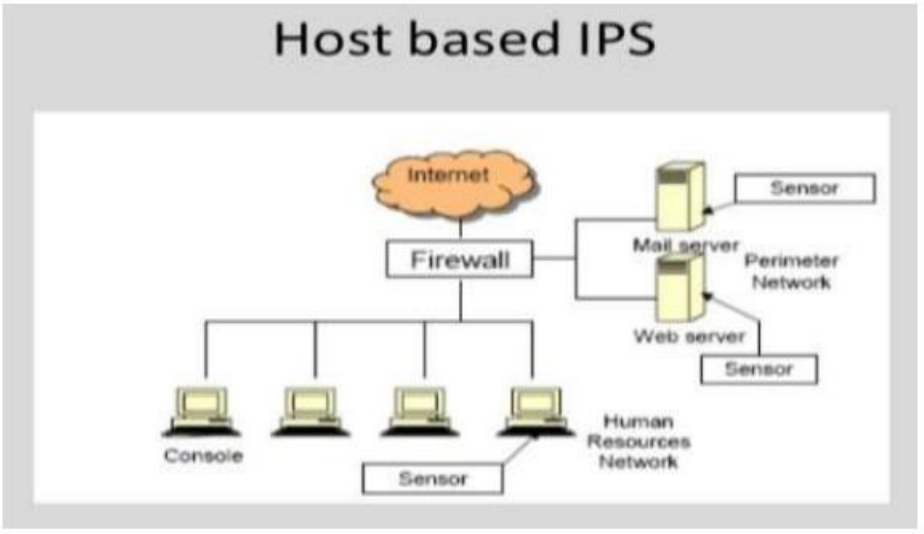


Figure 1.4.1 Host-based IPS

The Host-based intrusion prevention system (HIPS) is a method of security that relies on third-party software tools to detect and prevent malicious activities HIPS are specially used to protect endpoint devices. Once the malicious activity is identified, the HIPS tool can take multiple actions, including raising the alarm to the concern computer user, logging the malicious activity for future processing, connection resetting, dropping malicious packets and blocking further traffic from the suspect IP address.

Most HIPS use known attack patterns, signatures to identify malicious activity. Signature-based detection is efficient, but it can only protect the host device from the known attacks. It didn't protect against zero-day attacks or signatures that are not available in the database repository. Since stateful analysis is known about the actual packet contents, the chances for false positives are moreover lower than statistical anomaly detection method. For example, McAfee's Host Intrusion Prevention for Desktop and Dell's Managed Intrusion Prevention System (IPS) service are just two offerings that rely on multiple approaches to intrusion prevention.

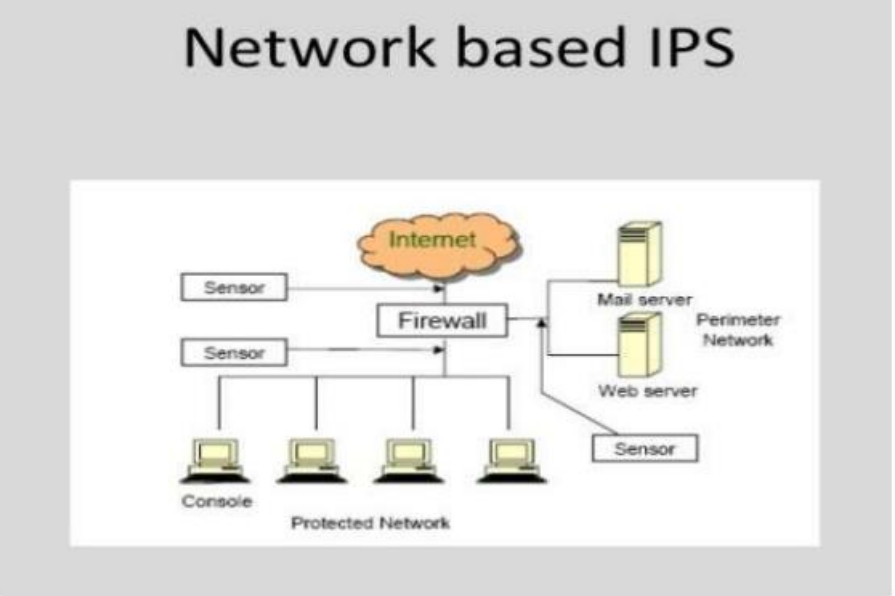


Figure 1.4.2 Network-based IPS

A network-based intrusion prevention system (NIPS) is a system that helps to monitor a network traffic and also concentrate on confidentiality and data integrity as well as network availability. The main functions include protecting the network against threats, such as a denial of service (DoS) and unauthorized access. The NIPS monitors and track the network for malicious activity or suspicious traffic in a network by analyzing the activity of protocol. If the NIPS is installed in a network, it can create physical security zones in a network. It makes the network intelligent and quickly discerns good traffic from bad traffic. Also stated that the NIPS becomes like a prison for hostile traffic such as Trojans, worms, viruses, and polymorphic threats.

1.5 HONEYPOTS

Honeypot is defined as a security resource whose value lies in being probed, attacked, or compromised. Unlike production systems, honeypots are resourcing whose main function is to be probed, compromised and attacked. If a honeypot cannot be attacked and probed, it will have no value. Firewalls and IDSs are security tools that are developed to take care of specific problem. When we come to honeypots, they are highly flexible and have a capability of addressing various kinds of attacks. Similar to firewalls, honeypots can deter attacks. They can also detect intrusions, which are the functionality of IDSs. Honeypots are used to study the activities of the blackhat community. They are used to capture and analyse different attacks or act as an early warning security entity.

Honeypot does not have any production value. Literally, no one should communicate or connect to a honeypot. If anyone is connecting to a honeypot, then it is most likely a probe, scan or attack. Honeypots can be compromised by attackers and used to attack other systems. If a honeypot tries to initiate an outgoing connection, the system has most likely been compromised. To illustrate the working principle of honeypot, one can see a typical scenario on a network system that deploys network-based honeypot. Suppose we have an organization having multiple hosts and service providing servers. In addition, a honeypot system can be configured to decoy and lure attackers. The honeypot does not offer any kind of service. Therefore, no one should attempt to make connection to it. If anyone connects to the honeypot, then it is either an attack or a probe. If any system or server inside the organization's network makes connection to the honeypot, such as one of the web servers, this server may have been attacked and compromised by attackers. Thus, honeypots are used to detect attacks and get alerts for unauthorized activities. In addition to detection, the other purpose of a honeypot is to slow down or stop the attacks coming from the internal network. A typical honeypot system can be developed to detect connection attempts destined to a system that does not exist. This kind of honeypot can forge replays to the attackers on the behalf of non-exiting systems. The false replays sent by the honeypot are specially crafted replay packets to keep a connection session of the attacker and slow down or stop the attack. An example of simplified honeypot architecture, which is implemented in the demilitarized zone (DMZ), is shown in figure 1.5.1

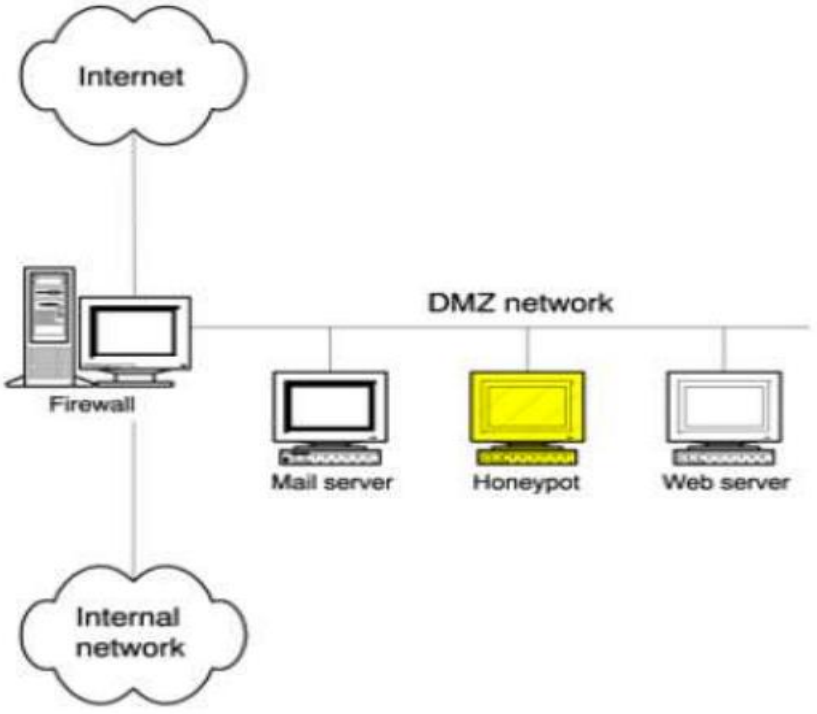
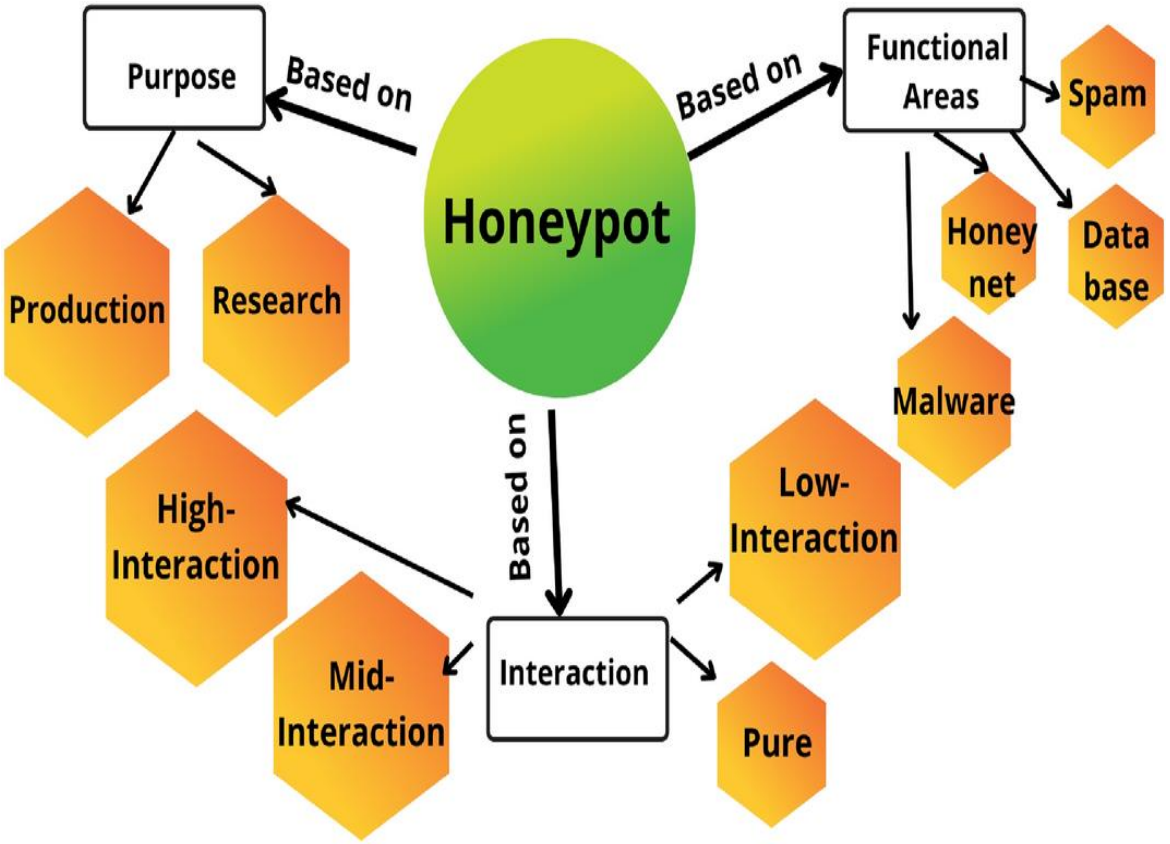


Figure 1.5.1 Honeypot implementation on a DMZ to detect attacks

1.6.1 Types of Honeypots

Honeypots do not address specific problem as in the case of firewalls and intrusion detection systems. The value and significance of honeypots deployment depend on the way they are built and used. Honeypots have some advantages as well as disadvantages that can affect their value. Next, the advantages and the disadvantages of honeypots will be presented.



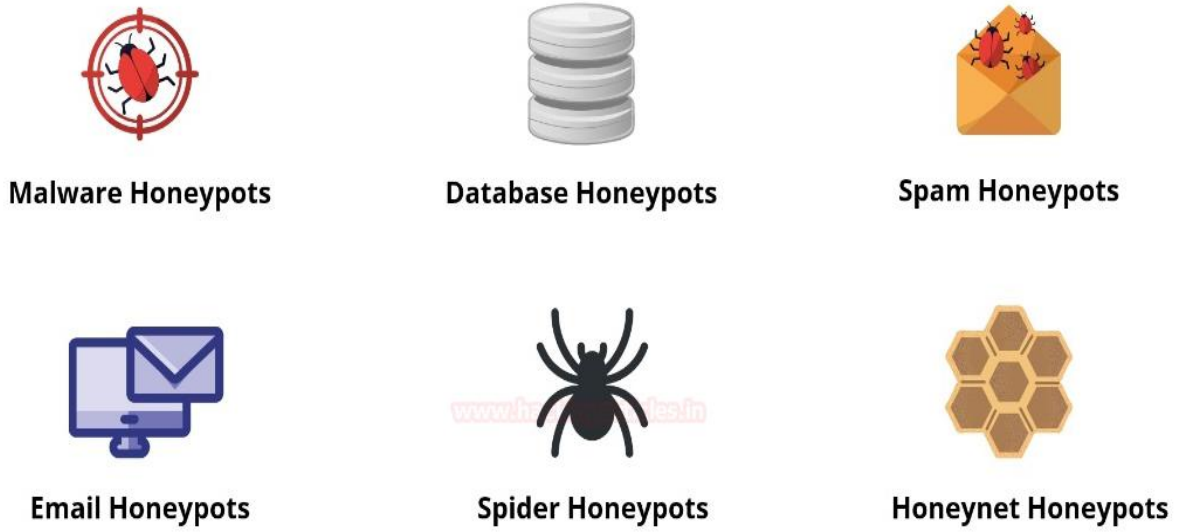


Figure 1.6 Types of Honeypot

Pure Honeypot:

A functioning replication system for production that is portable between servers. It has many sensors and false "confidential" data on board.

High-Interaction Honeypot:

- These honeypots need a lot of resources and require more upkeep, but they can produce valuable results.
- A high-interaction honeypot is made to entice intruders to stay inside for as long as feasible. This increases the security team's chances of observing the aims and motives of the attacker and of identifying systemic weaknesses.

Mid-Interaction Honeypot:

- They don't have their own operating system, and their main purpose is to divert the attention of the attackers long enough for the security team to respond to the breach.

- Their goal is to disorient or delay an attacker so that the organisation has more time to decide how to respond to the specific type of attack.

Low-Interaction Honeypot:

- When building a production environment, this type is frequently used. It is employed for sophisticated warning spotting systems. They are fairly easy to deploy and keep up.

- Low-interaction honeypots use fewer resources and acquire basic information about the type of threat and its origin. These utilise network services, the Transmission Control Protocol (TCP), the Internet Protocol (IP), and are reasonably easy to set up. However, there is nothing in the honeypot that may keep the attacker's interest for a long period.

Malware honeypots:

- It is common practise to reproduce malware and use attack vectors to spot it.

- Malware honeypots use well-known attack mechanisms to draw in malware. They can mimic something like a USB storage device, for instance. The honeypot tricks the malware into attacking the simulated USB whenever a PC is attacked.

Spam Honeypots:

- It is primarily set up to act as a mail relay and an open proxy. It would identify and stop a lot of spam emails.

- Open proxies and mail relays are used in spam honeypots to entice spammers. By utilising them to send themselves an email, spammers test mail relays. If they are successful, they will be able to send out a lot of spam. A spam trap can detect a spammer's attempt to send spam and then stop it in its tracks.

Database honeypots:

- These honeypots are useful for preventing actions like SQL injections, which firewalls commonly fail to detect.

- used to create fake databases to draw database-specific assaults like SQL injections, which illegitimately control data. An application firewall for databases can be used to implement these honeypots.

Client Honeypots:

- They aid in identifying hostile servers that are primarily in charge of attacking clients. It operates using virtualization technology and employs a containment technique to reduce the risk that has been imposed.

- Client honeypots try to draw malicious servers that attackers employ while hacking clients into their network. They assume the role of a client to watch as an attacker modifies a server during the assault. In order to lessen the risk of exposure to the researchers, client honeypots are frequently operated in a virtualized environment and with containment safeguards.

Honeynets:

- honeynets are a single system made up of multiple honeypots. The honeynets' goal is to tactically contain all kinds of inbound and outgoing traffic while tracing the attacker's intentions and techniques.

- A honeypot network makes up a honeynet. Numerous sorts of assaults, including as distributed denial-of-service (DDoS) attacks, attacks on content delivery networks (CDN), and ransomware attacks, can be examined using various types of honeypots constituting a honeynet. A honeynet contains all traffic, both inbound and outbound, to safeguard the rest of the organization's system while being used to investigate various assaults.

1.6.2 Honeypot Detection Tools

Honeypot detection tools are used to monitor honeypots and detect any unauthorized access or activity. There are several honeypot detection tools available, each with its own strengths and weaknesses. Here are some examples of popular honeypot detection tools:

- **Snort:** Snort is an open-source intrusion detection/prevention system that can be used to monitor honeypots. It uses signature-based detection to identify known attack patterns and can also be configured for anomaly detection.

