Blockchain Activity Data for Use in Collaborative Intrusion Detection Systems

A Thesis Presented

by

Minna Muzemil

to

The Faculty of Informatics

of

St. Mary's University

In Partial Fulfillment of the Requirements

For the Degree of Master of Science

in

Computer Science

Jan 2024.

ACCEPTANCE

Blockchain Activity Data for Use in Collaborative Intrusion Detection System

By

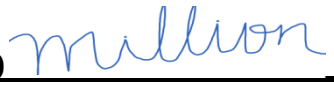
Minna Muzemil

Accepted by the Faculty of Informatics, St. Mary's University, in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Thesis Examination Committee:

_____

Internal Examiner

_Million Meshesha (PhD)  **__March 01, 2024**

External Examiner

_____

Dean, Faculty of Informatics

Jan 2024.

DECLARATION

I, the undersigned, declare that this thesis work is my original work, has not been presented for a degree in this or any other university, and all sources of materials used for the thesis work have been duly acknowledged.

Minna Muzemil

Student

_____

Signature

Addis Ababa

Ethiopia

This thesis has been submitted for examination with my approval as an advisor.

Asrat Mulatu (Ph.D.)

Advisor

_____

Signature

Addis Ababa, Ethiopia

Jan 2024.

Acknowledgments

Foremost, I would like to express my deepest and sincere gratitude to my advisor Dr. Asrat Mulatu for the continuous support of my M.Sc. thesis, and for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me a lot in all parts of the thesis. Besides my advisor, I would like to thank every person who has contributed his/her part in any means to support me for the past two years. Finally, nothing is impossible without the support and motivation of my beloved family.

Table of Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

AR	Attribute Ratio
CART	Classification and Regression Tree
CFS	Correlation-based Feature Selection
CIA	Confidentiality, Integrity, and Availability
DOS	Denial of Service
DT	Decision Tree
FAR	False Alarm Rate
FN	False Negative
FP	False Positive
GB	Gradient Boosted
HIDS	Host-Based Intrusion Detection System
IDS	Intrusion Detection System
IG	Information Gain
KDD	Knowledge Discovery Dataset
LR	Logistic Regression
MI	Mutual Information
ML	Machine Learning
MLlib	Machine Learning Library
MLP	Multilayer Perceptron
NB	Naïve Bayes
NIDS	Network-Based Intrusion Detection System
NN	Neural Network
NSL-KDD	Network Services Library Knowledge Discovery Dataset
PCA	Principal Component Analysis
R2L	Remote to Local
RDD	Resilient Distributed Dataset
RF	Random Forest
SOM	Self-Organizing Map
SPLR	Space Logistic Regression
SVM	Support Vector Machine
U2R	User to Root

Abstract

Today's fast expanding use of information technology has led to a dynamic rise in hacking and other unauthorized operations. The variety and quantity of assaults are increasing dramatically as a result of advancements in both hardware and software. Classifying network traffic is becoming increasingly important because of the rapid increase in Internet users. Every day, numerous threats are developed by people and groups looking to breach computer networks and steal data and personally identifiable information. Many organizations implement a broad defense to thwart these attacks, including setting up robust firewalls, authentication systems, encryption, antivirus software, the newest gear, and so on. A further method for reducing network breaches is intrusion detection. Numerous intrusion detection systems have been created to monitor and identify any unusual behavior on networks or systems. Low detection rate, long training time, and a comparatively high false alarm rate are achieved in the majority of them. In order to address the issues, we put out a strategy that combines the ideas of big data, anomaly detection, and machine learning to produce better outcomes faster. The major components of the proposed system are testing, validation, and training. The gathered training data is preprocessed and sent to the classification model in the training component. We employ and compare four categorization models: Random Forest, Neural Network, Logistic Regression, and Decision Tree.

To discover the best value for each hyperparameter and raise the models' detection rate, the validation component's hyperparameter tuning for each machine learning algorithm use a grid search strategy in conjunction with 5-fold cross-validation. The final model is then constructed by training the classification models with the optimal parameters. Lastly, the test data is divided into normal and attack categories using the trained model. The Apache Spark big data framework is used to create each classification model. Data from assaults and normal conditions are included in the NSL-KDD dataset, which is used for the experimental study. The dataset was divided into three categories: training (80%), validation (10%), and testing (10%). The outcomes demonstrate that nearly every algorithm produces high prediction results.

Neural Network has achieved the greatest results out of all the algorithms, with 96.9% accuracy, 96.8% precision, 96.7% recall, and 96.7% f1-score.

Keywords: Collaborative Intrusion Detection System, Machine Learning, Security Attacks, Neural Network, Anomaly Detection, Intrusion Detection System

Chapter One

Introduction

1.1 Background

The rapid growth of the internet and the use of distributed systems in almost every business aspect of our lives make cybercriminals use their efforts in developing network-based attacks to get their control on the target, the sophistication of the attack is increasing from time to time. The classical usages of antivirus, antispam, antimalware solutions make PCs and data more secure depending on the prior subscriptions that are made for them to refer. On the other hand, Internet Service Providers (ISPs) improve the security of their networks using firewalls by blocking malicious traffic. Further security mechanisms are required as large complicated and distributed network and internet usage growing tremendously and as to be able to get a better solution for the cybercriminals sophisticated attacks as much as possible. Network-based Intrusion Detection/Prevention Systems (NIDS/NIPS) are developed as a solution.

Businesses around the world are worried about their vulnerability to cyber threats as recent outbreaks showed that cyber-attacks are growing tremendously and sophisticatedly with a global impact and with a clear devastation shown. Mega ransomware attacks dominated the news in 2017 with WannaCry and NotPetya. Cryptominers' attacks made headlines in 2018 [1]. In 2019, cyber-attacks have been a mixed bag. Phishing email cyber-attacks remain a constant thorn for most organizations [2].

NIDS are basically intended to monitor, analyze network traffic and to detect variety of attacks inflicted in the network. IDS can be classified as signature-based, and Anomaly based detection. A Signature based IDS designed to detect known attacks using the signatures generated for those attacks, but with a slight modification, the attacker can get access to the network. Anomaly based IDS are useful for detecting unwanted traffic that is specifically unknown. It models the normal network and system behavior based on deviation from normal i.e., anomalies are detected. The advantage is that the profiles of normal activity are customized for every system, hence making it difficult for the attacker. The limitation is that it has a high false alarm rate [3].

Intruders exploit security flaws in the system or network to attack it. Intrusion is a deliberate attempt to obtain information, manipulate information, or render a system untrustworthy or inoperable. Individuals and organizations are affected by security breaches. Privacy and vital

data can be jeopardized as a result of an assault. Attacks are typically triggered by a failure to set security policies and use easily available security technologies. There are numerous real-world examples, including: The Citibank security breach, for example, resulted in the loss of $10 million by the time the crime was discovered in 1994. Only $400,000 was eventually recovered.

While there are several different types of intrusion detection systems (IDS), collaborative IDS (CIDS) offer particular promise in identifying distributed, coordinated attacks that might otherwise elude detection by allowing a set of IDS nodes to exchange required messages and understand the protected environment [4,5]. Even for this type of IDS, there are unresolved issues associated with trusting participants and aggregating data. Due to the distributed nature, malicious nodes within such collaborative network are able to generate untruthful signatures or alerts and share to others. Blockchain technology appears capable of addressing those trusts on the distributed networks by implementing tests for computers that want to join and add blocks to the chain/database called consensus protocol, which require users to prove themselves before they can participate in the blockchain network.

The underlying blockchain technique, which is an ingenious combination of multiple technologies such as peer-to-peer network, consensus protocol over a distributed network, cryptographic schemes, distributed database, smart contract and game theory, provides a decentralized way to build trust in our social and economic activities [4]. That is, in order the block to be added to the blockchain a transaction must occur, the transaction must be verified (here the consensus protocol will be implemented through a network of computers to check the detail of the transaction including transaction date, time, transaction data, participants); the checking through the network of computers makes blockchain decentralized, the transaction must be stored in a block, that block must be given a hash (i.e. a unique identifying code which is the hash of the current and the previous blocks). Once that new block is added to the blockchain, it is publicly available. A transaction in a blockchain can be any type of transaction depending on the system developed using the blockchain technology. For example, for a bank system the transaction can be money, for a candidate's system the transaction can be voted on and when we come to blockchained based Collaborative Intrusion Detection System the transaction is the alert data of the IDS.

For a blockchain network, achieving consensus ensures that all nodes in the network agree upon a consistent global state of the blockchain. A blockchain-based system is a classical distributed system with shared state (i.e. the blockchain) where all participants are geographically

distributed and connected via different kinds of networks. Blockchain platforms can be classified into two main types – permission less and permissioned. Open-ended systems such as Bitcoin and Ethereum are permission less. They are publicly available for use. Any node can conduct transactions as well as take part in the consensus process to advance the blockchain. Permissioned platforms such as Hyperledger Fabric and Multichain are aimed at consortiums where participation is close ended. While clients are allowed to submit transactions, advancing the blockchain is restricted to a fixed set of peering nodes that are run by consortium members. In [6], it explains that many of the industries now a day are incorporating the distributed ledger system, i.e. Blockchain in their business sectors for the advantage of the blockchain to secure their system in its consensus property. Recently research on CIDS is focusing on this technology consensus property to fill the gap that is seen on the current CIDS limitation on building trust between participating parties.

There are several developed mechanisms by which a blockchain based collaborative intrusion detection system can be implemented [5]. Blockchain can be described as a distributed data structure, which is shared and replicated between the participants of a peer-to-peer network. The data structure itself is built from a back-linked list of blocks, where each block is identified by its cryptographic hash and also contains the hash of the previous block [7]. The hash code in the blockchain network make it difficult for hackers to change the transaction once it has happened because a hacker has to edit the hash of the block and the hash of every other block in the blockchain in order to change a single transaction. That is because each block has the hash of its own and the hash of the block before it in the blockchain. This makes it difficult to edit and delete a transaction in the block once the block is added in the chain.

The paper in [7] proposed a framework using blockchain as a mechanism for CIDS. Currently, blockchains have been applied to many domains like IoT, transportation, energy. The strong encryption used to secure blockchains can greatly increase the difficulty for cyber criminals to brute-force their way into private and sensitive environments. Due to these merits, research has started trying to combine blockchains with CIDSs / CIDNs [4].

A significant number of unapproved security events occurred, and in 2000, 70% of firms reported at least one security incident. In comparison to the 1996 report, this indicated a 42 percent increase. According to Joseph and Rod [3], the Computer Emergency Response Team (CERT) reported 3734 events in 1998, 9859 in 1999, and 8836 instances in the first half of 2000 alone. The identification of assaults in network traffic is one of the key objectives of security,

and organizations work hard to preserve the confidentiality, integrity, and availability of their networked resources.

Several strategies have been used to prevent network infiltration. These procedures, while offering some protection, have been shown to be deficient in some areas. As an illustration:

Using a firewall: On a private network, a firewall is a hardware or software solution that is used to enforce security policies. Controlling traffic entering or leaving a private network is its primary function. These rules are merely a list of permissions; as such, they might not always be able to identify invasions. A certain amount of security is provided by firewalls, user authentication, data encryption, and virtual private networks (VPNs), but it is not possible to guard against malicious codes, insider threats, or unprotected modems. thus, would only work as one of the defenses that are available. Cryptography conceals data from unauthorized users, although it might be challenging to determine whether an attack has occurred using this technique. Key management is typically a difficult task. Special key management systems, such as Remote Authentication Dials or Terminal Access Controller Access Systems (TACACS), may be needed for crypto systems. This could refer to specific hardware or configuration in the User Service (RADIUS) server. If not, hackers might be able to access these keys and compromise the system. supplying servers or the network site with physical security: Nevertheless, these are constrained by the possibility that physical security won't offer an effective countermeasure to intruders who use telnet sessions to access a network. Moreover, one method for confirming users of a network resource is authentication. The fact that many people still "use easy to crack passwords" and that some users are negligent or dishonest with their passwords, making them easily obtained by unauthorized users, reduces the effectiveness of this. Additionally, a lot of companies utilize antivirus software, although this might not tell you whether there has been an incursion or not. Antivirus software also needs to be updated often.

We must defend our privacy and selves by utilizing alarms, guards, and other measures since, regrettably, we do not live in a perfect world without fences, gates, locks, or guards, where people coexist harmoniously. The same idea holds true in the realm of computers: in order to protect our data and assets, we need devices and technologies, and one well-known technology is the IDS.

Blockchain data in motion (i.e., block-chain Activity Data) can offer additional opportunities for CIDS analysis, which can help to identify, in near-real time, changes in the blockchain’s

activities. Seeing those changes as they’re happening helps the system to take immediate action so as to protect the system before any attack gets its target.

1.2 Motivation of the study

The motivation behind conducting a study on utilizing blockchain activity data for collaborative Intrusion Detection System (CIDS) with anomaly-based approach stems from utilizing the decentralized features of the blockchain technology for fulfilling the participants trust gap in collaborative Intrusion Detection System (CIDS). By exploring the potential of utilizing blockchain activity data within a Collaborative Intrusion Detection System, the research aims to contribute to the development of effective and efficient security solutions that can bolster the trust, reliability, and resilience of collaborative Intrusion Detection System (CIDS) in a blockchain networks.

Traditional intrusion detection systems often rely on signature-based methods that are ill-suited for the unique characteristics of blockchain networks. The decentralized nature, complex data structures, and diverse transaction formats of blockchain make it interesting in many of the industries. Hence those formats can also be used to enhance the security features of collaborative Intrusion Detection System (CIDS) that incorporates anomaly-based methods. Therefore, there is a pressing need for a novel approach that can leverage the inherent transparency and immutability features of blockchain in the collaborative Intrusion Detection System (CIDS).

The anomaly-based approach within a Collaborative Intrusion Detection System offers a promising solution to this challenge. By focusing on detecting abnormal patterns and behaviors within blockchain activity data, this approach can identify potential security threats that may go unnoticed by traditional methods. Furthermore, by incorporating collaboration and threat intelligence sharing among multiple entities or organizations within the blockchain network, the system can benefit from collective intelligence and provide a more comprehensive defense against evolving attacks.