- **Suricata:** Suricata is an open-source intrusion detection/prevention system that can be used to monitor honeypots. It uses signature-based detection and anomaly detection to identify suspicious activity.

- **OSSEC:** OSSEC is an open-source host-based intrusion detection system that can be used to monitor honeypots. It uses log analysis and file integrity monitoring to identify unauthorized access or modifications.

- **Honeyd:** Honeyd is a low-interaction honeypot that can be used to detect and track attackers. It simulates a variety of services and protocols to attract attackers, and logs their activity for analysis.

- **Cowrie:** Cowrie is a medium-interaction SSH honeypot that can be used to detect and track attackers. It simulates an SSH server and records attackers' activity for analysis.

- **Kippo:** Kippo is another medium-interaction SSH honeypot that can be used to detect and track attackers. It simulates an SSH server and records attackers' activity for analysis.

- **Dionaea:** Dionaea is a low-interaction honeypot that can be used to detect and track attackers. It simulates a variety of services and protocols to attract attackers, and logs their activity for analysis.

- **Glastopf:** Glastopf is a web application honeypot that can be used to detect and track attackers. It simulates a vulnerable web application and records attackers' activity for analysis.

Honeypot detection tools are used to monitor honeypots and detect any unauthorized access or activity. There are several honeypot detections tools available, including open-source intrusion detection/prevention systems such as Snort and Suricata, host-based intrusion detection systems such as OSSEC, and honeypot tools such as Honeyd, Cowrie, Kippo, Dionaea, and Glastopf. Each of these tools has its own strengths and weaknesses, and the choice of tool will depend on the specific use case and security requirements.

1.6.3 Advantages of Honeypots

The main advantage of honeypot is their ability to discover and trace new attacks. In case of IDS, it is impossible to detect attacks that do not have a previous signature or recorded anomaly. Honeypots have several advantages. Some of the advantages are listed below.

Data value - One of the problems in the Information Technology security mechanisms is the huge amount of data collected in the form of system logs, IDS alerts and firewall logs. The huge amount of data gathered makes the extraction of valuable information very cumbersome and difficult. Since honeypots collect a small amount of data that have higher value, they have no such drawbacks. Honeypots do not have production activities. Any data coming to them is most likely probes, attacks and scans. This helps them to reduce the noise level.

Resource - In most security mechanisms, resource exhaustion is a critical issue. In case of an IDS and firewall, the traffic analysed is too high. The huge amount of data may overwhelm or even lead them to stop working due to their limited capacity like buffer size overflow. Honeypots only capture those activities that are directed to them. By nature, the traffic coming to honeypots is less. In case of honeypots such resource limitation and malfunction due to exhaustion is not an issue. The other benefit of honeypots is that they do not require high performance computers. Compared to firewalls and IDS, they do not require high processor speed, vast amounts of RAM or large disk drive.

Simplicity - The other advantage of honeypots is the simplicity of deploying and maintaining. Unlike signature based IDSs, which need to be configured every time new threats are discovered, honeypots are very reliable due to their simplicity. In fact, deploying high interaction research honeypots is somewhat complex. However, once they are configured and installed one can simply watch them while they capture attacks. Due to their simplicity, misconfiguration and failure probability is very less.

Return on investment - When firewalls are deployed and becomes successful, managers of organizations may think that there is no attack. In case of honeypots, they quickly show and prove their value. When attackers are captured in the honeypots, people can be aware of the presence of attackers.

1.6.4 Disadvantages of Honeypot

Honeypots can be an effective security tool for detecting and analyzing attacks, there are also some disadvantages to using them:

- **Resource-intensive:** Honeypots require resources such as computing power, memory, and storage space. Running multiple honeypots can be resource-intensive, and the additional load on the network can also impact performance.

- **False positives:** Honeypots can generate false positives if they are not properly configured or if attackers use evasion techniques. False positives can lead to unnecessary alerts and wasted resources.

- **False sense of security:** Honeypots can give a false sense of security if they are not properly integrated into the security infrastructure and response processes. They should be used as part of a broader security strategy and not relied on as the sole security measure.

- **Legal and ethical concerns:** Depending on the jurisdiction, the use of honeypots may be subject to legal and ethical concerns. For example, if honeypots are used to lure attackers into committing illegal acts, there may be legal implications.

- **Maintenance and management:** Honeypots require ongoing maintenance and management to ensure that they remain effective. This can include updating software, monitoring logs, and analyzing data.

Honeypots have several disadvantages, including being resource-intensive, generating false positives, giving a false sense of security, legal and ethical concerns, and requiring ongoing maintenance and management. These issues should be carefully considered before implementing honeypots as part of a security strategy.

1.7 INTRODUCTION TO DATA MINING

Data mining is the process of bringing out valid, authentic, and actionable information from large databases. It is a logical process designed to explore data in search of consistent patterns and well-ordered relationships between variables, and then to confirm the findings by appealing the detected patterns against the new subsets of data. Prediction is the supreme goal of data mining - and predictive data mining is one of the primary types of data mining, and one that has connected with most direct business applications. The KDD cup data set process consists of many steps. They are, Data Cleaning - removing noise and inconsistent data

Data Integration - multiple data sources may be combined.

Data Selection - data relevant to the analysis task are retrieved from the database.

Data Transformation - data are consolidated into forms appropriate for mining by performing or aggregation operations.

Data Mining - Intelligent methods are applied in order to extract data patterns.

Pattern Evaluation - Identify the truly interesting patterns representing knowledge based on certain measures.

Knowledge Representation - Visualization of the mined knowledge to the user.

Data mining is the investigation step of the knowledge discovery in databases process, or standard benchmark databases like KDD, a relatively young and interdisciplinary field of computer science is the method of discovering new patterns from the bank of data sets involving methods at the intersection of artificial intelligence, machine learning, statistics and database systems. The importance of data mining is to extract knowledge from a data set in a human-understandable structure and involves data management, data preprocessing, model and inference considerations, interestingness metrics, complexity considerations, post-processing of found structure, visualization and online updating. In recent years, the use of data mining knowledge for intrusion detection system has won more and more attention, but there are a lot of problems.

1.8 CLUSTERING TECHNIQUE

Clustering is a process which splitting a given data set into equivalent groups based on given features such that similar objects are included in a group whereas non identical objects are in different groups. It is the superior unsupervised learning problem. It deals with the identical structure in a collection of unlabeled data. For better understanding, please refer to Figure 3.2

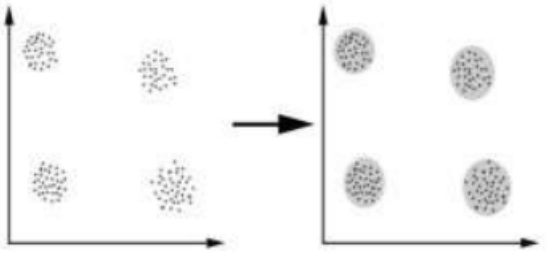


Figure1.8 Clusters formed from the set of unlabeled data

The Clustering algorithm to be advantageous and beneficial some of the conditions need to be satisfied. Data must be scalable. Clustering algorithm can deal with different types of attributes. Clustering algorithm must be able to find clustered data with the arbitrary shape. This algorithm must be insensitive to noise and outliers. Result obtained must be interpretable and usable. It also deals with the data set of high dimensionalities.

K-means clustering: K-means is a popular unsupervised machine learning algorithm that can be used to identify patterns in data. In anomaly detection, K-means can be used to group similar attacks together based on their characteristics, such as source IP, destination IP, protocol type, and payload.

Hierarchical clustering: Hierarchical clustering is another popular clustering method that can be used to group similar attacks together. This approach involves building a tree-like structure of clusters, where each cluster contains a subset of attacks with similar characteristics.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise): DBSCAN is a clustering method that can be used to group attacks based on their density. This approach can identify clusters of attacks that are more densely packed together, which may indicate a higher likelihood of an attack.

Spectral clustering: Spectral clustering is a graph-based clustering method that can be used to identify patterns in complex data. This approach involves building a similarity matrix of attacks based on their characteristics and using eigenvalues and eigenvectors to identify clusters.

Fuzzy clustering: Fuzzy clustering is a soft clustering method that assigns each attack to one or more clusters with a degree of membership. This approach can be useful when there are overlaps between clusters or when an attack does not clearly belong to a single cluster.

These clustering methods can be combined with other machine learning techniques, such as outlier detection and dimensionality reduction, to improve the accuracy and effectiveness of anomaly detection in honeypots.

1.9 MOTIVATION AND JUSTIFICATION

The use of clustering approaches for anomaly detection in honeypots is encouraged since they enable security staff to swiftly identify possible risks and respond appropriately. Clustering algorithms can assist in identifying patterns of behaviour that differ from the anticipated norm and may signal to the presence of an attack or other anomalous activity by grouping related data points together. In order to protect vital systems and data, it is necessary to utilize more efficient and effective security measures. This justifies the usage of clustering methods for anomaly detection in honeypots. Organizations must be able to promptly identify and eliminate possible threats in light of the number and sophistication of cyberattacks that are becoming more common. The efficiency of honeypot-based security techniques can be increased through the use of clustering approaches.

1.10 PROBLEM STATEMENT

To detect anomalies in honeypot user access data using K-Means and KNN clustering algorithms

1.11OBJECTIVE

To Design a Machine Learning techniques to detect the anomalies in the honeypot by applying K-Means and KNN clustering to detect the anomalies in the given problem.

Honeypot environments are often subject to evolving attack techniques. Clustering methods can aid in adapting to new and emerging threats by dynamically updating the clusters and adjusting the threshold for anomaly detection. This objective involves monitoring the honeypot data continuously and incorporating new patterns or behaviors into the clustering model.

CHAPTER 2

2. SYSTEM CONFIGURATION

2.1 Hardware Requirement

PROCESSOR : Intel I7 And Above

RAM : 8 GB

HARD DISK CAPACITY : 1TB

2.2 Software Requirement

OPERATING SYSTEM : Windows10

PACKAGE : Anaconda

FRONT END : Python

2.3 About the Software

The tools used in this project are listed below.

- Anaconda
- Jupyter Notebook

2.3.1Anaconda

Rather to using command lines, Anaconda Navigator's graphical interface can be usedto install packages, manage environments, and run standard Python tasks. Moreover, itenables easy management of channels, environments, and packages for Conda withoutthe need for command-line input. On the Anaconda Cloud or in a local Anaconda Repository, Navigator can search for packages. Windows, macOS, and Linux are all supported**.**

2.3.2 Jupyter Notebook

An open-source web programmed called the Jupyter Notebook allows users to create andshare documents with live code, equations, visualizations and text. The folks who work on Project Jupyter maintain Jupyter Notebook. The IPython project, which once had its own IPython Notebook project, is where Jupyter Notebooks got their start. Jupiter's name is derived from the three primary programming languages it supports: Julia, Python, and R. There are already more than100 additional kernels available, however Jupyter comes with the IPython kernel, whichenables Python programmed writing.

2.3.3 Python Package

a) Scikit-Learn

Scikit-learn, a free Python package that is frequently seen as a direct extension of SciPy, is based on NumPy and SciPy. It is especially made for creating supervised and unsupervised machine learning algorithms and data modelling.Scikit-learn is user-friendly and beginner-friendly because of its straightforward, intuitive, and consistent interface. Scikit-learn performs admirably by enabling users to alter and exchange data as they need, despite the fact that its utility is constrained becauseit only excels at data modelling.

b) Pandas

Python's Pandas package for data research and analysis enables programmers to createsimple, seamless high-level data structures. Pandas, which is based on NumPy, is in charge ofgetting data sets and data points ready for machine learning. Pandas uses one-dimensional (series) and two-dimensional (Data Frame) data structures. These two types of data structures allow Pandas to be used in a range of industries, from science and statistics to banking and engineering. Due to its adaptability, the Pandas library can be used with other scientific and numerical libraries. Because they are rapid, compliant, and highly descriptive, their data structures are simple to use. By aggregating, integrating, and re-indexing data with Pandas, onecan modify data functionality with a minimum of keystrokes.

c) Matplotlib

Matplotlib is a data visualization library that is used for making plots and graphs. It isan extension of SciPy and is able to handle NumPy data structures as well as complex data models made by Pandas. Although it is expertise is limited to 2D plotting, Matplotlib can produce high- quality and publish-ready diagrams, graphs, plots, histograms, error charts, scatter plots and bar charts. Matplotlib is intuitive and easy to use, making it a great choice for beginners. It is eveneasier to use for people with pre-existing knowledge in various other graph-plotting tools. It offers GUI tool kitsupport, including wxPython, Tkinter, and Qt.

d) Numpy

Open-source and well-known Python library for numbers, NumPy. It is capable of carrying out a wide range of mathematical operations on matrices and arrays. One of the most popular libraries for scientific computing, it is frequently used by scientists to analysedata. It is perfect for machine learning and artificial intelligence (AI) projects since it can process multidimensional arrays, handle linear algebra, and perform Fourier transformation. NumPy arrays demand a considerable reduction in storage space when compared to standard Python lists. They are also lot easier to operate and considerably faster than the earlier.One can reshape,

transpose, and modify data in matrix form with NumPy. Combining NumPy'scapabilities makes it easy to enhance the machine learning model's performance.

e) Seaborn

An open-source Python package for data visualization and graphing is called Seaborn.It uses sophisticated Panda's data structures and is based on the graphing software Matplotlib. Seaborn offers a high-level, feature-rich interface for creating precise, illuminating statistical graphs on its own. Because it can produce logical graphs of learning and execution data, it is employed in machine learning and deep learning applications. The most beautiful and eye-catching graphs and plots are produced by Seaborn, which makes it ideal for use in publishing and marketing. Seaborn can also save you time and effort because it enables you to build complex graphs with little code and basic instructions.

f) train-test split

The train-test split is used to determine how well machine learning algorithms work whenemployed with prediction-based methods and applications. To compare the output of one's ownmachine learning model, one can use this quick and simple procedure.

CHAPTER 3

3. REVIEW OF LITERATURE

SI. NO	**TITLE OF THE PAPER**	**AUTHOR & YEAR OF PUBLISH**	**DATASET USED**	**ALGORITHM APPLIED**	**RESULT&ACCURACY**
1	IP Network Anomaly Detection using Machine Learning	Roshan Nair TechBU Nextgen R&D TCS Hyderabad Chaithanya Pramodh Kasula TechBU Nextgen R&D TCS Hyderabad 2019	SNMP Dataset	Gaussian Probability Distribution and K-means	93.83% Gaussian 80% K-means
2	An Outlier Ensemble for Unsupervised Anomaly Detection in Honeypot Data	Lynda Boukela , Gongxuan Zhang, Saima Bouzefrane and Junlong Zhou 2019	Kyoto 2006+ dataset	outlier ensemble with LOF (Local Outlier Factor	92%
3	Machine Learning for Anomaly Detection: A Systematic Review	ALI BOU NASSIF 2021	NSL-KDD dataset	Decision Forest LOF	99.9%
4	A novel honeypot based security approach for real-time intrusion	Muhammet Baykara, Resul Das Department of Software Engineering, Technology Faculty, Fırat	DARPA dataset	IDS	reducing the false positive rate in IDSs

	detection and prevention systems	University, Elazıg˘ 23119, Turkey 2018			
5	A Comparative Study of Unsupervised Anomaly Detection Techniques Using Honeypot Data	Jungsuk SONG, Hiroki TAKAKURA 2019	KDD Cup 99	LOF, DBScan,K-Means,KNN,OCSVM	helps in reducing the false positive level in IDSs
6	Hybrid System Between Anomaly Based Detection System and Honeypot to Detect Zero Day Attack	Nisreen Innab Department of Information Security Naif Arab University for Security Science Riyadh, Saudi Arabia nisreen.innab@nauss.edu.sa Eman Alomairy Department of Information Security Naif Arab University for Security Science Riyadh, Saudi Arabia eman.omairy@nauss.edu.sa Lamya Alsheddi Department of Information Security Naif Arab University for Security Science Riyadh, Saudi Arabia 4370885@nauss.edu.sa 2018	SWORD	CSS	to minimize false positive alarms and latency

7	Combining Naive Bayes and Decision Tree for Adaptive Intrusion Detection	Dewan Md. Farid ,Nouria Harbi 1, University Lumière Lyon 2 - France dewanfarid@gmail.com, nouria.harbi@univ-lyon2.fr Mohammad Zahidur Rahman Jahangirnagar University · Department of Computer Science and Engineering **2010**	KDD99 benchmark dataset	Decision tree and naïve Bayes for adaptive IDS	99%
8	A Hybrid Anomaly Detection Framework in Cloud Computing Using One-Class and Two-Class Support Vector Machines	Song Fu1, Jianguo Liu2, and Husanbir Pannu2 1 Department of Computer Science and Engineering, University of North Texas, Denton, Department of Mathematics, University of North Texas, Denton, TX 76203, USA Song. Fu, jgliu@unt.edu HusanbirPannu@my.unt.edu 2012	Cloud-specific datasets	One class and two class SVM	92.1% detection sensitivity and 83.8% detection specificity
9	Anomaly Based Intrusion Detection Using Hybrid Learning Approach of Combining k-	Roshan Chitrakar Nepal College of Information	Kyoto 2006+ dataset.	Naïve Bayes and k- medoids	96.08% K-medoids 96.55% Naïve Bayes

	Medoids Clustering and Naïve Bayes Classification	Technology · Computer Science PhD in Information Security *Managing all academic projects of Computer related programmes of NCIT.* Chuanhe Huang Wuhan University | WHU · College of Computer Science 2012			
10	Decision Tree based Support Vector Machine for Intrusion Detection	Mrs. Snehal A. Mulay Department of Information Technology, Bharati Vidyapith's CO E, Pune, India snehalmulay@gmail.com Prof. P. R. Devale HOD, Department of Information Technology Bharati Vidyapith's CO E, Pune, India prdevale@bvucoep.edu.in	KDDCU P'99 data set	SVM and Decision Tree	95%

		Prof. G.V. Garje HOD, Department of Computer and IT PVG's CO ET, Pune, India garjegv@yahoo.com 2012			
11	A novel unsupervised classification approach for network anomaly detection by k-Means clustering and ID3 decision tree learning methods	Yasser Yasami · Saadat Pour Mozaffari 2009	ARP traffic data set	K-means, ID3	96% K-means, 98% ID3
12	Modeling intrusion detection system using hybrid intelligent systems	Sandhya Peddabachigaria, Ajith Abrahamb, Crina Grosanc, Johnson Thomas Sandhya Peddabachigaria, Ajith Abrahamb, Crina Grosanc, Johnson Thomas 2005	KDD cup 99	Decision trees, support vector machines, Ensemble approach	100%
13	Anomaly Detection using Support Vector Machine Classification with k-Medoids Clustering	Roshan Chitrakar and Huang Chuanhe School of Computer, Wuhan University Wuhan, Hubei, China E-mail:	KDD cup 99, Kyoto 2006+ dataset	k- medoids, SVM, Naïve Bayes	84.82% K-medoids, 88.30% NB, 99.21% SVM

		roshanchi, huangch@whu.edu.cn 2012			

Roshan Nair, Chaithanya Pramodh Kasula 2019 [1] have proposed IP Network Anomaly Detection using Machine Learning Gaussian Probability Distribution and K-means and the highest accuracy obtained by 93.83% Gaussian Probability Distribution.

Lynda Boukela, Gongxuan Zhang, Saima Bouzefrane and Junlong Zhou 2019 [2] have proposed An Outlier Ensemble for Unsupervised Anomaly Detection in Honeypot Data using outlier ensemble with LOF (Local Outlier Factor and the highest accuracy obtained by 92%.

ALI BOU NASSIF 2021 [3] have proposed Machine Learning for Anomaly Detection: A Systematic Review

using Decision Forest, LOF and the highest accuracy obtained by 99.9% Decision Forest.

Muhammet Baykara, Resul Das 2018 [4] have proposed A novel honeypot-based security approach for real-time intrusion detection and prevention systems using Intrusion Detection System and reducing the false positive rate in IDSs

Jungsuk SONG, Hiroki TAKAKURA 2019 [5] have proposed A Comparative Study of Unsupervised Anomaly Detection Techniques Using LOF, DBScan, K-Means,KNN,OCSVM helps to reducing the false positive level in IDSs.

Nisreen Innab, Eman Alomairy 2018 [6] have proposed Hybrid System Between Anomaly Based Detection System and Honeypot to Detect Zero Day Attack Using CSS use to minimize false positive alarms and latency.

Dewan Md. Farid, Nouria Harbi, Mohammad Zahidur Rahman 2010 [7] have proposed Combining Naive Bayes and Decision Tree for Adaptive Intrusion Detection Using Decision tree and naïve Bayes for adaptive IDS and the highest accuracy obtained by 99%.

Song Fu1, Jianguo Liu2, and Husanbir Pannu 2012 [8] have proposed A Hybrid Anomaly Detection Framework in Cloud Computing Using One-Class and Two-Class Support Vector Machines Using One class and two class Support Vector System and the accuracy obtained by 92.1% detection sensitivity and 83.8% detection specificity.

Roshan Chitrakar 2012 [9] have proposed Anomaly Based Intrusion Detection Using Hybrid Learning Approach of Combining k-Medoids Clustering and Naïve Bayes Classification Using Naïve Bayes and k-medoids and the highest accuracy obtained by 96.55% Naïve Bayes.

Mrs. Snehal A. Mulay, Prof. P. R. Devale, Prof. G.V. Garje 2012 [10] have proposed Decision Tree based Support Vector Machine for Intrusion Detection Using SVM and Decision Tree and the highest accuracy obtained by 95%.

Yasser Yasami · Saadat Pour Mozaffari 2009 [11] have proposed A novel unsupervised classification approach for network anomaly detection by k-Means clustering and ID3 decision tree learning methods Using K-means, ID3and the highest accuracy obtained by 98% ID3.

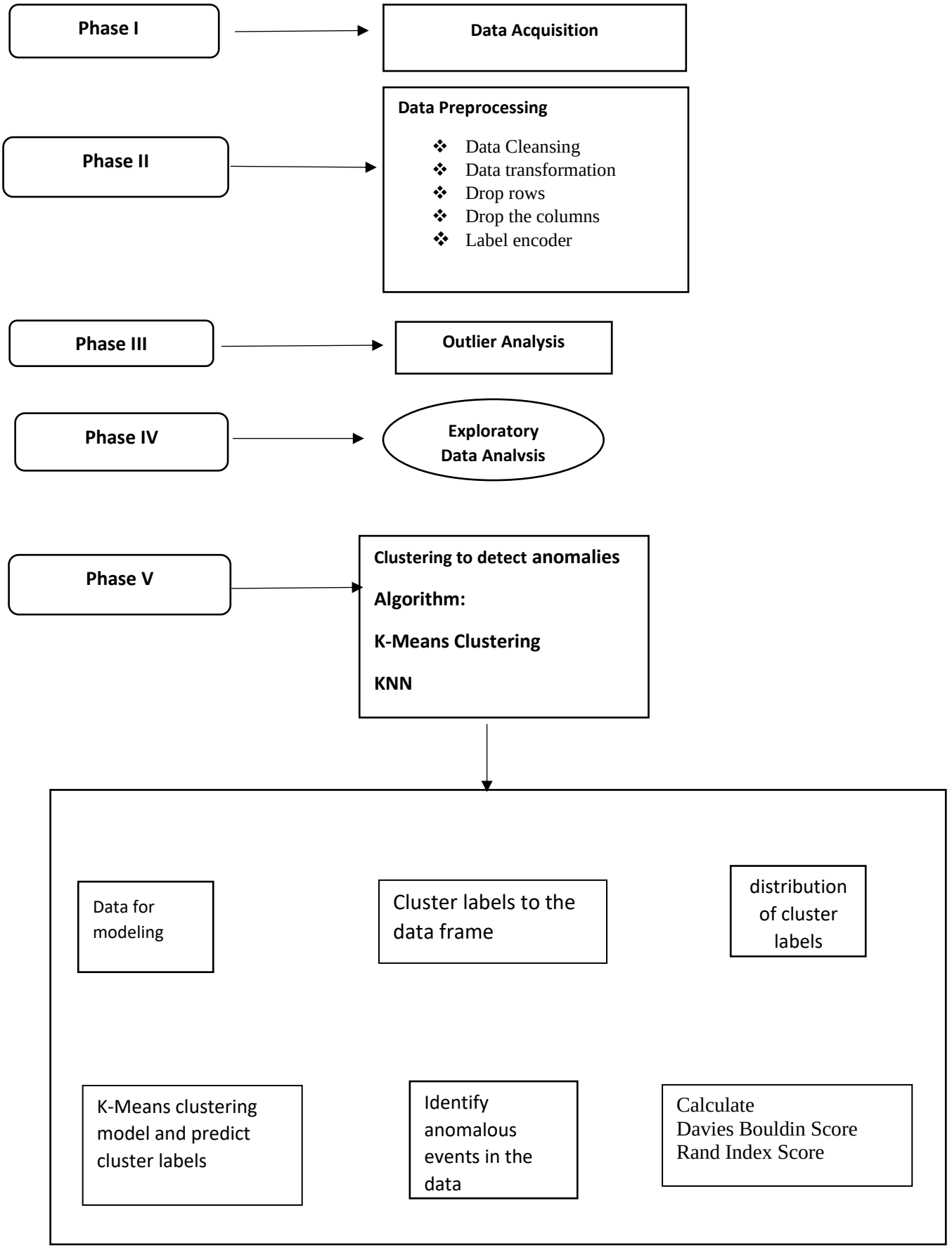
Sandhya Peddabachigaria, Ajith Abrahamb, Crina Grosanc, Johnson Thomas 2005 [12] have proposed Modeling intrusion detection system using hybrid intelligent systems Using Decision trees, support vector machines, Ensemble approach and the highest accuracy obtained by 100%.

Roshan Chitrakar and Huang Chuanhe 2012 [13] have proposed Anomaly Detection using Support Vector Machine Classification with k-Medoids Clustering Using k- medoids, SVM, Naïve Bayes and the highest accuracy obtained by 99.21% SVM.

CHAPTER 4

4. METHODOLOGY

The choice of clustering algorithm and the specific parameters used in the clustering process can vary depending on the specific application and the nature of the data being analyzed. Additionally, multiple clustering algorithms may be used in combination to improve the accuracy of the detection. Overall, the methodology for anomaly detection in honeypot using clustering methods involves collecting and preprocessing data, applying clustering algorithms to the data, detecting anomalies, validating the anomalies, and reporting the results. Figure 4.1 shows the specific tasks performed during each stage.


Phase I
Data Acquisition
Data Preprocessing
Data Cleansing
Data transformation
Drop rows
Drop the columns
Label encoder
Phase II
Phase III
Outlier Analysis
Phase IV
Exploratory
Data Analvsis
Phase V
Clustering to detect anomalies
Algorithm:
K-Means Clustering
KNN
Data for modeling
Cluster labels to the data frame
distribution of cluster labels
K-Means clustering model and predict cluster labels
Identify anomalous events in the data
Calculate
Davies Bouldin Score
Rand Index Score

PHASE I – DATA ACQUISITION

4.1 Data Acquisition

The process of collecting data include compiling datasets from relevant sources in order to evaluate and test the suggested system. This project deals with the Honeypot data from sensors Dionaea installed at the Information Lab of the University of Muhammadiyah Malang, from January 2019 to December 2019, there were around 6.000,000 more attacks, the average attack reaching 500 to 700 attacks every day. This data is only in the form of attack data (Date, Sensor, Country, Src, IP, etc., port, Protocol, Honeypot Sensor).

Dataset Description

Dataset Name : Mendeley Honeypot Dataset

Number of Instances: 5598567

Number of Features: 7

Dataset Link: https://data.mendeley.com/datasets/6fc7my86t4/1

PHASE II – DATA PREPROCESSING

4.2 Data Preprocessing

Data pre-processing is a vital phase in machine learning that improves the quality of the data to promote the extraction of valuable insights from the data. Preparing (cleaning and arranging) raw data in orderto make it acceptable for creating and training Machine Learning Model.It raises reliability and accuracy. Pre-processing data can increase the correctness and quality of a dataset, making it more stable by removing missing or inconsistent data values brought on by human or computer mistake. It ensures consistency in data.

4.2.1 Data Pre- Processing Techniques Implemented In the Datasets

a) Handling Null Values

- There are almost never any null values in a real-world dataset. No model can handle these NULL or NaN values on its own, thus we must step in regardless of whether the issue is one ofregression, classification, or any other kind. Python represents NULL with NaN. So, don't mixthe two as they can be used alternately.
- We can approach this issue in a number of different ways. Dropping the null-valued rows or columns is the simplest solution to this issue.

- If the missing values are not handled correctly, you can wind up creating a machine learning model that is biased and produces inaccurate results. Missing data can make the statistical analysis less precise.

b) Removing Duplicate Values

- A dataset contains many instances of a duplicate value. It is frequently discovered when usingExcel to work with huge databases.
- Data processing will be unsuccessful if duplicate records are not eliminated. The goal of this control is to eliminate multiple records from the dataset in order to make it ready for further processing.

c) Dropping NAN Values

- The numeric data type NaN (Not a Number) refers to undefined values or values that cannot berepresented, particularly the outcomes of floating-point calculations.
- Your machine learning model will function better if you clean the data. Hence, processing databefore usage is crucial.

d) Label Encoding

- Label encoding is the process of transforming labels into a numeric form so that they may beread by machines.
- Machine learning methods can then be used to determine how well those labels arefunctioning.
- It is a crucial stage in the supervised learning pre-processing of the structured dataset.

PHASE III – OUTLIER ANALYSIS

4.3 Outlier Analysis

A data analysis approach called outlier analysis is used to find and examine data points that differ considerably from the rest of the data. Data points known as outliers differ from the typical pattern or behaviour of the data; they may be data errors, abnormalities, or significant insights.

The following steps are involved in outlier analysis:

- Data visualization: The first step is to use graphs or charts to visualize the data. This makes it easier to spot any data points that are spread apart from the others.
- Statistical analysis: To find outliers, statistical methods are applied. These methods consist of quartile range analysis, and z-score analysis.
- Machine learning algorithms: Outliers can also be found using machine learning techniques like support vector machines, decision trees, and clustering.
- Cleansing the data: After outliers have been located, the next step is to get rid of or fix them. Investigating the origin of the outliers is necessary to establish if they represent data errors or significant insights.

PHASE IV EXPLORATORY DATA ANALYSIS

Exploratory data analysis (EDA) is the process of examining and comprehending data sets in order to highlight the key features and identify trends and relationships between the variables. Gaining understanding of the data and creating a hypothesis that can be tested later are the main objectives of EDA.

The following specific statistical approaches and operations are possible with EDA tools:

Techniques like clustering and dimension reduction assist in producing graphical representations of highly dimensional data with several variables.

- **Summary statistics** are shown along with a univariate visualisation of each field in the raw dataset.
- **K-means Unsupervised learning** uses the clustering technique known as clustering, in which data points are divided into K groups, or the number of clusters, according to how far they are from the centroid of each group. The data points that fall into the same category are those that are closest to a certain centroid. K-means Market segmentation, pattern identification, and image compression all frequently use clustering.
- **Data visualisation** is the process of representing data in an understandable manner using graphs, charts, and other visual aids. Histograms, scatter plots, and box plots are a few examples.
- **Calculating descriptive statistics** like mean, median, mode, standard deviation, and variance will help you summarise the data and spot any odd trends or outliers.
- **Using statistical tests** like t-tests, ANOVA, and chi-square tests, hypothesis testing entails evaluating the data's assumptions and hypotheses.
- **Dimensionality reduction** includes lowering the number of dimensions in the data by determining which factors are most crucial in causing its variability.
- **Clustering and classification** require giving labels to the data points based on their properties and putting similar data points together.

PHASE V Detect Anomalies and their performance

In cybersecurity, it's crucial to find data abnormalities. The effectiveness of anomaly detection approaches depends on a number of variables, including the type of data being utilized, the difficulty of the challenge, and the particular algorithm being employed. The following metrics are frequently used to assess the effectiveness of anomaly detection techniques:

- **Detection rate:** This is the proportion of real abnormalities that the algorithm accurately identifies.
- **False positive rate:** This is the proportion of times that regular data points are mistakenly classified as anomalies.
- **Precision:** This is the proportion of abnormalities that are genuinely real that are discovered.
- **Recall:** This is the proportion of real anomalies that the algorithm properly classified.
- **F1-score:** This is a widely used metric for assessing the overall effectiveness of an anomaly detection algorithm. It is the harmonic mean of precision and recall.
- The choice of anomaly detection algorithm can significantly impact the performance of the technique. Some of the commonly used anomaly detection algorithms include:
- **Statistical procedures:** To find anomalies, these techniques employ statistical techniques like z-score analysis, quartile range analysis.
- **Algorithms for machine learning:** These algorithms look for abnormalities using methods like clustering, decision trees, and support vector machines.
- **Deep learning techniques:** These techniques make use of neural networks to recognize patterns in data and identify anomalies.
- **Ensemble techniques:** These techniques combine several algorithms for anomaly detection to enhance overall performance.

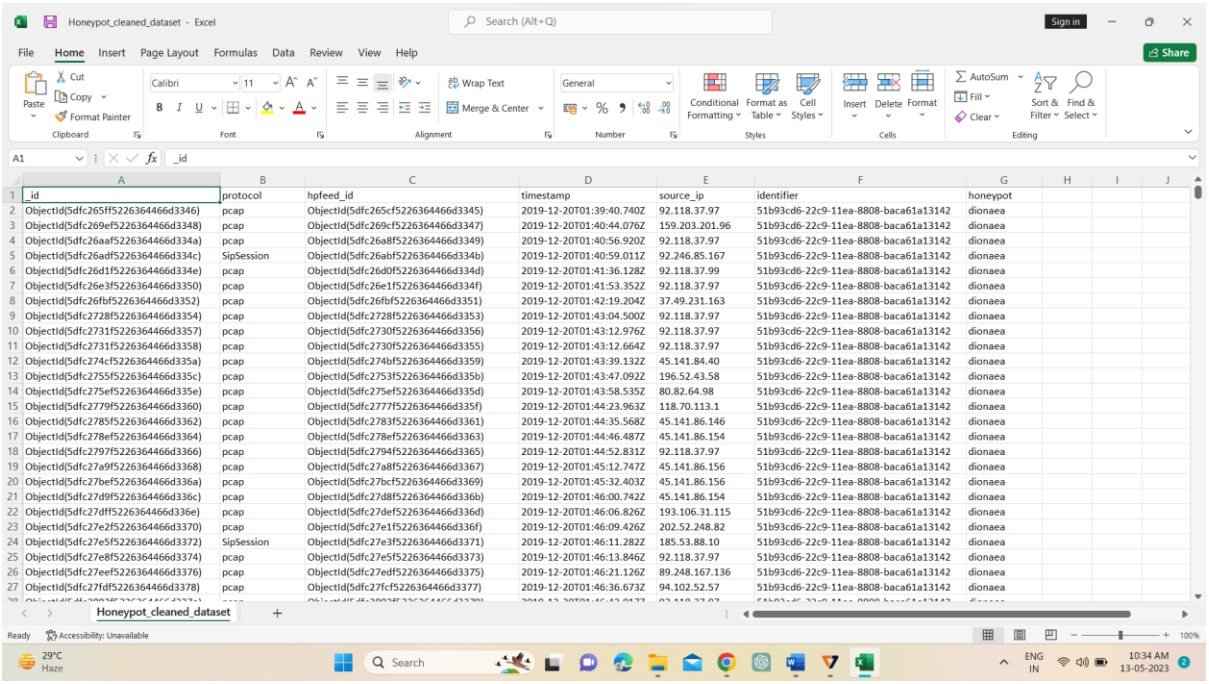
In conclusion, finding anomalies in data is a crucial activity that may be carried out using a variety of methods. Metrics like detection rate, false positive rate, precision, recall, and F1-score can be used to gauge how well these techniques' function. The algorithm used and the kind of anomalies found in the data can have a big impact on how well the technique performs.