Even though intrusion detection systems have been available for more than ten years and are suitable for many organizations, there are still a lot of problems with them. The majority of the existing IDS use a signature-based method, which only examines known assaults, and requires constant database updates to detect all attacks in a network, which limits its effectiveness. Thus, it encourages us to integrate anomaly-based systems for attack detection.

The outcomes of this study have the potential to benefit a wide range of stakeholders. Network operators and administrators can leverage the findings to enhance the security of their networks, protect valuable assets, and maintain user trust. Organizations and Users will benefit from the increased security, ensuring the confidentiality and integrity of their transactions and data. Cybersecurity professionals and researchers will gain insights into innovative anomaly detection techniques tailored to the blockchain context, advancing the field of Network security. Regulatory bodies and compliance authorities can utilize the research findings to establish guidelines and standards for secure blockchain implementation. Ultimately, the study aims to contribute to the overall trust and adoption of blockchain technology, enabling its potential to be fully realized across various industries and sectors.

1.3 Statement of the Problem

Recently there is much research done in blockchain based CIDS, most of them are on providing architectural framework on how it can be implemented on anomaly-based approach. The research that is done in blockchain based CIDS as a proof-of-concepts is mostly on signature-based detection method.

Using a signature-based Collaborative Intrusion Detection System (CIDS) for analyzing blockchain activity data presents certain limitations than an anomaly-based approach. The key problem lies in the nature of blockchain data and the characteristics of signature-based detection methods. Here are some challenges associated with using signature based CIDS in the context of blockchain activity data:

Limited Coverage: Signature-based CIDS relies on predefined signatures that represent known security threats. However, in the context of blockchain activity data, new and emerging threats may not have established signatures. This limitation leads to a reduced ability to detect novel or previously unseen attacks, as signature-based systems are typically less effective in identifying unknown threats.

Inflexibility: Signature-based CIDS requires the creation and maintenance of signatures for each specific type of attack. In the dynamic and rapidly evolving landscape of blockchain technologies, where new attack vectors and techniques emerge frequently, the process of continuously updating signatures becomes challenging and resource intensive. This inflexibility hinders the ability of signature based CIDS to adapt and respond effectively to evolving threats.

False Negatives and False Positives: Signature-based CIDS is susceptible to both false negatives and false positives. False negatives occur when an attack or suspicious activity does

not match any existing signature and goes undetected. False positives occur when legitimate blockchain activity is incorrectly flagged as malicious due to imperfect or incomplete signatures. These false alerts can lead to alert fatigue and the risk of overlooking genuine threats or wasting resources on false alarms.

Complex Data Structures: Blockchain activity data often involves intricate data structures, such as transactions, smart contracts, and addresses. These structures can be challenging to capture accurately with signatures, as they may vary significantly in format and content. Signature-based CIDS may struggle to effectively represent and match these complex data structures, potentially resulting in reduced detection accuracy and reliability.

Limited Anomaly Detection: Signature-based CIDS predominantly focuses on matching patterns against predefined signatures, neglecting the detection of anomalous behavior that does not fit established patterns. Since blockchain activity data can exhibit unique and evolving patterns, an anomaly-based approach can complement signature-based methods by identifying deviations or unusual activities that deviate from expected norms.

Privacy Concerns: Signature-based CIDS often relies on inspecting the content of transactions or specific data elements to match signatures. In the context of blockchain, privacy is a crucial concern, and revealing sensitive information within transactions or data structures can compromise user privacy. Anomaly-based approaches, which focus on behavioral analysis rather than content inspection, can offer greater privacy preservation while still detecting suspicious activities.

To overcome these limitations, integrating anomaly-based techniques alongside signature-based CIDS can enhance the detection capabilities for blockchain activity data. Anomaly-based approaches can identify previously unseen attacks, adapt to evolving threats, handle complex data structures, and provide a more comprehensive view of potential security risks without relying solely on predefined signatures.

On this thesis work, the author focuses on fulfilling the security challenges to use blockchain activity data to detect and prevent attacks that can be used for all Blockchain apparatus regarding CIDS on the bases of anomaly detection method.

1.4 Research Questions

To solve the problem stated above the research will focus on the following research questions.

RQ1: Does the integration of blockchain activity data in a collaborative-based intrusion detection system lead to enhanced accuracy and efficiency in detecting and mitigating intrusions?

RQ2: How can we increase the detection rate of anomaly-based systems, make them detect attacks and minimize false alarm rate?

RQ3: what are the most suitable machine learning algorithms and feature engineering approaches for analyzing blockchain activity data in the context of CIDS?

1.5 Objectives

1.5.1 General Objective

The general objective of the study is to develop and evaluate an effective and adaptive security model that leverages blockchain activity data within a Collaborative Intrusion Detection System (CIDS) using an anomaly-based approach.

1.5.2 Specific Objectives

To attain the general objective of the study, the following specific objectives has attempted to:

- To review previously proposed related works and there by identify different techniques datasets, and algorithms that have been used in the area of machine learning model of CIDS approaches.
- Design and develop an anomaly-based approach tailored to the specific context of blockchain activity data.
- Explore collaborative mechanisms within the CIDS model to facilitate the sharing of anomaly findings, within the blockchain network.
- Utilize real-world or simulated blockchain activity data to evaluate the performance and effectiveness of the proposed anomaly based CIDS model.
- Assess the practical implementation considerations and provide guidance for integrating the anomaly based CIDS model into existing blockchain networks.

1.6 Scope and Limitation

The study focuses on utilizing blockchain activity data, including transaction records, network interactions, and other relevant data sources, for intrusion detection purposes within the Collaborative Intrusion Detection System (CIDS).

The study specifically employs an anomaly-based approach for detecting and responding to security threats within the blockchain network. It will explore various anomaly detection techniques in machine learning algorithms tailored to the characteristics of blockchain activity data.

It also investigates the integration of collaborative mechanisms within the CIDS framework, enabling the sharing of anomaly findings, threat intelligence, and insights among multiple entities or organizations within the blockchain network.

Finally, evaluate the performance and effectiveness of the anomaly based CIDS framework using real-world or simulated blockchain activity data. The evaluation assesses key metrics such as detection accuracy, false positive and false negative rates, scalability, and computational overhead.

Limitations:

Data Availability and Access: The study's scope is limited by the availability and access to real-world blockchain activity data.

Scalability: As the blockchain network grows in size and complexity, the ability of the system to handle large volumes of data and maintain real-time detection capabilities can be a challenge.

Privacy and Confidentiality: The study should consider privacy concerns associated with the analysis of blockchain activity data. The use of sensitive or personally identifiable information within transactions or data structures should be handled with caution to preserve user privacy.

Complex Data Structures: The study will need to address the complexities of blockchain data structures, such as varying transaction formats, and network protocols. The effectiveness of anomaly detection techniques may be influenced by the ability to accurately represent and analyze these complex data structures.

Resource Constraints: The implementation and deployment of the anomaly based CIDS framework may require computational resources, including processing power and storage capacity. Resource constraints may affect the scalability and real-time capabilities of the system.