CHAPTER 5

5. RESULTS AND DISCUSSION

5.1 Data Collection

The Honeypot data from sensors Dionaea installed at the Information Lab of the University of Muhammadiyah Malang, from January 2019 to December 2019, there were around 6.000,000 more attacks, the average attack reaching 500 to 700 attacks every day. This data is only in the form of attack data (Date, Sensor, Country, Src, IP, etc., port, Protocol, Honeypot Sensor).



_id	protocol	hpfeed_id	timestamp	source_ip	identifier	honeypot
ObjectId(5dfc265ff5226364466d3346)	pcap	ObjectId(5dfc265cf5226364466d3345)	2019-12-20T01:39:40.740Z	92.118.37.97	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc269ef5226364466d3348)	pcap	ObjectId(5dfc269cf5226364466d3347)	2019-12-20T01:40:44.076Z	159.203.201.96	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc26aaf5226364466d334a)	pcap	ObjectId(5dfc26a8f5226364466d3349)	2019-12-20T01:40:56.920Z	92.118.37.97	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc26adf5226364466d334c)	SipSession	ObjectId(5dfc26abf5226364466d334b)	2019-12-20T01:40:59.011Z	92.246.85.167	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc26d1f5226364466d334e)	pcap	ObjectId(5dfc26d0f5226364466d334d)	2019-12-20T01:41:36.128Z	92.118.37.99	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc26e3f5226364466d3350)	pcap	ObjectId(5dfc26e1f5226364466d334f)	2019-12-20T01:41:53.352Z	92.118.37.97	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc26fbf5226364466d3352)	pcap	ObjectId(5dfc26fbf5226364466d3351)	2019-12-20T01:42:19.204Z	37.49.231.163	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc2728f5226364466d3354)	pcap	ObjectId(5dfc2728f5226364466d3353)	2019-12-20T01:43:04.500Z	92.118.37.97	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc2731f5226364466d3357)	pcap	ObjectId(5dfc2730f5226364466d3356)	2019-12-20T01:43:12.976Z	92.118.37.97	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc2731f5226364466d3358)	pcap	ObjectId(5dfc2730f5226364466d3355)	2019-12-20T01:43:12.664Z	92.118.37.97	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc274cf5226364466d335a)	pcap	ObjectId(5dfc274bf5226364466d3359)	2019-12-20T01:43:39.132Z	45.141.84.40	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc2755f5226364466d335c)	pcap	ObjectId(5dfc2753f5226364466d335b)	2019-12-20T01:43:47.092Z	196.52.43.58	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc275ef5226364466d335e)	pcap	ObjectId(5dfc275ef5226364466d335d)	2019-12-20T01:43:58.535Z	80.82.64.98	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc2779f5226364466d3360)	pcap	ObjectId(5dfc2777f5226364466d335f)	2019-12-20T01:44:23.963Z	118.70.113.1	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc2785f5226364466d3362)	pcap	ObjectId(5dfc2783f5226364466d3361)	2019-12-20T01:44:35.568Z	45.141.86.146	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc278ef5226364466d3364)	pcap	ObjectId(5dfc278ef5226364466d3363)	2019-12-20T01:44:46.487Z	45.141.86.154	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc2797f5226364466d3366)	pcap	ObjectId(5dfc2794f5226364466d3365)	2019-12-20T01:44:52.831Z	92.118.37.97	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc27a9f5226364466d3368)	pcap	ObjectId(5dfc27a8f5226364466d3367)	2019-12-20T01:45:12.747Z	45.141.86.156	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc27bef5226364466d336a)	pcap	ObjectId(5dfc27bcf5226364466d3369)	2019-12-20T01:45:32.403Z	45.141.86.156	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc27d9f5226364466d336c)	pcap	ObjectId(5dfc27d8f5226364466d336b)	2019-12-20T01:46:00.742Z	45.141.86.154	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc27dff5226364466d336e)	pcap	ObjectId(5dfc27def5226364466d336d)	2019-12-20T01:46:06.826Z	193.106.31.115	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc27e2f5226364466d3370)	pcap	ObjectId(5dfc27e1f5226364466d336f)	2019-12-20T01:46:09.426Z	202.52.248.82	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc27e5f5226364466d3372)	SipSession	ObjectId(5dfc27e3f5226364466d3371)	2019-12-20T01:46:11.282Z	185.53.88.10	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc27e8f5226364466d3374)	pcap	ObjectId(5dfc27e5f5226364466d3373)	2019-12-20T01:46:13.846Z	92.118.37.97	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc27eef5226364466d3376)	pcap	ObjectId(5dfc27edf5226364466d3375)	2019-12-20T01:46:21.126Z	89.248.167.136	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
ObjectId(5dfc27fdf5226364466d3378)	pcap	ObjectId(5dfc27fcf5226364466d3377)	2019-12-20T01:46:36.673Z	94.102.52.57	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea

Figure 5.1 Mendeley Honeypot Dataset

5.2 Data Pre-Processing in Honeypot Dataset

5.2.1Removing Duplicate values in Dataset

A dataset contains many instances of a duplicate value. It is frequently discovered when using Excel to work with huge databases. Data processing will be unsuccessful if duplicate records are not eliminated. The goal of this control is to eliminate multiple records from the dataset in order to make it ready for further processing.

```
# Drop duplicates
df = df.drop_duplicates()
print(df)
```

```
                                        _id      protocol  \
2237   ObjectId(5dfc2ceef5226364466d44c3)  microsoft-ds
2635   ObjectId(5dfc2d51f5226364466d47dd)  microsoft-ds
2904   ObjectId(5dfc2d96f5226364466d49fe)  microsoft-ds
3596   ObjectId(5dfc2e4df5226364466d4f61)  microsoft-ds
8984   ObjectId(5dfc3403f5226364466d7986)  microsoft-ds
9048   ObjectId(5dfc3412f5226364466d79fb)  microsoft-ds
9294   ObjectId(5dfc3458f5226364466d7bf0)  microsoft-ds
13830  ObjectId(5dfc3948f5226364466d9f58)  microsoft-ds
15259  ObjectId(5dfc3ad4f5226364466daa82)  microsoft-ds
23261  ObjectId(5dfc4223f5226364466de90c)  microsoft-ds
23600  ObjectId(5dfc4262f5226364466debbb)  microsoft-ds
23755  ObjectId(5dfc427df5226364466decf0)  microsoft-ds
24210  ObjectId(5dfc42d4f5226364466df076)  microsoft-ds
25831  ObjectId(5dfc4412f5226364466dfd20)  microsoft-ds
25886  ObjectId(5dfc441bf5226364466dfd88)  microsoft-ds
28682  ObjectId(5dfc4638f5226364466e135f)  microsoft-ds
29042  ObjectId(5dfc467df5226364466e1630)  microsoft-ds
29100  ObjectId(5dfc468ff5226364466e16b1)  microsoft-ds
```

Figure 5.2.1 Dropping Duplicates

5.2.2. Dropping NAN Values

The numeric data type NaN (Not a Number) refers to undefined values or values that cannot be represented, particularly the outcomes of floating-point calculations.

```
df.isnull()
print(df)
```

```
                                identifier honeypot
2237   51b93cd6-22c9-11ea-8808-baca61a13142  dionaea
2635   51b93cd6-22c9-11ea-8808-baca61a13142  dionaea
2904   51b93cd6-22c9-11ea-8808-baca61a13142  dionaea
3596   51b93cd6-22c9-11ea-8808-baca61a13142  dionaea
8984   51b93cd6-22c9-11ea-8808-baca61a13142  dionaea
9048   51b93cd6-22c9-11ea-8808-baca61a13142  dionaea
9294   51b93cd6-22c9-11ea-8808-baca61a13142  dionaea
13830  51b93cd6-22c9-11ea-8808-baca61a13142  dionaea
15259  51b93cd6-22c9-11ea-8808-baca61a13142  dionaea
23261  51b93cd6-22c9-11ea-8808-baca61a13142  dionaea
23600  51b93cd6-22c9-11ea-8808-baca61a13142  dionaea
23755  51b93cd6-22c9-11ea-8808-baca61a13142  dionaea
24210  51b93cd6-22c9-11ea-8808-baca61a13142  dionaea
25831  51b93cd6-22c9-11ea-8808-baca61a13142  dionaea
25886  51b93cd6-22c9-11ea-8808-baca61a13142  dionaea
28682  51b93cd6-22c9-11ea-8808-baca61a13142  dionaea
29042  51b93cd6-22c9-11ea-8808-baca61a13142  dionaea
29100  51b93cd6-22c9-11ea-8808-baca61a13142  dionaea
```

Figure 5.2.2 Dropping Null Values

5.2.3 Label Encoding

Label encoding is the process of transforming labels into a numeric form so that they may be read by machines. Machine learning methods can then be used to determine how well those labels are functioning. Itis a crucial stage in the supervised learning pre-processing of the structured dataset.

```
# Encode labels in column 'honeypot'.
df['honeypot']= label_encoder.fit_transform(df['honeypot'])
print(df.head())
```

```
                                  _id     protocol  \
0  ObjectId(5dfc265ff5226364466d3346)         pcap
1  ObjectId(5dfc269ef5226364466d3348)         pcap
2  ObjectId(5dfc26aaf5226364466d334a)         pcap
3  ObjectId(5dfc26adf5226364466d334c)   SipSession
4  ObjectId(5dfc26d1f5226364466d334e)         pcap

                           hpfeed_id                            timestamp   \
0  ObjectId(5dfc265cf5226364466d3345) 2019-12-20 01:39:40.740000+00:00
1  ObjectId(5dfc269cf5226364466d3347) 2019-12-20 01:40:44.076000+00:00
2  ObjectId(5dfc26a8f5226364466d3349) 2019-12-20 01:40:56.920000+00:00
3  ObjectId(5dfc26abf5226364466d334b) 2019-12-20 01:40:59.011000+00:00
4  ObjectId(5dfc26d0f5226364466d334d) 2019-12-20 01:41:36.128000+00:00

         source_ip                                identifier  honeypot  hour  \
0     92.118.37.97  51b93cd6-22c9-11ea-8808-baca61a13142          1     1
1  159.203.201.96  51b93cd6-22c9-11ea-8808-baca61a13142          1     1
2     92.118.37.97  51b93cd6-22c9-11ea-8808-baca61a13142          1     1
3    92.246.85.167  51b93cd6-22c9-11ea-8808-baca61a13142          1     1
4     92.118.37.99  51b93cd6-22c9-11ea-8808-baca61a13142          1     1

   day_of_week  cluster
0            4        0
1            4        0
2            4        0
3            4        0
4            4        0
```

Figure 5.2.3 Label Encoding for Honeypot

```
# Encode labels in column 'protocol'.
df['protocol']= label_encoder.fit_transform(df['protocol'])
print(df.head())
```

```
                                _id  protocol  \
0  ObjectId(5dfc265ff5226364466d3346)        10
1  ObjectId(5dfc269ef5226364466d3348)        10
2  ObjectId(5dfc26aaf5226364466d334a)        10
3  ObjectId(5dfc26adf5226364466d334c)         2
4  ObjectId(5dfc26d1f5226364466d334e)        10

                          hpfeed_id                        timestamp  \
0  ObjectId(5dfc265cf5226364466d3345) 2019-12-20 01:39:40.740000+00:00
1  ObjectId(5dfc269cf5226364466d3347) 2019-12-20 01:40:44.076000+00:00
2  ObjectId(5dfc26a8f5226364466d3349) 2019-12-20 01:40:56.920000+00:00
3  ObjectId(5dfc26abf5226364466d334b) 2019-12-20 01:40:59.011000+00:00
4  ObjectId(5dfc26d0f5226364466d334d) 2019-12-20 01:41:36.128000+00:00

        source_ip                          identifier  honeypot  hour  \
0    92.118.37.97  51b93cd6-22c9-11ea-8808-baca61a13142         1     1
1  159.203.201.96  51b93cd6-22c9-11ea-8808-baca61a13142         1     1
2    92.118.37.97  51b93cd6-22c9-11ea-8808-baca61a13142         1     1
3   92.246.85.167  51b93cd6-22c9-11ea-8808-baca61a13142         1     1
4    92.118.37.99  51b93cd6-22c9-11ea-8808-baca61a13142         1     1

   day_of_week  cluster
0            4        0
1            4        0
2            4        0
3            4        0
4            4        0
```

Figure 5.2.4 Label Encoding for Protocol

```
df['_id']= label_encoder.fit_transform(df['_id'])
print(df.head())
```

```
   _id  protocol                          hpfeed_id  \
0    0        10  ObjectId(5dfc265cf5226364466d3345)
1    1        10  ObjectId(5dfc269cf5226364466d3347)
2    2        10  ObjectId(5dfc26a8f5226364466d3349)
3    3         2  ObjectId(5dfc26abf5226364466d334b)
4    4        10  ObjectId(5dfc26d0f5226364466d334d)

                        timestamp       source_ip  \
0 2019-12-20 01:39:40.740000+00:00    92.118.37.97
1 2019-12-20 01:40:44.076000+00:00  159.203.201.96
2 2019-12-20 01:40:56.920000+00:00    92.118.37.97
3 2019-12-20 01:40:59.011000+00:00   92.246.85.167
4 2019-12-20 01:41:36.128000+00:00    92.118.37.99

                          identifier  honeypot  hour  day_of_week  cluster
0  51b93cd6-22c9-11ea-8808-baca61a13142         1     1            4        0
1  51b93cd6-22c9-11ea-8808-baca61a13142         1     1            4        0
2  51b93cd6-22c9-11ea-8808-baca61a13142         1     1            4        0
3  51b93cd6-22c9-11ea-8808-baca61a13142         1     1            4        0
4  51b93cd6-22c9-11ea-8808-baca61a13142         1     1            4        0
```

Figure 5.2.5 Label Encoding for Id

```
df['hpfeed_id']= label_encoder.fit_transform(df['hpfeed_id'])
print(df.head())
```

```
   _id  protocol  hpfeed_id                       timestamp       source_ip  \
0    0        10          0 2019-12-20 01:39:40.740000+00:00    92.118.37.97
1    1        10          1 2019-12-20 01:40:44.076000+00:00  159.203.201.96
2    2        10          2 2019-12-20 01:40:56.920000+00:00    92.118.37.97
3    3         2          3 2019-12-20 01:40:59.011000+00:00   92.246.85.167
4    4        10          4 2019-12-20 01:41:36.128000+00:00    92.118.37.99

                             identifier  honeypot  hour  day_of_week  cluster
0  51b93cd6-22c9-11ea-8808-baca61a13142         1     1            4        0
1  51b93cd6-22c9-11ea-8808-baca61a13142         1     1            4        0
2  51b93cd6-22c9-11ea-8808-baca61a13142         1     1            4        0
3  51b93cd6-22c9-11ea-8808-baca61a13142         1     1            4        0
4  51b93cd6-22c9-11ea-8808-baca61a13142         1     1            4        0
```

Figure 5.2.6 Label Encoding for Hpfeed Id

```
df['timestamp']= label_encoder.fit_transform(df['timestamp'])
print(df.head())
```

```
   _id  protocol  hpfeed_id  timestamp       source_ip  \
0    0        10          0          0    92.118.37.97
1    1        10          1          1  159.203.201.96
2    2        10          2          2    92.118.37.97
3    3         2          3          3   92.246.85.167
4    4        10          4          4    92.118.37.99

                             identifier  honeypot  hour  day_of_week  cluster
0  51b93cd6-22c9-11ea-8808-baca61a13142         1     1            4        0
1  51b93cd6-22c9-11ea-8808-baca61a13142         1     1            4        0
2  51b93cd6-22c9-11ea-8808-baca61a13142         1     1            4        0
3  51b93cd6-22c9-11ea-8808-baca61a13142         1     1            4        0
4  51b93cd6-22c9-11ea-8808-baca61a13142         1     1            4        0
```

Figure 5.2.7 Label Encoding for Timestamp

```
# Encode labels in column 'source_ip'.
df['source_ip']= label_encoder.fit_transform(df['source_ip'])
print(df.head())
```

```
   _id  protocol  hpfeed_id  timestamp  source_ip  \
0    0        10          0          0       1126
1    1        10          1          1        293
2    2        10          2          2       1126
3    3         2          3          3       1131
4    4        10          4          4       1127

                             identifier  honeypot  hour  day_of_week  cluster
0  51b93cd6-22c9-11ea-8808-baca61a13142         1     1            4        0
1  51b93cd6-22c9-11ea-8808-baca61a13142         1     1            4        0
2  51b93cd6-22c9-11ea-8808-baca61a13142         1     1            4        0
3  51b93cd6-22c9-11ea-8808-baca61a13142         1     1            4        0
4  51b93cd6-22c9-11ea-8808-baca61a13142         1     1            4        0
```

Figure 5.2.8 Label Encoding for Source Ip

```
df['identifier']= label_encoder.fit_transform(df['identifier'])
print(df.head())
```

```
   _id  protocol  hpfeed_id  timestamp  source_ip  identifier  honeypot  hour  \
0    0        10          0          0       1126           0         1     1
1    1        10          1          1        293           0         1     1
2    2        10          2          2       1126           0         1     1
3    3         2          3          3       1131           0         1     1
4    4        10          4          4       1127           0         1     1

   day_of_week  cluster
0            4        0
1            4        0
2            4        0
3            4        0
4            4        0
```

Figure 5.2.9 Label Encoding for Identifier

5.3 Outlier Analysis

A data analysis approach called outlier analysis is used to find and examine data points that differ considerably from the rest of the data. Data points known as outliers differ from the typical pattern or behaviour of the data; they may be data errors, abnormalities, or significant insights.

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_2664\2386164608.py:2: FutureWarning: The default value of numeric_only in DataFram
e.quantile is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of nume
ric_only to silence this warning.
  Q1 = df.quantile(0.25)
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_2664\2386164608.py:3: FutureWarning: The default value of numeric_only in DataFram
e.quantile is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of nume
ric_only to silence this warning.
  Q3 = df.quantile(0.75)
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_2664\2386164608.py:5: FutureWarning: Automatic reindexing on DataFrame vs Series co
mparisons is deprecated and will raise ValueError in a future version. Do `left, right = left.align(right, axis=1, copy=False)`
before e.g. `left == right`
  df = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
```

```
Outlier Analysis
                                        _id    protocol  \
0        ObjectId(5dfc265ff5226364466d3346)        pcap
1        ObjectId(5dfc269ef5226364466d3348)        pcap
2        ObjectId(5dfc26aaf5226364466d334a)        pcap
3        ObjectId(5dfc26adf5226364466d334c)  SipSession
4        ObjectId(5dfc26d1f5226364466d334e)        pcap
...                                     ...         ...
1048567  ObjectId(5dfd8738f52263448f85e94a)      mssqld
1048568  ObjectId(5dfd8738f52263448f85e94b)      mssqld
1048569  ObjectId(5dfd8738f52263448f85e94c)      mssqld
1048570  ObjectId(5dfd8738f52263448f85e94d)      mssqld
1048571  ObjectId(5dfd8738f52263448f85e94e)      mssqld

                                  hpfeed_id                 timestamp  \
0        ObjectId(5dfc265cf5226364466d3345)  2019-12-20T01:39:40.740Z
1        ObjectId(5dfc269cf5226364466d3347)  2019-12-20T01:40:44.076Z
2        ObjectId(5dfc26a8f5226364466d3349)  2019-12-20T01:40:56.920Z
3        ObjectId(5dfc26abf5226364466d334b)  2019-12-20T01:40:59.011Z
4        ObjectId(5dfc26d0f5226364466d334d)  2019-12-20T01:41:36.128Z
...                                     ...                       ...
1048567  ObjectId(5dfd8738f52263448f85e947)  2019-12-21T02:45:12.631Z
1048568  ObjectId(5dfd8738f52263448f85e946)  2019-12-21T02:45:12.430Z
1048569  ObjectId(5dfd8738f52263448f85e945)  2019-12-21T02:45:12.240Z
1048570  ObjectId(5dfd8737f52263448f85e944)  2019-12-21T02:45:11.689Z
1048571  ObjectId(5dfd8737f52263448f85e943)  2019-12-21T02:45:11.491Z
```

	source_ip	identifier	honeypot
0	92.118.37.97	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
1	159.203.201.96	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
2	92.118.37.97	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
3	92.246.85.167	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
4	92.118.37.99	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
...	...	...	...
1048567	202.100.211.228	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
1048568	175.115.247.157	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
1048569	41.41.67.69	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
1048570	41.41.67.69	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea
1048571	43.251.254.43	51b93cd6-22c9-11ea-8808-baca61a13142	dionaea

[1048572 rows x 7 columns]

Figure 5.3.1 Outlier Analysis

4.4 Exploratory Data Analysis

Data scientists utilise exploratory data analysis (EDA) to examine and analyse data sets and summarise their key properties, frequently using data visualisation techniques.

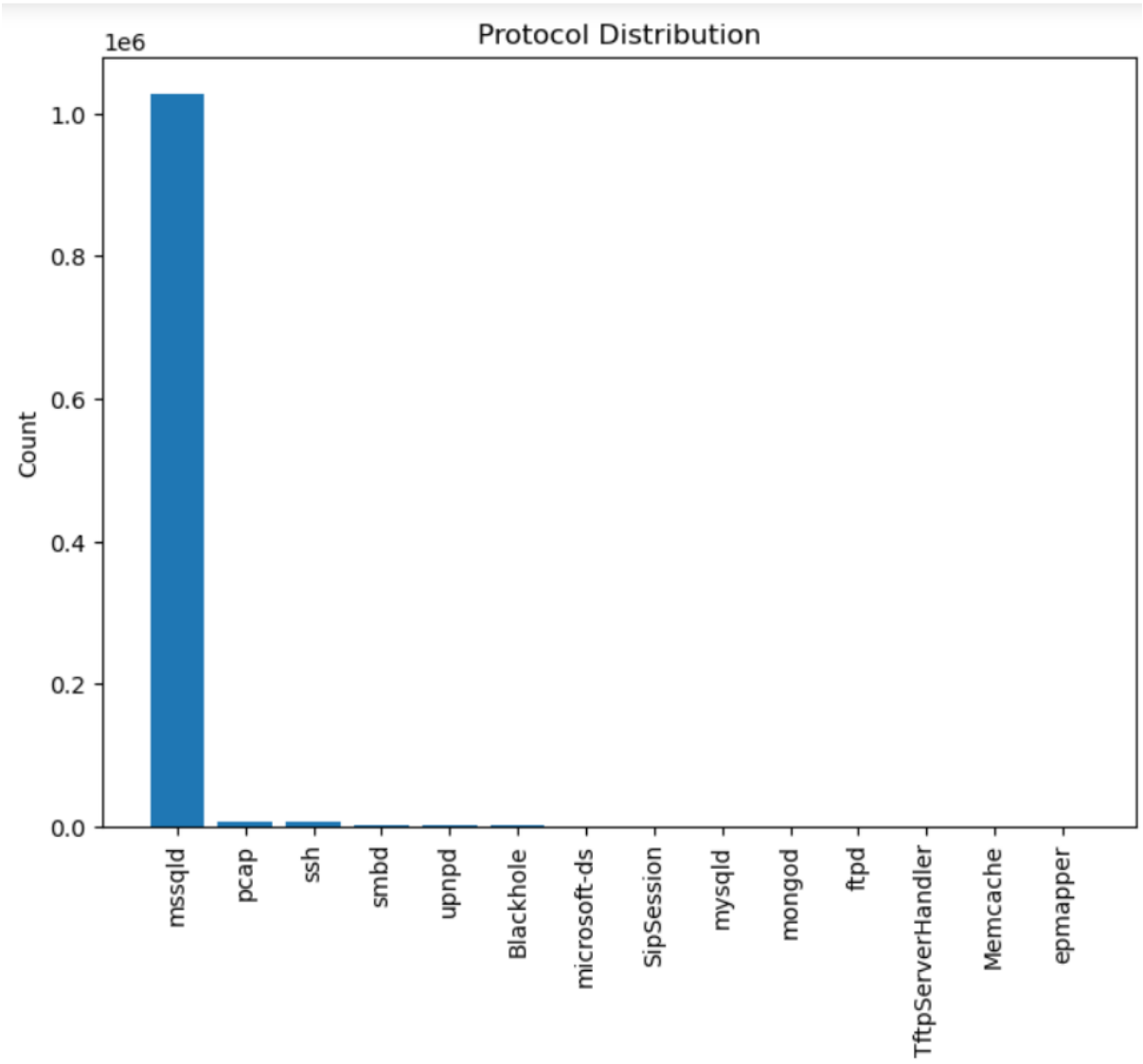


Figure 5.4.1 Protocol Distribution

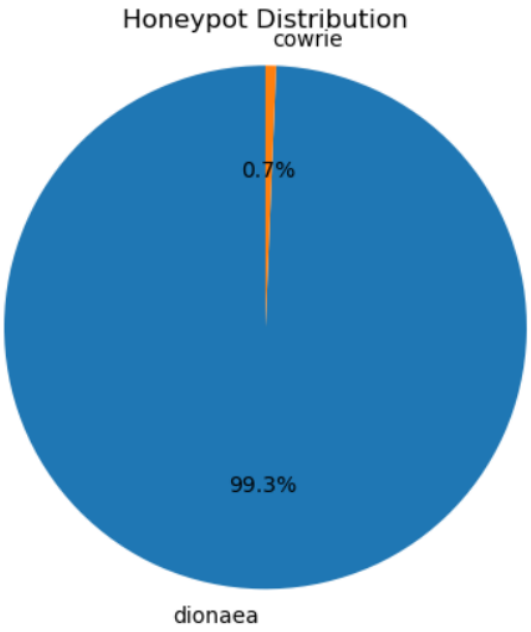


Figure 5.4.2 Honeypot Distribution

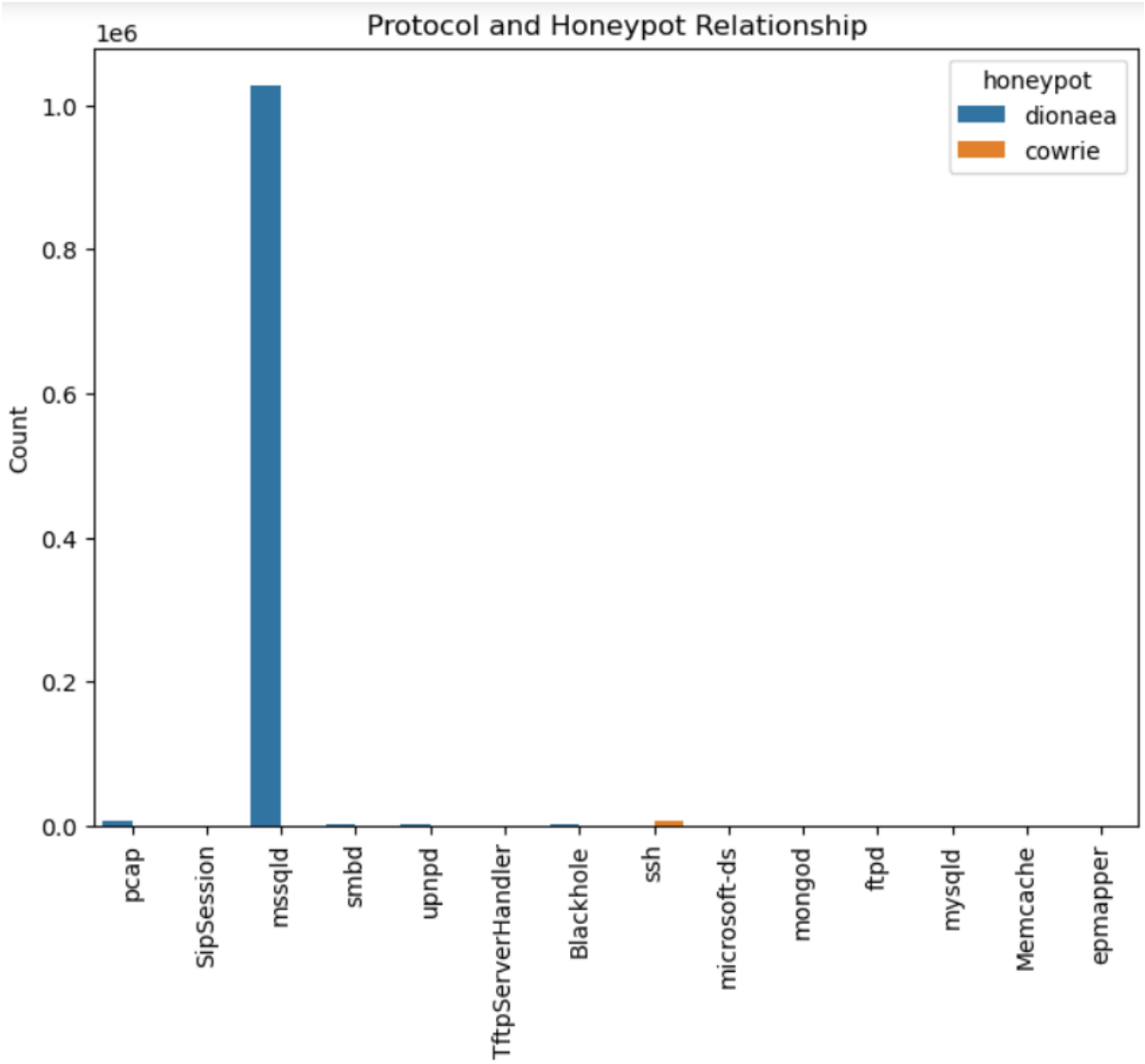


Figure 5.4.3 Relationship between protocol and honeypot

```
Feature Engineering
                                  _id    protocol  \
0  ObjectId(5dfc265ff5226364466d3346)        pcap
1  ObjectId(5dfc269ef5226364466d3348)        pcap
2  ObjectId(5dfc26aaf5226364466d334a)        pcap
3  ObjectId(5dfc26adf5226364466d334c)  SipSession
4  ObjectId(5dfc26d1f5226364466d334e)        pcap

                            hpfeed_id                        timestamp  \
0  ObjectId(5dfc265cf5226364466d3345) 2019-12-20 01:39:40.740000+00:00
1  ObjectId(5dfc269cf5226364466d3347) 2019-12-20 01:40:44.076000+00:00
2  ObjectId(5dfc26a8f5226364466d3349) 2019-12-20 01:40:56.920000+00:00
3  ObjectId(5dfc26abf5226364466d334b) 2019-12-20 01:40:59.011000+00:00
4  ObjectId(5dfc26d0f5226364466d334d) 2019-12-20 01:41:36.128000+00:00

        source_ip                            identifier honeypot  hour  \
0    92.118.37.97  51b93cd6-22c9-11ea-8808-baca61a13142  dionaea     1
1  159.203.201.96  51b93cd6-22c9-11ea-8808-baca61a13142  dionaea     1
2    92.118.37.97  51b93cd6-22c9-11ea-8808-baca61a13142  dionaea     1
3   92.246.85.167  51b93cd6-22c9-11ea-8808-baca61a13142  dionaea     1
4    92.118.37.99  51b93cd6-22c9-11ea-8808-baca61a13142  dionaea     1