1.7 Methodology

Various research approaches were utilized in this study to achieve both the overall and targeted goals. The first step in gaining a deeper grasp of the research topic and its issue domains is to undertake an extensive review of the literature. We can determine the significance of earlier research in the field of intrusion detection by looking through this body of literature. Previous research on this topic has been evaluated to identify difficulties and provide guidance for solving them. Searching for information and various datasets to analyze trends that can distinguish assaults from routine is the second step. We must choose the proper tools, techniques, and algorithms based on the suggested solution from the identified problem in the literature review. We then build the architecture of an intrusion detection system and decide which of its components to deploy after determining those needs. Lastly, we assess the system using several metrics, such as accuracy, recall, precision, and so forth. The creation of artifacts, such as experiments, algorithms, and architecture, is the foundation of the entire technique.

1.8 Contributions of the study

Enhanced Security for IDS Networks: The research contributes to enhancing the security of networks by developing an anomaly-based approach within a Collaborative Intrusion Detection System (CIDS). This approach leverages blockchain activity data to detect and respond to security threats, providing an additional layer of protection against malicious activities.

Improved Detection Accuracy: The study aims to improve the accuracy of intrusion detection by utilizing anomaly detection techniques tailored to the unique characteristics of blockchain activity data. By focusing on anomalous patterns and behaviors, the proposed approach can identify potential threats that may go undetected by traditional signature-based methods.

Adaptation to Blockchain Context: The research contributes to adapting intrusion detection techniques to the specific context of blockchain technology. By considering the complexities of blockchain data structures, transaction formats, and network interactions, the anomaly-based approach is designed to effectively analyze and detect anomalies within blockchain activity data.

Collaboration and Threat Intelligence Sharing: The integration of collaborative mechanisms within the CIDS framework enhances the collective intelligence of the system. By enabling the sharing of anomaly findings, threat intelligence, and insights among multiple entities or organizations within the blockchain network, the research promotes collaboration, knowledge sharing, and a more comprehensive defense against security threats.

Evaluating Performance and Effectiveness: The study contributes by evaluating the performance and effectiveness of the anomaly based CIDS framework. Through rigorous evaluation using real-world or simulated blockchain activity data, the research provides insights into the system's detection accuracy, false positive and false negative rates, scalability, and computational overhead. This evaluation helps validate and refine the proposed approach.

Advancement of Intrusion Detection Techniques: The research contributes to the advancement of intrusion detection techniques specifically tailored to blockchain technology. By exploring and adapting anomaly detection methods to the unique characteristics of blockchain activity data, the study expands the knowledge base in the field of blockchain security and contributes to the development of more effective and adaptive security solutions.

1.9 Significance and beneficiaries of the study

Some of the significance of the study includes:

Enhanced Security for IDS Networks: The research holds significant importance for enhancing the security of blockchain networks. By leveraging anomaly detection techniques, the proposed approach can help detect and respond to security threats that traditional signature-based methods may miss.

Early Threat Detection and Mitigation: The anomaly-based approach enables the early detection of anomalous patterns and behaviors within blockchain activity data. This early detection allows for prompt response and mitigation of potential security threats, minimizing the impact of attacks, unauthorized access, or malicious activities within the blockchain network.

Improved Resilience and Trustworthiness: By integrating collaborative mechanisms within the CIDS framework, the research promotes the sharing of anomaly findings, threat intelligence, and insights among different entities or organizations in the blockchain network. This collaboration enhances the resilience and trustworthiness of the network by enabling a collective defense against security threats.

Adaptation to Blockchain-specific Challenges: Blockchain technology presents unique challenges for intrusion detection due to its decentralized nature, complex data structures, and diverse transaction formats. The research addresses these challenges by tailoring anomaly detection techniques to the specific characteristics of blockchain activity data, contributing to the development of more effective security solutions for blockchain networks.

Beneficiaries of the study:

IDS Network Operators: Organizations and entities operating blockchain networks can benefit from the research by gaining access to an enhanced security framework. The anomaly based CIDS, utilizing blockchain activity data, helps improve the overall security posture of blockchain networks, safeguarding valuable assets, sensitive data, and transactions.

Individuals and Organizations Utilizing Blockchain Technology: Users, developers, and organizations relying on blockchain technology for various purposes, such as financial transactions, supply chain management, or smart contracts, stand to benefit from the research. The anomaly-based approach enhances the security of blockchain applications, protecting their data, assets, and operations from potential intrusions and attacks.

Cybersecurity Professionals and Researchers: The research contributes to the advancement of intrusion detection techniques and methodologies tailored to the blockchain context. This benefits cybersecurity professionals and researchers by expanding their knowledge base, providing insights into effective anomaly detection approaches, and fostering further research in the field of blockchain security.

Regulatory Bodies and Compliance Authorities: Regulatory bodies and compliance authorities responsible for overseeing blockchain technology usage can benefit from the research. The anomaly based CIDS framework can assist in meeting regulatory requirements and ensuring the integrity, confidentiality, and availability of blockchain networks by detecting and mitigating security threats.

Overall, Trust in IDS: By enhancing the security of IDS networks and mitigating potential security risks, the research contributes to building trust in IDS by adding Blockchain technology. This can foster wider adoption of IDS applications across various industries and sectors, promoting innovation, efficiency, and transparency.

1.9 Thesis Organization

The rest of the thesis was organized in the following way. The literature review, frameworks and detail of the CIDR in Blockchain has been included in Chapter 2. After all, in Chapter 3, the research design and methodologies, chapter 4 the result and discussion of the study, chapter 5 conclusion and recommendation of the study will be incorporated.

Chapter Two

Literature Review

This chapter discusses a number of ideas that are crucial to comprehending the CIDS system that is suggested in this thesis. An introduction to an IDS and its drawbacks is given first. After that, a CIDS—a system that can correct certain flaws—is detailed. The topic of Blockchain is then brought up as a potential way to get around CIDS' implementation problems.

2.1 Overview

Network security appliances called intrusion detection systems (IDS) keep an eye out for potentially harmful behavior on networks and/or systems. It can be any hardware or software that restricts access to prevent hackers from taking advantage of computers. While some view "intrusion prevention" technology as a continuation of intrusion detection (ID) technology, it is actually an additional type of access control, similar to an application layer firewall.

A computer network or system's intrusion detection system (IDS) is a safety measure created to identify and respond to unauthorized or malicious actions. In order to spot indications of infiltration or unusual activity, it keeps an eye on network traffic, system logs, and various system operations. An IDS's main objective is to offer early detection of security problems so that quick action and mitigation can be taken [9].

IDSs are software or hardware systems that automate the process of keeping track of and analyzing network or computer system activity for indications of security issues. Intrusion detection systems are now a required component of the security infrastructure of the majority of organizations as a result of the rise in frequency and severity.

2.1.1 Detection Methods

Intrusion Detection Systems (IDS) employ various methods to detect and identify potential intrusions or malicious activities within a network or system [10]. Here are the commonly detection methods used in IDS:

2.1.1.1 Signature-based Detection:

With this technique, network activity or system events are compared to a database of recognizes attack signatures. An alert is generated if a match is discovered. A database of recognized attack signatures or patterns is the foundation of signature-based detection, sometimes referred to as rule-based or pattern matching detection. The IDS searches for matches by comparing network

traffic or system events against these signatures. If a match happens, a notice about a possible intrusion is generated [11].

Network protocols, packet contents, or behavioral patterns connected to well-known attacks can all serve as the basis for signatures. This technique works well in identifying known attacks, but it may have trouble identifying fresh or unheard-of attacks (zero-day attacks) for which there are no known signatures.

Signature-based detection is a widely used method in intrusion detection systems (IDS). It involves comparing observed network traffic or system activity against a database of known attack signatures. These attack signatures are predefined patterns or characteristics that match specific known attack types or malicious behaviors.

Here's how signature-based detection typically works:

Signature Database: The IDS maintains a database of attack signatures. These signatures are created based on the analysis of previously identified attacks and their unique characteristics.

Traffic Monitoring: The IDS continuously monitors network traffic or system activity, capturing relevant data for analysis.

Signature Matching: The captured data is compared against the attack signatures in the database. The IDS looks for an exact match or a close match to a known attack signature.

Alert Generation: If a match is found, the IDS generates an alert to notify system administrators or security personnel about the presence of a known attack. The alert typically includes details about the attack type, source, and potential impact.

Response and Mitigation: Upon receiving the alert, appropriate actions can be taken to respond to and mitigate the identified attack. This may involve blocking malicious traffic, isolating compromised systems, or initiating countermeasures.

Signature-based detection is effective in identifying known attacks and is particularly useful against well-known malware, viruses, and other common threats. It has the advantage of being precise and producing fewer false positives compared to some other detection methods. However, signature-based detection has limitations as it relies on a database of known signatures and is less effective against new or unknown attacks (zero-day exploits). It requires regular updates to the signature database to stay current with emerging threats.

2.1.1.2 Anomaly-based Detection

Establishing a baseline of typical behavior and spotting deviations from it are necessary for anomaly detection. It searches for odd behavior or patterns that might point to an intrusion. Establishing a baseline of typical behavior for the network or system under observation is necessary for anomaly-based detection. In order to find deviations from the predefined baseline, the IDS continuously monitors and examines network traffic, system logs, or other system activity. Anomalies or questionable behavior are identified when deviations exceed predetermined thresholds or statistical models [12].

Since anomaly-based detection doesn't rely on predetermined signatures, it might be effective for identifying new or emerging threats. It may nevertheless produce false positives as a result of valid differences in network or system behavior [13].

Anomaly-based detection is a method used in intrusion detection systems (IDS) to identify suspicious or malicious behavior by comparing observed activity against a baseline of normal behavior. It focuses on detecting deviations or anomalies that may indicate the presence of unauthorized or malicious activities.

Baseline Creation: The IDS establishes a baseline of normal behavior by analyzing historical data or observing network traffic and system activity during a training period. This baseline represents the expected patterns and characteristics of legitimate activity.

Traffic Monitoring: The IDS continuously monitors network traffic, system logs, or other relevant data sources, capturing information about ongoing activity.

Anomaly Detection: The captured data is compared against the established baseline of normal behavior. The IDS looks for deviations or anomalies that fall outside the expected range or patterns. These anomalies may indicate potentially malicious activities.

Alert Generation: If an anomaly is detected, the IDS generates an alert to notify system administrators or security personnel about the potential threat. The alert typically includes details about the observed anomaly, its severity, and any additional relevant information.

Response and Mitigation: Upon receiving the alert, appropriate actions can be taken to respond to and mitigate the identified threat. This may involve investigating the anomaly, blocking access to suspicious sources, or implementing additional security measures to prevent further compromise.

Anomaly-based detection is effective in detecting previously unseen attacks, including zero-day exploits or novel attack techniques. It can adapt to evolving threats since it focuses on deviations

from normal behavior rather than specific attack signatures. However, anomaly-based detection may generate false positives if the baseline of normal behavior is not accurately established or if legitimate activities are not properly characterized.

2.1.1.3 Heuristic-based Detection

Utilizing predetermined rules or algorithms to spot possibly harmful activities is known as heuristic detection. It incorporates aspects of anomaly-based detection as well as signature-based detection.

Heuristic-based detection makes use of established guidelines or formulas to spot possibly harmful activities. These guidelines are based on well-known attack methods, weaknesses in the system, or unusual behaviors [14].

When suspicious activity is identified, the IDS applies these rules to network traffic or system events and generates an alert. Heuristic-based detection may be successful in identifying new attacks or zero-day exploits, but depending on how accurately the rules are written, it may also produce false positives or false negatives.

Heuristic-based detection is a method used in intrusion detection systems (IDS) to identify potentially malicious behavior based on predefined rules or algorithms, known as heuristics. Unlike signature-based detection, which relies on specific attack signatures, heuristic-based detection focuses on general patterns or characteristics associated with malicious activity.

The working of this approach is as follows:

Heuristic Rules: The IDS uses a set of predefined rules or algorithms that are designed to identify indicators of potentially malicious behavior. These rules are created based on expert knowledge, security best practices, and common attack patterns.

Traffic Analysis: The IDS continuously monitors network traffic, system logs, or other relevant data sources to gather information about ongoing activity.

Heuristic Matching: The captured data is analyzed against the set of heuristic rules. The IDS looks for patterns or behaviors that match the predefined rules, indicating potential malicious activity. For example, the IDS might detect multiple failed logins attempts within a short time period, which could indicate a brute-force attack.

Alert Generation: If the observed behavior matches a heuristic rule, the IDS generates an alert to notify system administrators or security personnel about the potential threat. The alert typically includes details about suspicious behavior, its severity, and any additional relevant information.

Response and Mitigation: Upon receiving the alert, appropriate actions can be taken to respond to and mitigate the identified threat. This may involve blocking or limiting access to the source of the suspicious behavior, implementing additional security measures, or investigating further to gather more information.

2.1.1.4 Statistical Analysis:

Statistical analysis uses statistical models and algorithms to track and examine system events or network traffic.

The IDS searches for trends, patterns, or statistical outliers that might point to an intrusion or malicious activity [15]. Statistical analysis can be used to spot anomalous resource utilization, traffic patterns, and other system or network behavior.

2.1.1.5 Machine Learning (ML) and Artificial Intelligence (AI):

IDS detection capabilities can be improved by implementing ML and AI approaches. With these methods, models are trained on sizable datasets in order to identify typical and anomalous network or system behavior. The classification of network traffic or system events as legitimate or possibly malicious can then be done using ML models. The detection accuracy of ML-based IDS can be improved over time [16] and it can adapt to new attack patterns. However, ML models need to be updated frequently and could be vulnerable to evasion tactics used by attackers.

2.1.2 Network-based IDS (NIDS):

NIDS keeps track of network activity and instantly examines packets. It can recognize a range of network-based assaults, including port scans, denial-of-service attacks, and intrusion attempts.

To monitor traffic, NIDS can be installed at strategic locations inside a network, like the network perimeter or internal network segments. Network-based Intrusion Detection Systems (NIDS) are security tools that keep an eye on network activity and examine it for indications of harmful or unauthorized activity. In order to find potential invasions or security breaches, NIDS functions at the network level, looking at packets that are passing over the network [17].