   day_of_week
0            4
1            4
2            4
3            4
4            4
```

Figure 5.4.4 Feature Engineering

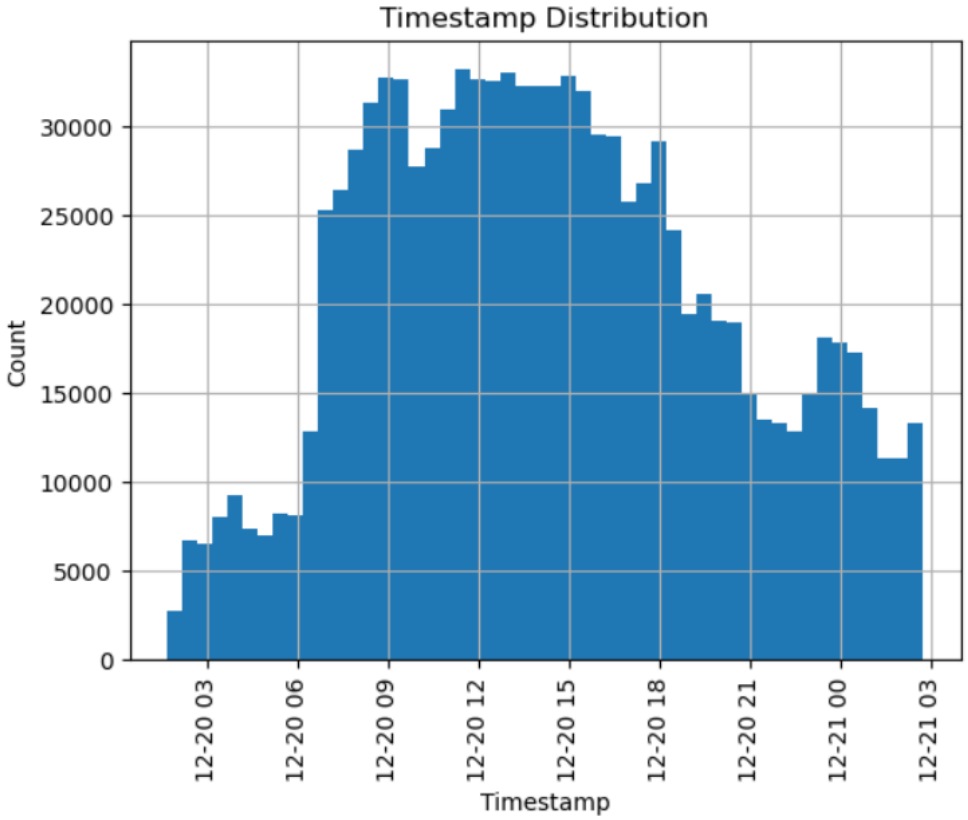


Figure 5.4.5 Timestamp Distribution

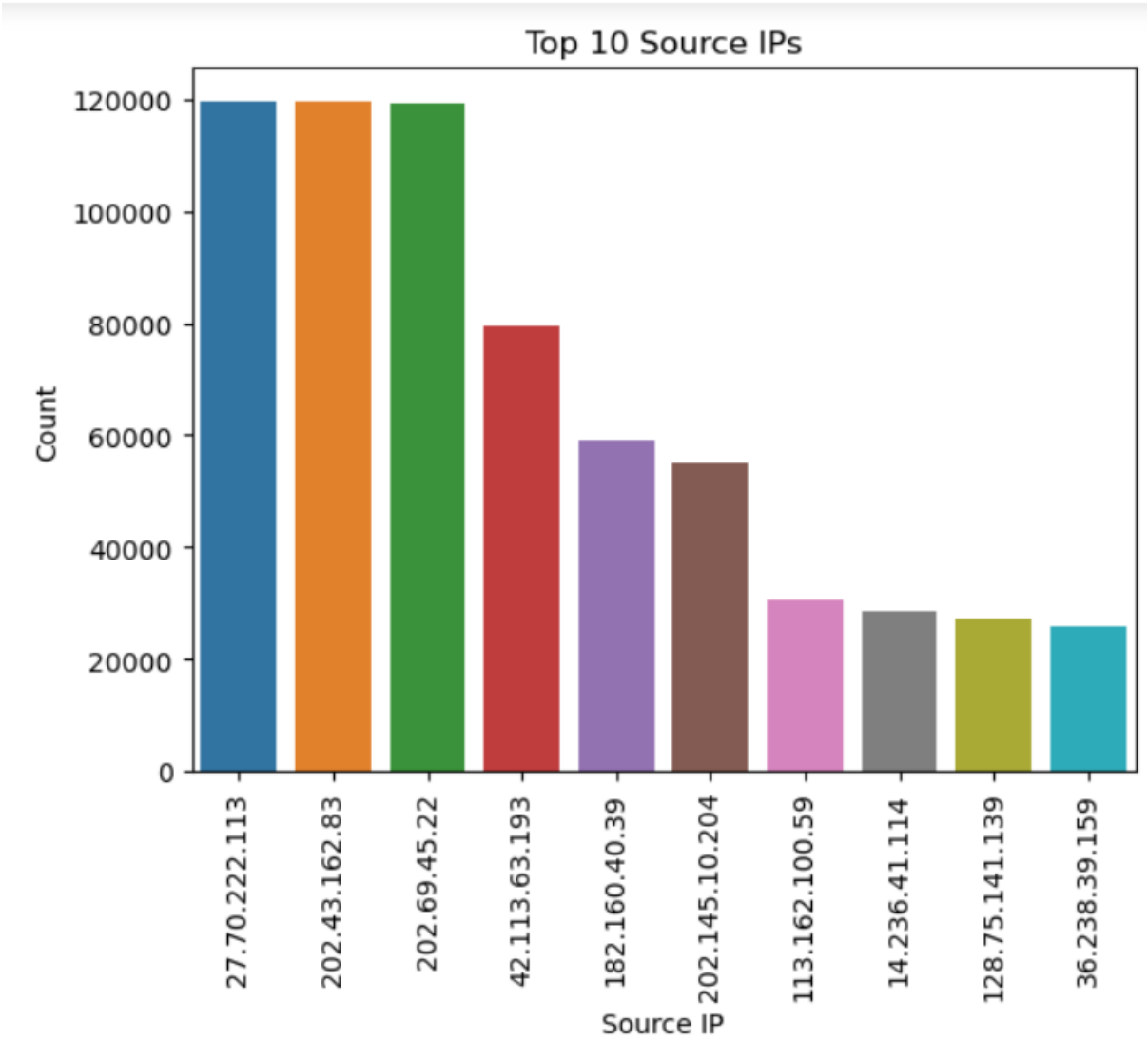


Figure 5.4.6 Top 10 Source Ip

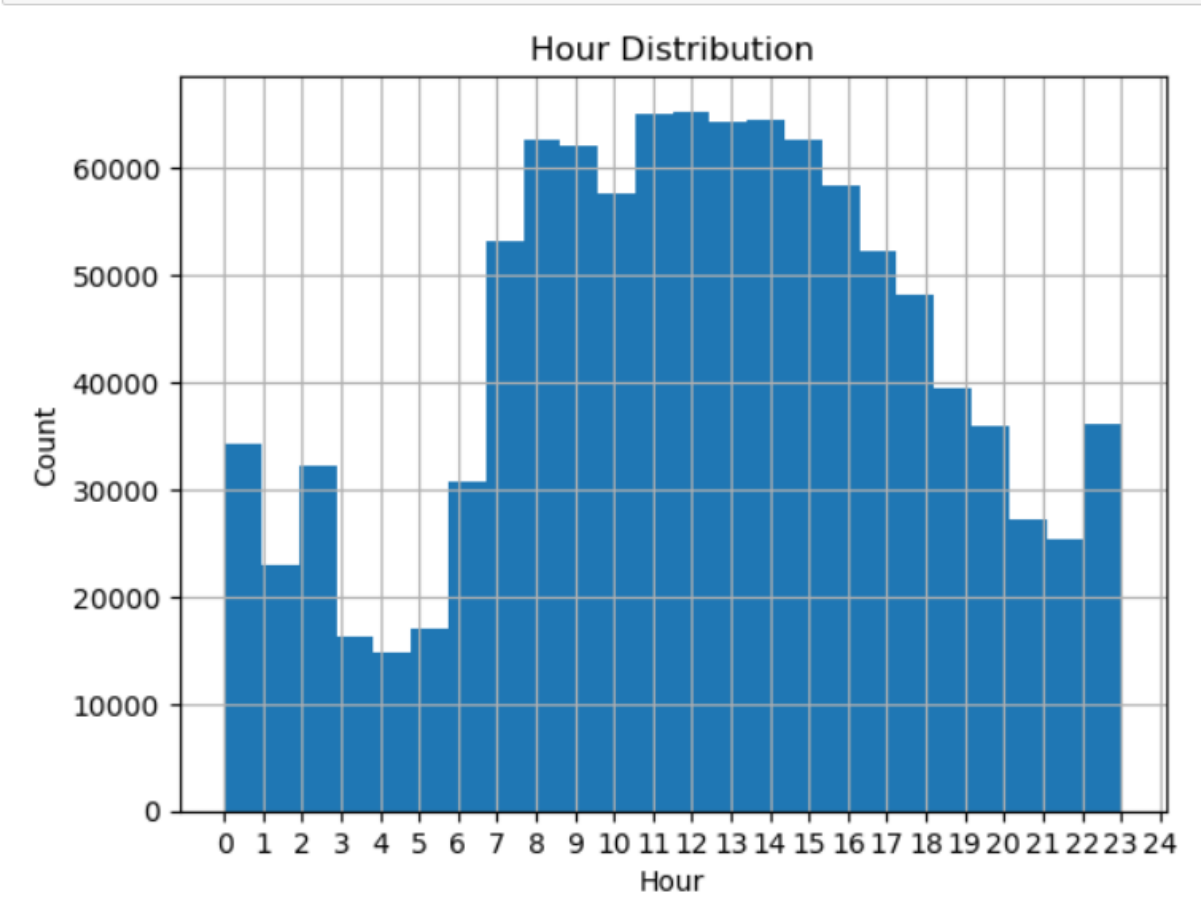


Figure 5.4.7 Hour Distribution

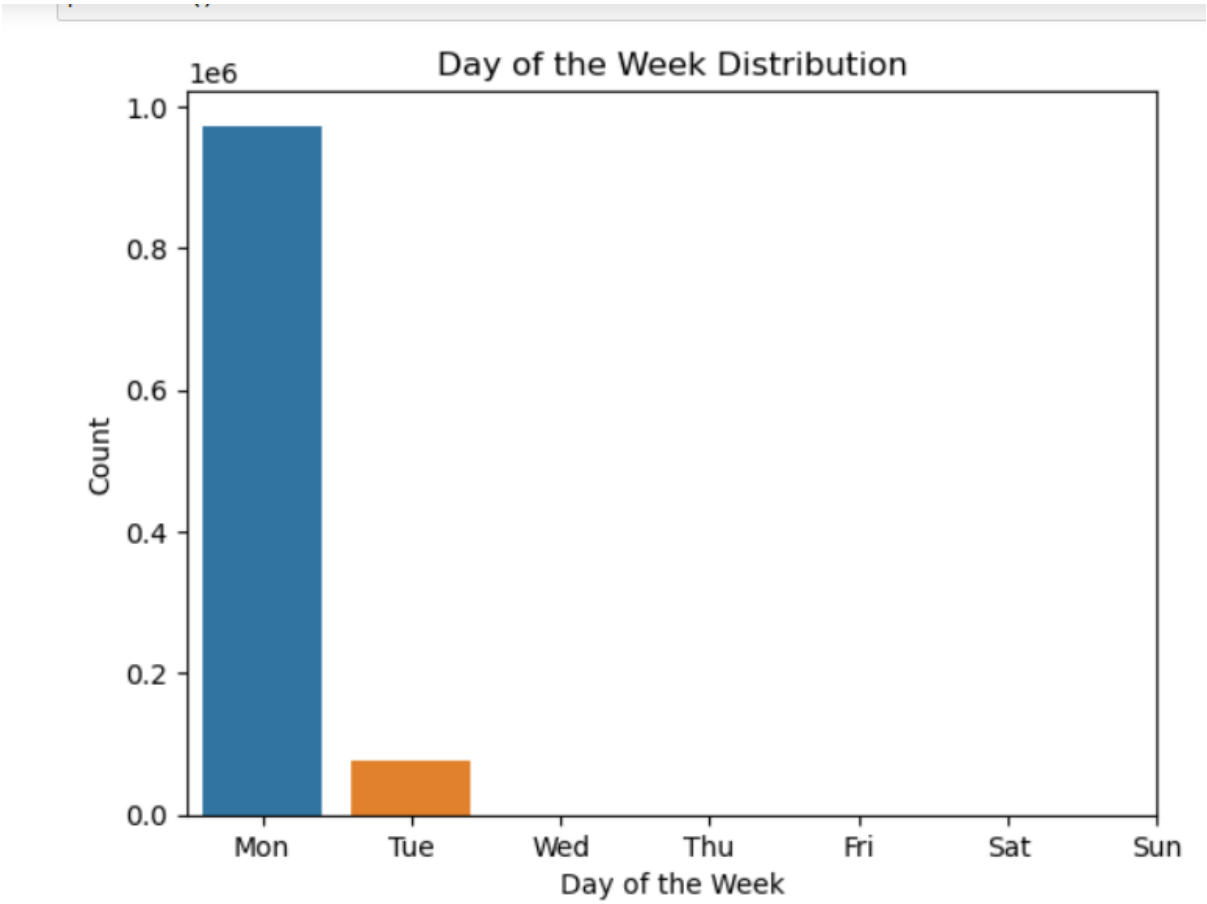


Figure 5.4.8 Day of the week distribution

Model Building

In [12]:
```
## Prepare the data for modeling
X = df[['hour', 'day_of_week', 'protocol', 'honeypot']]
X_encoded = pd.get_dummies(X)

X_encoded.head()
```

Out[12]:

	hour	day_of_week	protocol_Blackhole	protocol_Memcache	protocol_SipSession	protocol_TftpServerHandler	protocol_epmapper	protocol_ftpd	protocol_mic
0	1	4	0	0	0	0	0	0	
1	1	4	0	0	0	0	0	0	
2	1	4	0	0	0	0	0	0	
3	1	4	0	0	1	0	0	0	
4	1	4	0	0	0	0	0	0	

Figure 5.4.9 Model Building

Model Building

In [12]:
```
## Prepare the data for modeling
X = df[['hour', 'day_of_week', 'protocol', 'honeypot']]
X_encoded = pd.get_dummies(X)

X_encoded.head()
```

Out[12]:

crosoft-ds	protocol_mongod	protocol_mssqld	protocol_mysqld	protocol_pcap	protocol_smbd	protocol_ssh	protocol_upnpd	honeypot_cowrie	honeypot_dionaea
0	0	0	0	1	0	0	0	0	1
0	0	0	0	1	0	0	0	0	1
0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	1

Figure 5.4.10 Model Building

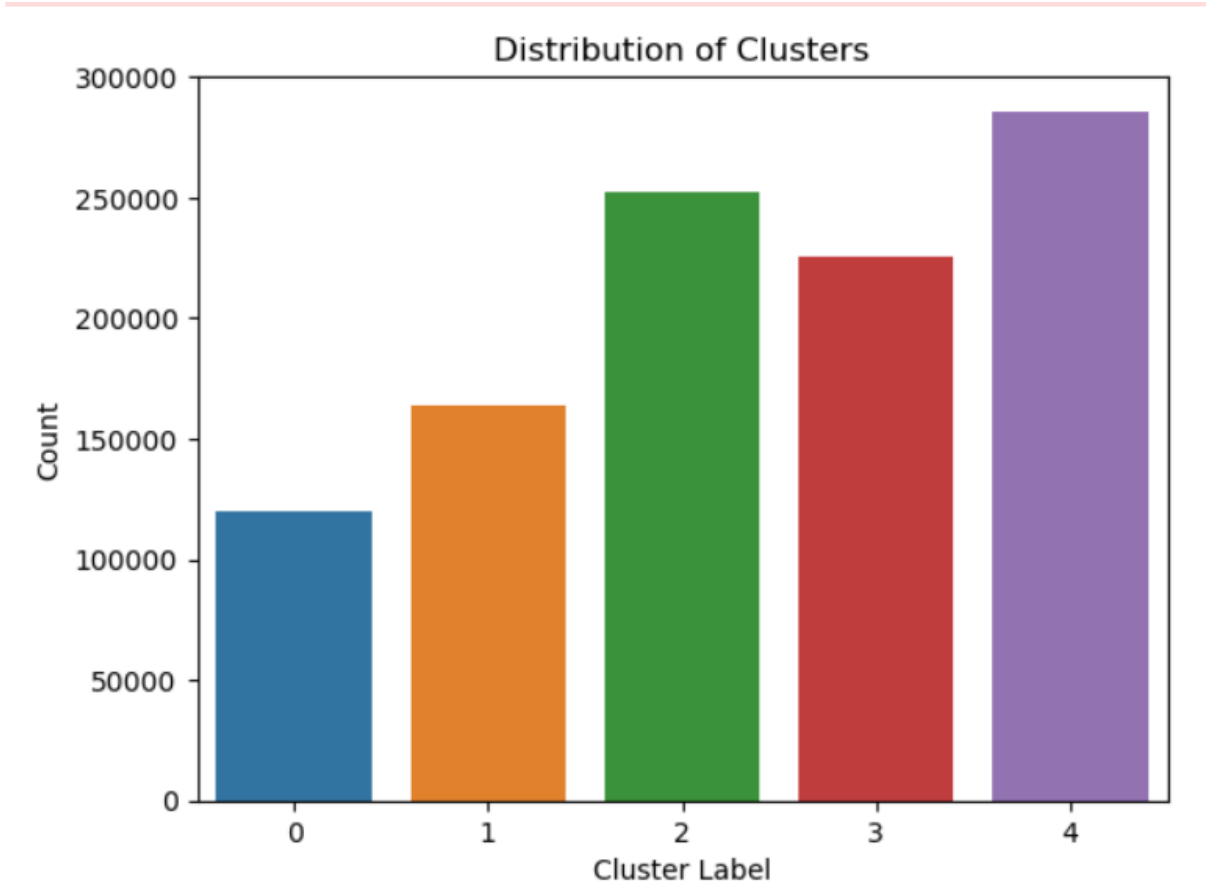


Figure 5.4.11 Cluster Distribution

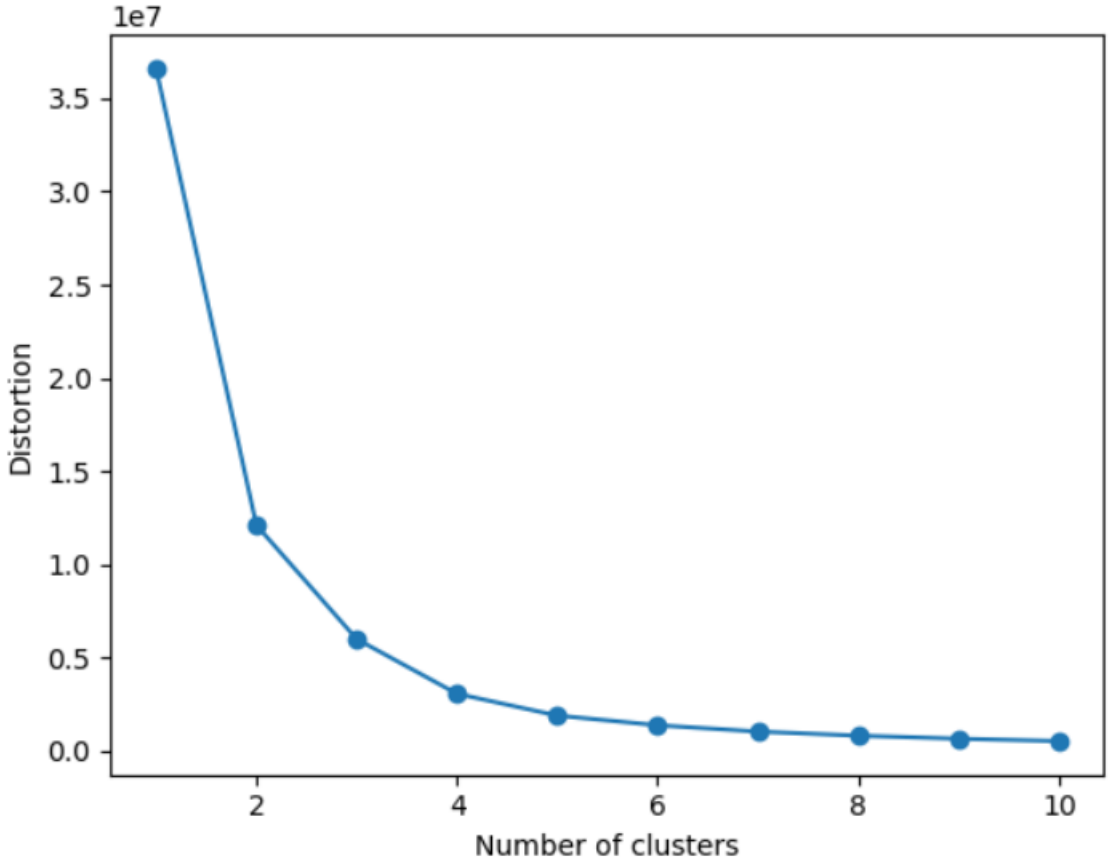


Figure 5.4.12 Elbow Method for cluster distribution