A network-based intrusion detection system (NIDS) can identify malicious network traffic. For NIDS to analyze all data, including all uni-cast traffic, promiscuous network access is typically required. The basic NIDS architecture is depicted in Fig. 1. NIDS are passive devices that do not obstruct the traffic they monitor. In read-only mode, the NIDS sniffs the firewall's internal interface and delivers notifications to a NIDS Management server over a separate (read/write) network interface [18].

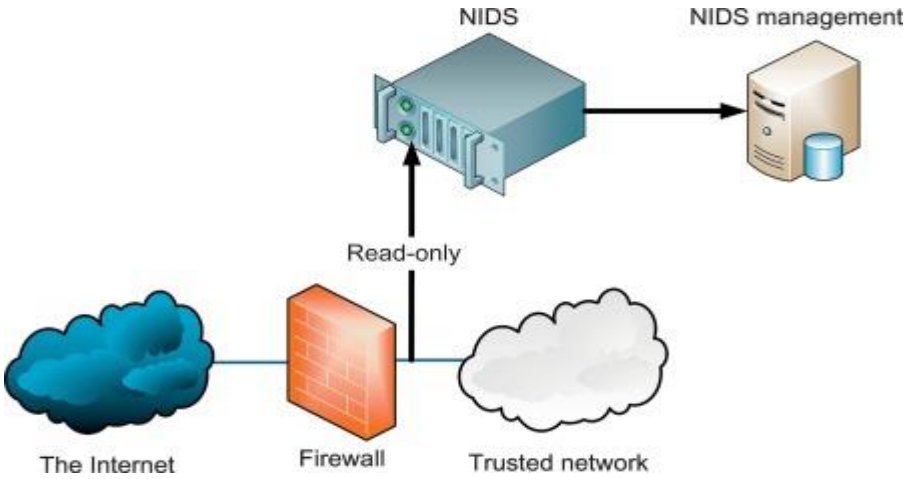


Figure 1: Typical NIDS architecture [18]

A NIDS scans network-related data for irregularities. This information includes, for instance, network packet flow and understanding of the various network protocols [19]. Figure 2 displays a typical NIDS architecture.

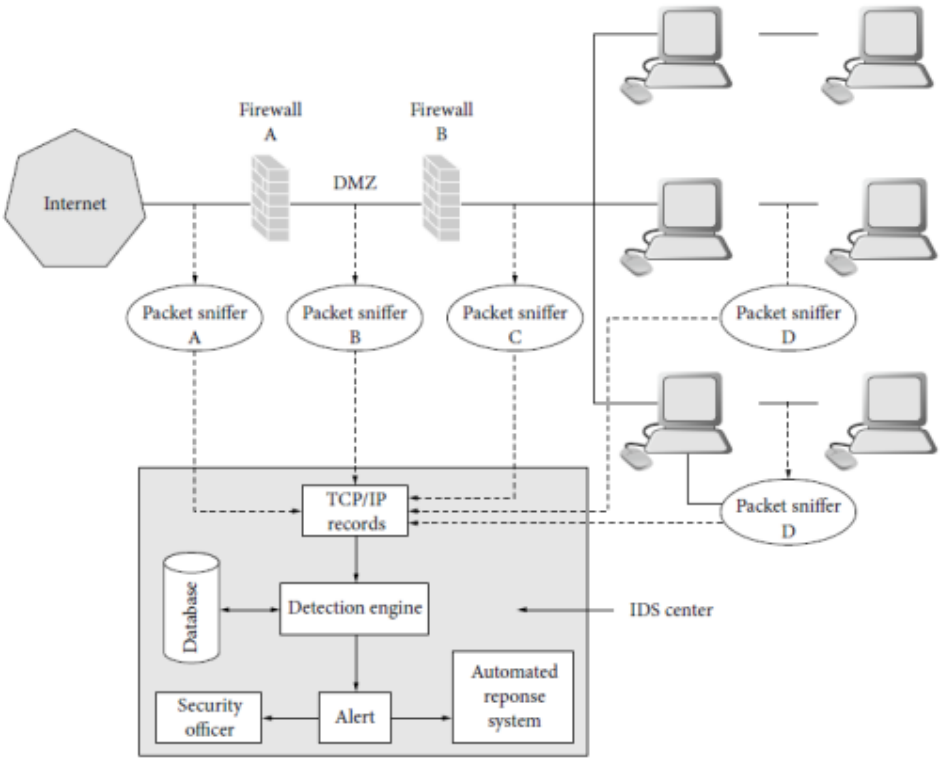


Figure 2: Standard NIDS architecture

2.1.3 Host-based IDS (HIDS):

Specific hosts or systems are the focus of HIDS. It keeps an eye on host-specific operations, file integrity, system calls, and system logs. Security tools called host-based intrusion detection systems (HIDS) keep an eye on specific hosts or systems for indications of unauthorized or malicious activity. HIDS operate directly on individual hosts, analyzing system logs, file integrity, user actions, and other host-specific events to detect potential intrusions. This is in contrast to network-based IDS, which concentrate on network traffic.

On a specific host, HIDS can identify attempts at unauthorized access, file alterations, odd processes, and other suspicious activity.

A HIDS uses information about the host and its computing activity to ascertain whether fraud is occurring.

Table 1: selection of host metrics.

Metric	**Description**
CPU Usage	High CPU Usage Could Be an Attacker's Sign or a Sign of a Poor Process
Memory Access	An attacker may be indicated by an odd memory access or improper timing.
File Access	Strange File Access Attempts Could Indicate an Attacker
Login Attempts/Times	High Numbers of Failed Login Attempts or Login Attempts at Strange Times could Indicate an Attacker

Both knowledge-based and behaviorally based HIDS are possible. Comparable to commercial virus-checking software is the knowledge-based case [19]. A virus-checking program compares features on a host computer to the signature of a virus or piece of malware. The software issues an alert if the signature matches. In the instance based on behavior, a profile of typical computer usage is created. An alarm is issued if the current metrics measurements deviate from the anticipated profile [19]. For instance, if a system anticipates login attempts to happen between 0800 and 1700, but one happens at 0200, an alarm should be issued.

2.1.4 Intrusion Prevention Systems (IPS)

Intrusion Prevention Systems (IPS) are security mechanisms that work in conjunction with Intrusion Detection Systems (IDS) to not only detect potential intrusions but also actively prevent or block them [20]. Some IDS solutions include IPS functionality, allowing them to take

automated actions to block or mitigate detected threats. IPS does more than simply notify users of intrusions; it also acts right away to eliminate threats.

2.1.5 IDS: The Scaling Issue

IDSs are frequently used as defensive solutions; however, they have limitations [21]. IDSs are often standalone. They keep an eye out for unusual activity on a single host or network [21]. When an attack spreads over multiple hosts or networks, a problem occurs. Such an attack cannot be fully detected or countered by a collection of standalone IDSs that do not interact or communicate with one another [21]. In particular, it is not possible to connect harmful actions that happen simultaneously across hosts in a network or across networks in a system [21]. The idea of CIDS was presented in an attempt to remedy this flaw.

2.1.6. Intrusion Detection Systems (IDSs) Technologies

As was said previously, IDSs concentrate on finding potential events. An intrusion detection system could identify instances in which an attacker has successfully gained access to a system by taking advantage of a vulnerability in the system. After the incident was reported by the IDS, security administrators could promptly take incident response measures to reduce the incident's negative effects. Information that the incident handlers could utilize could also be logged by the IDS. It is also possible for many DSs to be set up to identify security policy infractions. A few IDSs, for instance, have the ability to be configured with firewall rule-like settings, which enables them to recognize network traffic that deviates from the security or permissible use regulations of the company. Certain intrusion detection systems (IDSs) can also keep an eye on file transfers and spot any that seem odd, such moving a sizable database onto a user's laptop. Reconnaissance activity is another feature that many IDSs can detect and may be a sign of impending assault. To discover targets for follow-up attacks, certain attack tools and malware, especially worms, engage in reconnaissance tasks like host and port scans. Perhaps an intrusion detection system (IDS) might stop reconnaissance and alert security managers. Then, if necessary, those administrators could adjust other security measures to stop related occurrences. Protected internal networks are often the primary target for reconnaissance detection due to the high frequency of reconnaissance activity on the Internet. Organizations have discovered other uses for intrusion detection systems (IDSs) beyond detecting occurrences and assisting with incident response activities. These include the following:

Finding issues with security policy implementation: An intrusion detection system (IDS) can offer some quality control for the implementation of security policies by, for example, replicating

firewall rule sets and raising an alarm when it observes network traffic that the firewall should have blocked but did not because of a configuration error. IDSs keep a log of the dangers they identify in order to document the current threat facing an organization. Determining the right security measures to safeguard an organization's computing resources requires an understanding of the types and frequency of attacks against such resources. The information can also be used to educate management about the threats that the organization faces. The data can also be utilized to inform management of the dangers the company confronts. Preventing someone from breaking security regulations: People may be less inclined to break security rules if they are aware that IDS technologies are keeping an eye on their activities and will detect any infractions. IDSs are now an essential component of almost every organization's security infrastructure due to the growing reliance on information systems, as well as the frequency and possible consequences of attacks against them.

2.1.7. Key Functions of IDS Technologies

IDS systems come in a variety of forms, mainly distinguished by the kinds of events they can identify and the approaches they take to pinpoint occurrences. Apart from tracking and evaluating events to spot unwanted behavior, all kinds of intrusion detection systems generally carry out the following tasks [10].

- *logging data pertaining to incidents that have been witnessed:* Typically, data is stored locally, but it may also be transferred to other systems, including corporate management systems, centralized logging servers, and security information and event management (SIEM) programs.
- *Alerting the security administrators to noteworthy events that are observed:* There are various ways to get this alert, which is also called a notification: user-defined programs and scripts, emails, websites, messages on the IDS user interface, and Simple Network Management Protocol (SNMP) traps. Typically, a notification message only contains the most basic information about an occurrence; administrators must consult the IDS to obtain further details.
- *Preparing reports:* Reports offer an overview of the events under observation or specific information on noteworthy occurrences. When a new danger is discovered, certain IDSs have the capability to alter their security profile as well. For instance, if harmful activity is found during a session, an IDS may be able to gather more specific data for that session. When a specific threat is detected, an

IDS may also change the parameters for when specific alerts are issued or what priority should be given to subsequent warnings.

One feature sets IPS technologies apart from IDS technologies: they can react to a threat by trying to stop it before it has a chance to succeed. The assault is halted by the IPS itself. Here are some examples of possible ways to accomplish this [11]:

I. Cut off the user session or network connection that is being leveraged for the attack.

II. Prevent access from the offending user account, IP address, or other attacker attribute to the target (or perhaps other likely targets).

III. Prevent any access to the host, service, application, or other resource that is being targeted.

IV. The security landscape is altered by the IPS. To stop an attack, the IPS could alter how other security measures are configured. Reconfiguring a network device (such as a firewall, router, or switch) to prevent access from the attacker or to the target, as well as modifying a host-based firewall on a target to prevent inbound attacks, are frequent instances. If an intrusion prevention system (IPS) finds that a host has vulnerabilities, it may even force the application of updates. The attack's content is modified by the IPS. Certain intrusion prevention systems can neutralize an assault by removing or swapping out its harmful elements. An IPS might, for instance, remove an infected file attachment from an email and then let the cleaned email to be delivered to the intended recipient. An IPS that functions as a proxy and normalizes incoming requests—that is, repackages the payloads of the requests while discarding header information—is a more complicated example. As a result, some attacks may be dropped throughout the normalization phase.

As was previously mentioned, one further characteristic of IDS solutions is their inability to offer 100% accurate detection. A false positive occurs when an IDS mistakenly labels legitimate activity as harmful. A false negative happens when an intrusion detection system is unable to detect malicious activities. False positives and negatives cannot be completely eliminated; generally speaking, decreasing the incidence of one causes an increase in the incidence of the other. In other words, more dangerous events are discovered, but more analysis resources are required to distinguish between false positives and actual malicious events because many businesses opt to reduce false negatives at the expense of raising false positives. Tuning an IDS means changing its settings to increase detection accuracy.

Additionally, the majority of IDS technologies provide features that offset the use of popular evasion methods [10]. Evasion is the process of changing the time or format of malicious activity so that, although changing in appearance, the outcome remains the same. Attackers attempt to evade detection by IDS systems by employing evasion methods. An attacker may, for instance, encode text characters in a specific way, hoping that any IDSs that are monitoring them do not recognize the encoding, knowing full well that the target does. Common evasion strategies can be circumvented by most IDS solutions by copying the special processing that the targets execute. Evasion strategies typically fail to conceal assaults if the IDS is able to "see" the activity in the same way as the target.

2.1.8. Components of Intrusion Detection/Prevention System

An intrusion detection system is made up of various parts, all of which function in tandem with one another. This section discusses the components and their individual functions.

Sensor: To gather data, they are put on devices (since they are devices) that are situated inside the network. They collect data from a variety of sources, including log files, system call traces, and packets from network activity. After that, the gathered data is sorted and sent to other analyzers. Two categories of sensors exist. Sensors based on networks. Sensors based on hosts Increasingly, host-based network packets are being replaced with network-based sensors. One sensor can record and manage all inbound and outgoing traffic. In the event that two interfaces—one for management and one for monitoring—are employed, network sensors do not add to the network's traffic.

Response module: It manages the sensors and keeps an eye on events and warnings. Sensor data will be gathered by it. Alerts are generated by comparing this data to IDS behavior models. Higher level monitors may receive these alerts. After that, unfriendly reports are created, and a course of action is decided. There are three parts to a response module [12]. Interface for communication, Recipient Transmitter Interaction with other IDS components is facilitated by a communication interface. A listener watches for data and information to be sent to it by sensors and other agents. The data is then sent by a sender to the manager devices and other agents. Agents are also capable of performing correlation analysis on received data and creating alarms, among other tasks.

Manager: Fundamentally, the manager's purpose is to provide the ID master control. The following features are available from the manager component [12]. Data management, alerting, event correlation, high-level analysis, and component monitoring

Event generator: The end user's view of how to interact with the system is provided by the user interface in IDS. The system can only be configured or controlled by the user via the interface. The ability of the user interface to generate reports is another crucial feature.