```
## Fit KMeans clustering model with optimal number of clusters and predict cluster labels
kmeans = KMeans(n_clusters=5, random_state=42, n_init='auto')
clusters = kmeans.fit_predict(X_encoded)

## Add cluster labels to the dataframe
df['cluster'] = clusters
print('K-means Prediction')
print(clusters)
```

```
K-means Prediction
[4 4 4 ... 4 4 4]
```

Figure 5.4.13 K-means Prediction

```
# Calculate Davies Bouldin Score
davies_bouldin = davies_bouldin_score(X_encoded, clusters)
print('Davies Bouldin Score:', davies_bouldin)
```

```
Davies Bouldin Score: 0.49114245443298765
```

```
#Calculate randindex score
rand = rand_score(df['honeypot'],clusters)
print('Rand Index Score:',rand)
```

```
Rand Index Score: 0.22416359024697521
```

Figure 5.4.14 Davies Bouldin & Rand Index Score

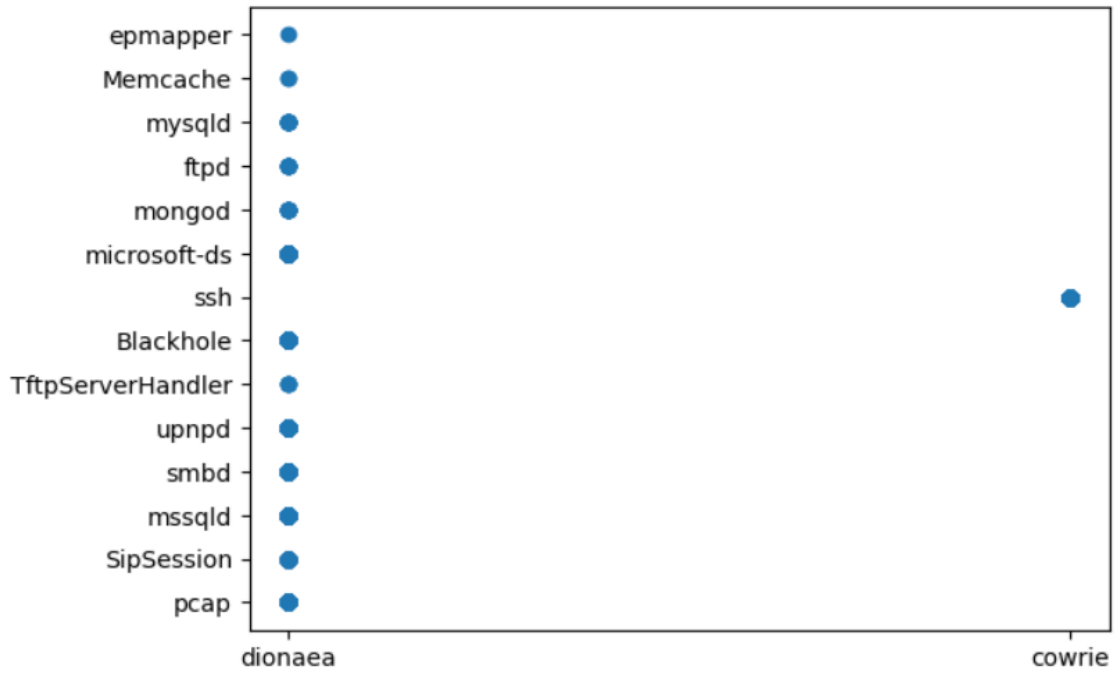


Figure 5.4.15 Distribution of Honeypot and Protocol using KNN

	hour	day_of_week	protocol	honeypot
0	1	4	10	1
1	1	4	10	1
2	1	4	10	1
3	1	4	2	1
4	1	4	10	1
...	...	...	...	...
1048567	2	5	8	1
1048568	2	5	8	1
1048569	2	5	8	1
1048570	2	5	8	1
1048571	2	5	8	1

[1048572 rows x 4 columns]

Figure 5.4.16 Filter Outlier

```
from sklearn.metrics.cluster import adjusted_rand_score
adjusted_rand_score=adjusted_rand_score(df['honeypot'], df['protocol'])
print(adjusted_rand_score)
```

```
0.4947130642088379
```

```
X = df[['hour', 'day_of_week', 'protocol', 'honeypot']]
X_encoded = pd.get_dummies(X)
```

```
db_index = davies_bouldin_score(X_encoded, df['protocol'])
print(db_index)
```

```
3.733145131787928
```

Figure 5.4.17 Davies Bouldin & Rand Index Score using KNN

Davies-Bouldin Index

The Davies-Bouldin Index (DBI) is a clustering evaluation metric used to assess the quality of a clustering algorithm's results. It measures the average similarity between clusters and the dissimilarity between clusters. A lower DBI value indicates better clustering performance.

- Run a clustering algorithm (e.g., K-means) on your dataset and obtain the cluster assignments for each data point.
- Calculate the centroid of each cluster. The centroid is the mean of all the data points within the cluster.
- For each cluster, compute the average distance between its data points and the centroid. This can be any distance metric, such as Euclidean distance or Manhattan distance.
- For each pair of clusters (i, j), calculate the dissimilarity between their centroids. Again, any distance metric can be used.
- **Compute the Davies-Bouldin Index using the formula:**
- **DBI = (1/n) * Σ[i=1 to n] max[(s(i) + s(j)) / d(i, j)]**
- where n is the number of clusters, s(i) is the average distance within cluster i, and d(i, j) is the dissimilarity between cluster i and cluster j.
- The DBI considers both the compactness of individual clusters (represented by s(i)) and the separation

between clusters (represented by d(i, j)). A lower DBI value indicates that the clusters are more distinct and well-separated.

Formula for Davies-Bouldin Index

$$
\begin{aligned}
k &= \text{Number of clusters} \\
s_i &= \text{Average distance between each point in cluster } i \text{ to cluster center } c_i \\
d_{ij} &= \text{Distance between cluster centers } c_i \text{ and } c_j \\
\text{Difference measure, } R_{ij} &= \frac{\text{Average within-cluster distance}}{\text{Between-cluster distance}} \\
&= \frac{s_i + s_j}{d_{ij}} \\
\text{Davies-Bouldin Index } (DB) &= \text{Average of maximum difference measure} \\
&= \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} R_{ij}
\end{aligned}
$$

Rand Index

The Rand Index (RI) is a clustering evaluation metric used to assess the similarity between two data partitionings, such as the predicted clustering from a clustering algorithm and the ground truth partitioning. It measures the agreement between the predicted clustering and the true clustering labels.

The Rand Index considers true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) in comparing the two partitionings. It computes the ratio of agreements (TP + TN) to the total number of pairwise comparisons (TP + TN + FP + FN).

Here's a step-by-step overview of how to calculate the Rand Index:

Obtain the predicted clustering labels from a clustering algorithm for your dataset.

Obtain the ground truth labels or the true partitioning of the data.

Calculate the number of agreements on pairs of data points between the predicted clustering and the ground truth. This includes both pairs that are correctly clustered together (true positives) and pairs that are correctly not clustered together (true negatives).

Calculate the number of disagreements on pairs of data points between the predicted clustering and the ground truth. This includes both pairs that are incorrectly clustered together (false positives) and pairs that are incorrectly not clustered together (false negatives).

Compute the Rand Index using the formula:

RI = (true positives + true negatives) / (true positives + true negatives + false positives + false negatives)

The Rand Index ranges from 0 to 1, where 1 indicates a perfect match between the predicted clustering and the true partitioning, and 0 indicates no agreement.

Formula for Rand Index(RI)

$$\text{Rand Index (RI)} = \frac{\text{Number of pair-wise correct predictions}}{\text{Total number of possible pairs}}$$

<table>
<tr><th>SI.NO</th><th>ALGORITHM</th><th>SCORE</th></tr>
<tr><td>1</td><td>K-MEANS</td><td>
<table>
<tr><th>Davies Bouldin</th><th>Randindex</th></tr>
<tr><td>0.491142454</td><td>0.22416359</td></tr>
</table>
</td></tr>
<tr><td>2</td><td>KNN</td><td>
<table>
<tr><th>Davies Bouldin</th><th>Randindex</th></tr>
<tr><td>3.733145132</td><td>0.494713064</td></tr>
</table>
</td></tr>
</table>

Figure 5.4.18 Comparison table K-Means & KNN

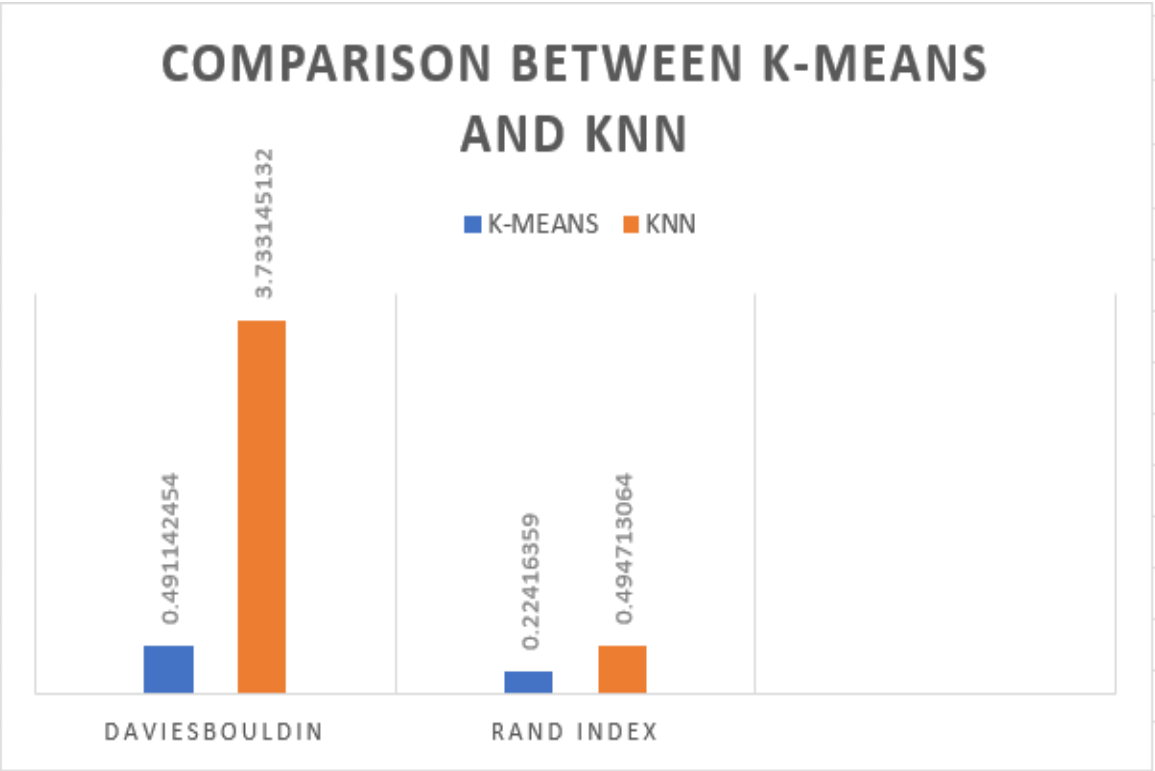


Figure 5.4.19 Comparison chart K-Means & KNN

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

Nowadays, the security aspect of computer networking is becoming critical issue. Production systems that are connected to the Internet are the main target for different cyber-attacks across the world. There are different network security measures implemented to mitigate the impact of attackers. Among the security measures, honeypots are effective network security systems which are built to study the intents and tactics of attackers. Honeypots are well known network security technologies deployed to study various kinds of attacks.

The primary aim of this project is to propose high interaction honeypot system is presented. Typically, the honeypot was deployed to study attacks on secure shell protocol coming from the Internet. The deployment of the proposed honeypot system is discussed in this chapter. In the virtual honeypot deployment, the system configuration and network connections are presented. Large amount of attack data has been gathered and analyzed in detail. Typically, the brute-force and intrusion activities of attacks on SSH protocol are presented. In the coming chapter, the deployment of honeypot is presented. It is a medium interaction honeypot system developed to gather attacks on SSH protocols. By deploying this honeypot, SSH based attack behaviors can be studied. From the attack data gathered a comparison is also made between the proposed honeypot systems.

FUTURE SCOPE

While K-means and KNN are widely used clustering techniques, there are other advanced clustering algorithms that can be explored for anomaly detection in honeypots. Algorithms like DBSCAN (Density-Based Spatial Clustering of Applications with Noise), OPTICS (Ordering Points To Identify the Clustering Structure), or hierarchical clustering methods could be investigated to improve the accuracy and effectiveness of anomaly detection. This approach can improve the accuracy of anomaly detection, reduce false positives, and enable early detection of potential security threats.

One potential future application of anomaly detection in honeypots using clustering methods is in the field of intrusion detection and prevention systems. By identifying and analyzing patterns of malicious behavior, clustering-based anomaly detection can help improve the efficiency and effectiveness of intrusion detection and prevention systems.

In addition, anomaly detection using clustering methods can be useful in identifying insider threats, detecting fraud, and improving network security. As more organizations rely on advanced analytics and

machine learning to detect and prevent security threats, the demand for clustering-based anomaly detection techniques is likely to increase.

CHAPTER 7

REFERENCES

[1] A novel unsupervised classification approach for network anomaly detection by k-Means clustering and ID3 decision tree learning methods Yasser Yasami · Saadat Pour Mozaffari Published online: 9 October 2009 © Springer Science+Business Media, LLC 2009. DOI 10.1007/s11227-009-0338-x

[2] Chitrakar, Roshan; Huang, Chuanhe (2012). *[IEEE 2012 8th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM) - Shanghai, China (2012.09.21-2012.09.23)] 2012 8th International Conference on Wireless Communications, Networking and Mobile Computing - Anomaly Based Intrusion Detection Using Hybrid Learning Approach of Combining k-Medoids Clustering and Naïve Bayes Classification. , (), 1–5.* doi:10.1109/WiCOM.2012.6478433

[3] Chitrakar, Roshan; Chuanhe, Huang (2012). *[IEEE 2012 Third Asian Himalayas International Conference on Internet (AH-ICI) - Kathmundu, Nepal (2012.11.23-2012.11.25)] 2012 Third Asian Himalayas International Conference on Internet - Anomaly detection using Support Vector Machine classification with k-Medoids clustering. , (), 1–5.* doi:10.1109/ahici.2012.6408446

[4] COMBINING NAIVE BAYES AND DECISION TREE FOR ADAPTIVE INTRUSION DETECTION Dewan Md. Farid1 Nouria Harbi1 , and Mohammad Zahidur Rahman2 1ERIC Laboratory, University Lumière Lyon 2 - France dewanfarid@gmail.com, nouria.harbi@univ-lyon2.fr 2Department of Computer Science and Engineering, Jahangirnagar University, Bangladesh rmzahid@juniv.edu Doi: http://dx.doi.org/10.5121/ijnsa.2010.2202

[5] SONG, Jungsuk; TAKAKURA, Hiroki; OKABE, Yasuo; INOUE, Daisuke; ETO, Masashi; NAKAO, Koji (2010). *A Comparative Study of Unsupervised Anomaly Detection Techniques Using Honeypot Data. IEICE Transactions on Information and Systems, E93-D(9), 2544–2554.* doi:10.1587/transinf.e93.d.2544

[6] Mulay, Snehal A.; Devale, P. R.; Garje, G.V. (2010). *[IEEE 2010 International Conference on Networking and Information Technology (ICNIT 2010) - Manila, Philippines (2010.06.11-2010.06.12)] 2010 International Conference on Networking and Information Technology - Decision tree based Support Vector Machine for Intrusion Detection. , (), 59–63.* doi:10.1109/ICNIT.2010.5508557

[7] Innab, Nisreen; Alomairy, Eman; Alsheddi, Lamya (2018). *[IEEE 2018 21st Saudi Computer Society National Computer Conference (NCC) - Riyadh, Saudi Arabia (2018.4.25-2018.4.26)] 2018 21st Saudi Computer Society National Computer Conference (NCC) - Hybrid System Between Anomaly Based Detection System and Honeypot to Detect Zero Day Attack. , (), 1–5.* doi:10.1109/NCG.2018.8593030

[8] J. Levine; J. Grizzard; H. Owen, "Application of a methodology to characterize rootkits retrieved from honeynets" Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC 2004, page 15 – 21.

[9] R. McGrew; B. Rayford; J.R Vaughn, "Experiences with Honeypot Systems: Development, Deployment, and Analysis", System Sciences, 2006. HICSS '06. Proceedings of the 39th Annual Hawaii International Conference on, on page(s): 220a - 220a Volume: 9, 04- 07 Jan. 2006

[10] M. O'Leary; S. Azadegan; J. Lakhani, "Development of a Honeynet laboratory: A case study," Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'06).

[11] F. Zhang et al, "Honeypot: A Supplemented Active Defense System for Network Security," in Proceeding's 4th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'03), Chengdu, China, 27-29 Aug. 2003, pages 231-235.

[12] S. Almotairi; A. Clark; G. Mohay; J. Zimmermann, "A technique for detecting new attacks in low-interaction Honeypot traffic," 2009 Fourth International Conference on Internet Monitoring and Protection, 2009.

[13] N. Tymoshyk; R. Tymoshyk; A. Piskozub; P. Khromchak; V. Pyvovarov; A. Novak, "Monitoring of malefactor's activity in virtualized honeypots on the base of semantic transformation in Qemu

hypervisor," 2009 IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Sep. 2009

[14] V. Pejovic; I. Kovaicevic; S. Bojanic; C. Leita; J. Popovic; O. NietoTaladriz, "Migrating a HoneyDepot to hardware," The International Conference on Emerging Security Information, Systems, and Technologies (SECUREWARE 2007), 2007

[15] J. Bhatia; Sehgal; S. Kumar, "Botnet command detection using virtual Honeynet," International Journal of Network Security & Its Applications, Volume 3, no. 5, pages 177–189, Sep. 2011

[16] M. Gruber; F. Fankhauser; S. Taber; C. Schanes; T. Grechenig, "Trapping and analyzing malicious voip traffic using a honeynet approach", In The 6th International Conference on Internet Technology and Secured Transactions (ICITST), pages 442-447, Dec. 2011.

[17] J. Ma; K. Chai; Y. Xiao; T. Lan; W. Huang, "High-interaction Honeypot system for SQL injection analysis", 2011 International Conference of Information Technology, Computer Engineering and Management Sciences, Sep. 2011.

[18] J. Hastings; D.M. Laverty; D.J. Morrow, "Tracking smart grid hackers", 2014 49th International Universities Power Engineering Conference (UPEC), Sep. 2014

[19] A. Puska; M. Nogueira; A. Santos, "Unwanted traffic characterization on IP networks by low interactive honeypot", 10th International Conference on Network and Service Management (CNSM) and Workshop, Nov. 2014.

[20] G. Kaur; J. S. Saini, "Implementation of high interaction Honeypot to analyze the network traffic and prevention of attacks on protocol/port basis", International Journal of Computer Applications, Volume 62, no. 16, pages 22–29, Jan. 2013.

[21] S. Li; R. Schmitz, “A novel anti-phishing framework based on honeypots”, 2009 eCrime Researchers Summit, Oct. 2009.

[22] L. Spitzner, “Honeypots: Catching the insider threat”, Proceedings 19th Annual Computer Security Application Conference, pages170 - 179, 2003

[23] K. Lin; L. Kyaw, “Hybrid Honeypot System for Network Security”, World Academy of Science, Engineering and Technology, 2008(48).

[24] Z. Li-juan, “Honeypot-based defense system research and design,” 2009 2nd IEEE International Conference on Computer Science and Information Technology, 2009

[25] R. Hu; J. Zeng; H. Huang; J. Xia, “Study and design of dynamic load balancing in hybrid Honeynet”, 2011 First International Conference on Instrumentation, Measurement, Computer, Communication and Control, Oct. 2011

CHAPTER 8

8. APPENDIX

8.1 Sample Coding:

```
# # IMPORT LIBRARIES

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.cluster import KMeans

from sklearn.metrics import davies_bouldin_score, silhouette_score, rand_score

# # Read Dataset

df = pd.read_csv("C:/Users/ADMIN/Honeypot_cleaned_dataset.csv")

# # Outlier Analysis

# Apply outlier analysis

Q1 = df.quantile(0.25)

Q3 = df.quantile(0.75)

IQR = Q3 - Q1

df = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]

# Print the dataframe after removing outliers

print('Outlier Analysis')

print(df)

# # Exploratory Data Analysis

## Protocol Distribution

protocol_counts = df['protocol'].value_counts()
```

```
fig, ax = plt.subplots(figsize=(8, 6))

ax.bar(protocol_counts.index, protocol_counts.values)

plt.xticks(rotation=90)

ax.set_title('Protocol Distribution')

ax.set_xlabel('Protocol')

ax.set_ylabel('Count')

plt.show()

## Honeypot Distribution

honeypot_counts = df['honeypot'].value_counts()

fig, ax = plt.subplots()

ax.pie(honeypot_counts.values, labels=honeypot_counts.index, autopct='%1.1f%%',
startangle=90)

ax.axis('equal')

ax.set_title('Honeypot Distribution')

plt.show()

## Protocol and Honeypot Relationship

fig, ax = plt.subplots(figsize=(8, 6))

ax = sns.countplot(x='protocol', hue='honeypot', data=df)

plt.xticks(rotation=90)

ax.set_title('Protocol and Honeypot Relationship')

ax.set_xlabel('Protocol')

ax.set_ylabel('Count')

plt.show()

# # Feature Engineering

## Convert timestamp to datetime
```

```
df['timestamp'] = pd.to_datetime(df['timestamp'])
## Create new columns for hour and day of the week
df['hour'] = df['timestamp'].dt.hour
df['day_of_week'] = df['timestamp'].dt.dayofweek
# Check the updated dataframe after feature engineering
print('Feature Engineering')
print(df.head())
# Check the distribution of the 'timestamp' column using a histogram
df['timestamp'] = pd.to_datetime(df['timestamp'])
df['timestamp'].hist(bins=50)
plt.xticks(rotation=90)
plt.title('Timestamp Distribution')
plt.xlabel('Timestamp')
plt.ylabel('Count')
plt.show()
# Check the top 10 source IPs with the most events using a bar plot
top_ips = df['source_ip'].value_counts().head(10)
sns.barplot(x=top_ips.index, y=top_ips.values)
plt.xticks(rotation=90)
plt.title('Top 10 Source IPs')
plt.xlabel('Source IP')
plt.ylabel('Count')
plt.show()
## Histogram of the 'hour' column
```

```
df['hour'].hist(bins=24)
plt.xticks(range(0, 25))
plt.title('Hour Distribution')
plt.xlabel('Hour')
plt.ylabel('Count')
plt.show()
## Bar plot of the 'day_of_week' column
day_counts = df['day_of_week'].value_counts()
sns.barplot(x=day_counts.index, y=day_counts.values)
plt.xticks(range(0, 7), ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'])
plt.title('Day of the Week Distribution')
plt.xlabel('Day of the Week')
plt.ylabel('Count')
plt.show()
# # Model Building
## Prepare the data for modeling
X = df[['hour', 'day_of_week', 'protocol', 'honeypot']]
X_encoded = pd.get_dummies(X)
X_encoded.head()
## Fit KMeans clustering model and predict cluster labels
kmeans = KMeans(n_clusters=5, random_state=42)
clusters = kmeans.fit_predict(X_encoded)
## Add cluster labels to the dataframe
df['cluster'] = clusters
```

```
## Plot the distribution of cluster labels
sns.countplot(x='cluster', data=df)
plt.title('Distribution of Clusters')
plt.xlabel('Cluster Label')
plt.ylabel('Count')
plt.show()
## Identify anomalous events in the data
anomalous_events = kmeans.predict(X_encoded)
## Determine optimal number of clusters using the Elbow method
distortions = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(X_encoded)
    distortions.append(kmeans.inertia_)
plt.plot(range(1, 11), distortions, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Distortion')
plt.show()
## Fit KMeans clustering model with optimal number of clusters and predict cluster labels
#kmeans = KMeans(n_clusters=5, random_state=42, n_init='auto')
kmeans = KMeans(n_clusters=5, random_state=42)
clusters = kmeans.fit_predict(X_encoded)

## Add cluster labels to the dataframe
```

```
df['cluster'] = clusters

print('K-means Prediction')

print(clusters)

# Calculate Davies Bouldin Score

davies_bouldin = davies_bouldin_score(X_encoded, clusters)

print('Davies Bouldin Score:', davies_bouldin)

#Calculate randindex score

rand = rand_score(df['honeypot'],clusters)

print('Rand Index Score:',rand)
```

KNN

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.neighbors import NearestNeighbors

from sklearn.preprocessing import normalize

from sklearn import preprocessing

data = df[["honeypot", "protocol"]]

# scatterplot of inputs data

ax.set_title('Protocol and Honeypot using KNN')

ax.set_xlabel('Honeypot')

ax.set_ylabel('Protocol')

plt.show()

plt.scatter(data["honeypot"], data["protocol"])

from sklearn.preprocessing import normalize
```

```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
# Encode labels in column 'honeypot'.
df['honeypot']= label_encoder.fit_transform(df['honeypot'])
print(df.head())
# Encode labels in column 'protocol'.
df['protocol']= label_encoder.fit_transform(df['protocol'])
print(df.head())
df['_id']= label_encoder.fit_transform(df['_id'])
print(df.head())
df['hpfeed_id']= label_encoder.fit_transform(df['hpfeed_id'])
print(df.head())
df['timestamp']= label_encoder.fit_transform(df['timestamp'])
print(df.head())
# Encode labels in column 'source_ip'.
df['source_ip']= label_encoder.fit_transform(df['source_ip'])
print(df.head())
df['identifier']= label_encoder.fit_transform(df['identifier'])
print(df.head())
# create arrays
X = df.values
# instantiate model
nbrs = NearestNeighbors(n_neighbors = 5)
# fit model
```

```
nbrs.fit(X)
# distances and indexes of k-neaighbors from model outputs
distances, indexes = nbrs.kneighbors(X)
# plot mean of k-distances of each observation
plt.plot(distances.mean(axis =1))
outlier_index = np.where(distances.mean(axis = 1) > 0.15)
outlier_index
outlier_values = df.iloc[outlier_index]
outlier_values
# plot data
plt.scatter(df["honeypot"], df["protocol"], color = "b", s = 65)
# plot outlier values
plt.scatter(outlier_values["honeypot"], outlier_values["protocol"], color = "r")
## Fit KMeans clustering model and predict cluster labels
nbrs = NearestNeighbors(n_neighbors=5, algorithm = 'ball_tree')
clusters = nbrs.fit(X)
print(X)
from sklearn.metrics.cluster import adjusted_rand_score
adjusted_rand_score=adjusted_rand_score(df['honeypot'], df['protocol'])
print(adjusted_rand_score)
X = df[['hour', 'day_of_week', 'protocol', 'honeypot']]
X_encoded = pd.get_dummies(X)
db_index = davies_bouldin_score(X_encoded, df['protocol'])
print(db_index)
```

8.3Screen Shots

IMPORT LIBRARIES

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
from sklearn.metrics import davies_bouldin_score, silhouette_score, rand_score
```

Figure 8.3.1 Importing Libraries

Read Dataset

```
df = pd.read_csv("C:/Users/ADMIN/Honeypot_cleaned_dataset.csv")
```

Figure 8.3.2 Load and Read the Honeypot dataset

Outlier Analysis

```
# Apply outlier analysis
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
df = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
# Print the dataframe after removing outliers
print('Outlier Analysis')
print(df)
```

Figure 8.3.3 Outlier Analysis

Exploratory Data Analysis

```
## Protocol Distribution
protocol_counts = df['protocol'].value_counts()
fig, ax = plt.subplots(figsize=(8, 6))
ax.bar(protocol_counts.index, protocol_counts.values)
plt.xticks(rotation=90)
ax.set_title('Protocol Distribution')
ax.set_xlabel('Protocol')
ax.set_ylabel('Count')
plt.show()
```

Figure 8.3.4 Protocol Distribution

```
## Honeypot Distribution
honeypot_counts = df['honeypot'].value_counts()
fig, ax = plt.subplots()
ax.pie(honeypot_counts.values, labels=honeypot_counts.index, autopct='%1.1f%%', startangle=90)
ax.axis('equal')
ax.set_title('Honeypot Distribution')
plt.show()
```

Figure 8.3.5 Honeypot Distribution

```
## Protocol and Honeypot Relationship
fig, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x='protocol', hue='honeypot', data=df)
plt.xticks(rotation=90)
ax.set_title('Protocol and Honeypot Relationship')
ax.set_xlabel('Protocol')
ax.set_ylabel('Count')
plt.show()
```

Figure 8.3.6 Relationship between Protocol and Honeypot

Feature Engineering

```
## Convert timestamp to datetime
df['timestamp'] = pd.to_datetime(df['timestamp'])

## Create new columns for hour and day of the week
df['hour'] = df['timestamp'].dt.hour
df['day_of_week'] = df['timestamp'].dt.dayofweek

# Check the updated dataframe after feature engineering
print('Feature Engineering')
print(df.head())
```

Figure 8.3.7 Feature Engineering

```
# Check the distribution of the 'timestamp' column using a histogram
df['timestamp'] = pd.to_datetime(df['timestamp'])
df['timestamp'].hist(bins=50)
plt.xticks(rotation=90)
plt.title('Timestamp Distribution')
plt.xlabel('Timestamp')
plt.ylabel('Count')
plt.show()
```

Figure 8.3.8 Timestamp Distribution

```
# Check the top 10 source IPs with the most events using a bar plot
top_ips = df['source_ip'].value_counts().head(10)
sns.barplot(x=top_ips.index, y=top_ips.values)
plt.xticks(rotation=90)
plt.title('Top 10 Source IPs')
plt.xlabel('Source IP')
plt.ylabel('Count')
plt.show()
```

Figure 8.3.9 Top 10 Source IP

```
## Histogram of the 'hour' column
df['hour'].hist(bins=24)
plt.xticks(range(0, 25))
plt.title('Hour Distribution')
plt.xlabel('Hour')
plt.ylabel('Count')
plt.show()
```

Figure 8.3.10 Hour Distribution

```
## Bar plot of the 'day_of_week' column
day_counts = df['day_of_week'].value_counts()
sns.barplot(x=day_counts.index, y=day_counts.values)
plt.xticks(range(0, 7), ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'])
plt.title('Day of the Week Distribution')
plt.xlabel('Day of the Week')
plt.ylabel('Count')
plt.show()
```

Figure 8.3.11 Day of week Distribution

```
## Fit KMeans clustering model and predict cluster labels
kmeans = KMeans(n_clusters=5, random_state=42)
clusters = kmeans.fit_predict(X_encoded)

## Add cluster labels to the dataframe
df['cluster'] = clusters
## Plot the distribution of cluster labels
sns.countplot(x='cluster', data=df)
plt.title('Distribution of Clusters')
plt.xlabel('Cluster Label')
plt.ylabel('Count')
plt.show()
```

Figure 8.3.12 Distribution of Cluster

```
## Identify anomalous events in the data
anomalous_events = kmeans.predict(X_encoded)
```

```
## Determine optimal number of clusters using the Elbow method
distortions = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(X_encoded)
    distortions.append(kmeans.inertia_)

plt.plot(range(1, 11), distortions, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Distortion')
plt.show()
```

Figure 8.3.13 Distribution of cluster using Elbow method

```
## Fit KMeans clustering model with optimal number of clusters and predict cluster labels
#kmeans = KMeans(n_clusters=5, random_state=42, n_init='auto')
kmeans = KMeans(n_clusters=5, random_state=42)
clusters = kmeans.fit_predict(X_encoded)

## Add cluster labels to the dataframe
df['cluster'] = clusters
print('K-means Prediction')
print(clusters)
```

Figure 8.3.14 K-means Prediction

```
# Calculate Davies Bouldin Score
davies_bouldin = davies_bouldin_score(X_encoded, clusters)
print('Davies Bouldin Score:', davies_bouldin)
```

```
Davies Bouldin Score: 0.49114245443298765
```

```
#Calculate randindex score
rand = rand_score(df['honeypot'],clusters)
print('Rand Index Score:',rand)
```

```
Rand Index Score: 0.22416359024697521
```

Figure 8.3.15 Davies Bouldin & Rand Index Score

```
data = df[["honeypot", "protocol"]]
```

```
# scatterplot of inputs data
ax.set_title('Protocol and Honeypot Distribution using KNN')
ax.set_xlabel('Honeypot')
ax.set_ylabel('Protocol')
plt.show()
plt.scatter(data["honeypot"], data["protocol"])
```

Figure 8.3.16 Protocol and Honeypot Distribution using KNN

```
## Fit KMeans clustering model and predict cluster labels
nbrs = NearestNeighbors(n_neighbors=5, algorithm = 'ball_tree')
clusters = nbrs.fit(X)
print(X)
```

Figure 8.3.17 Clustering model and predict cluster labels

```
from sklearn.metrics.cluster import adjusted_rand_score
adjusted_rand_score=adjusted_rand_score(df['honeypot'], df['protocol'])
print(adjusted_rand_score)
```

```
0.4947130642088379
```

```
X = df[['hour', 'day_of_week', 'protocol', 'honeypot']]
X_encoded = pd.get_dummies(X)
```

```
db_index = davies_bouldin_score(X_encoded, df['protocol'])
print(db_index)
```

```
3.733145131787928
```

Figure 8.3.18 Davies Bouldin & Rand Index Score using KNN