2.2 Collaborative Intrusion Detection Systems

The inability of IDSs to thwart distributed or parallel attacks is addressed by CIDSs. A series of monitor units, which serve as sensors, and a set of analysis units, which interpret the sensor data, make up CIDSs in most cases [21]. Co-locating these units is possible. A CIDS can gather information from several hosts or networks in this manner in order to draw conclusions about potential intrusions or anomalies [22].

2.2.1 Types of CIDS

The blocks with the letters "M" and "A" indicate whether the node is a monitor unit or an analysis unit, or sometimes both, depending on the situation. Figure 3 depicts the designs of the three major kinds of CIDS.

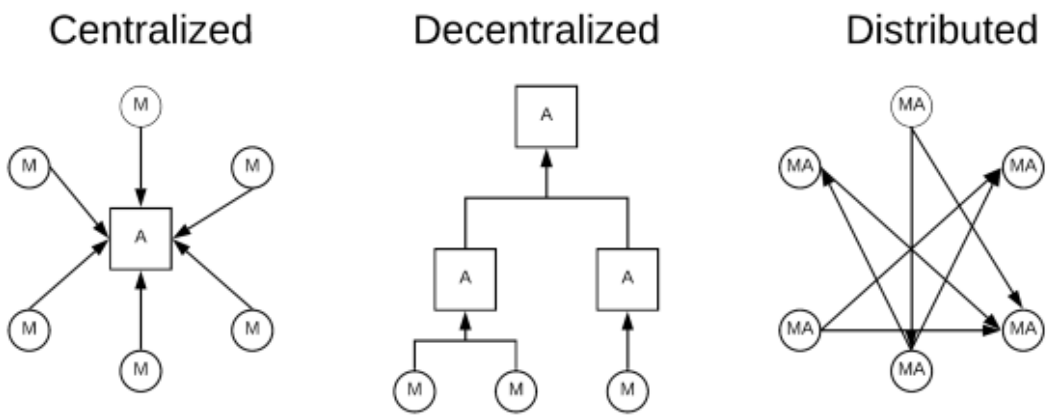


Figure 3: Overview of CIDS Architectures. Adapted from [21].

The simplest solution to the issue of scattered attacks is centralized CIDS. According to this approach, a single central analysis unit receives data from various monitor units, including host-based alarm data and network-based traffic data [21]. The analysis unit applies either standard detection algorithms to network data or alert correlation techniques to host-based data. This strategy does have flaws, which can make it undesirable in a number of situations. With growing networks, centralized CIDS scale is very poor, and the central analysis unit acts as a single point of failure (SPoF) and performance bottleneck [23].

A hierarchical structure of analysis and monitoring units is used by a decentralized CIDS. This topology benefits from not having the SPoF problem associated with centralized CIDS and is

expandable to larger systems. Additionally, because analysis units process data at every level of the hierarchy, performance at the top of the hierarchy is enhanced. The top analytical unit will have less work to do as a result [21]. This level-by-level strategy does have a price. Prior to reaching the top of the hierarchy, information is lost at each step of aggregation. This may result in the loss of vital information necessary for the detection of assaults. Additionally, as [23] points out, contemporary decentralized CIDS solutions still suffer SPoFs and bottlenecks.

A distributed CIDS distributes the analysis tasks among all monitors rather than using a single analysis unit. To ensure that all data is distributed, shared, aggregated, and correlated throughout the system, a peer-to-peer (P2P) architecture is required. This strategy prevents SPoFs and is scalable. However, because of additional signaling overhead, a distributed CIDS does result in higher network costs [21]. Finally, such a system needs to include procedures to ensure confidence among its nodes, as noted by [24].

2.2.2 CIDS: The Trust and Consensus Issue

While a distributed CIDS technique appears to be effective at fixing an IDS's flaws, several aspects of its implementation are challenging to get through. In a CIDS, the concept of trust is essential. A monitor starts to spread misleading information, according to [24]. The system must be able to decide whether or not to trust a monitor and whether the data it produces should be accepted. In other words, the system needs to agree on all alert data as well as the reliability of every node [24]. Block-chain is suggested as a remedy for this trust issue.

2.3 Blockchain

Cyber security is one of many businesses investigating how Blockchain technology might enhance its operations. Blockchain's characteristics are helpful in the context of CIDS. The technique by which Blockchain technology validates and stores data without the requirement for a central, trusted authority is just what makes it essential for CIDS applications. An overview of the beneficial characteristics of Blockchains is given in the following sections [25].

2.3.1 Categories of Blockchain

Public, consortium, and private Blockchain ledgers are the three types now in use [26]. Anyone with internet access and the desire to engage is able to do so using public Blockchain networks. Blockchain systems that are managed by a consortium are accessible to others. Private Blockchain systems are managed by a single body that grants access to other parties.

2.3.2 Block Structure

A Blockchain, in its most basic form, is a series of interconnected blocks, where each block is joined to the one before it and the one after it by a mathematical relationship. A block is only a place where bits of data are stored. Each block in a Blockchain contains a distinct self-identifying hash to maintain chain integrity, which is the main concept of Blockchains. This self-identifying hash is made up of the hashes of the data, timestamp, block index, and, of course, the preceding block hash [27]. Figure 4 depicts a condensed version of a Blockchain.

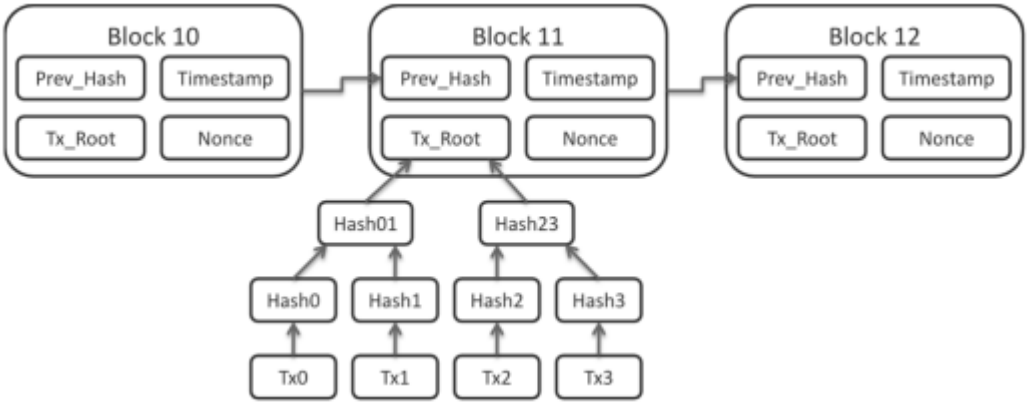


Figure 4: Block chain structure [28]

Each block also includes a ledger, or log, of all transactions that took place during block production. This block structure serves to codify each transaction [27]. Every transaction that took place prior to the creation of the current block is documented since every block reference to the one before it. Why changing a block is mathematically impractical is covered in Section 2.3.3 [27].

2.3.3 Consensus

Blockchain network participants can agree on the state of the network without the need for a centrally located trusted authority thanks to consensus algorithms [27]. Any block-chain-based system is only as efficient as its consensus model [29]. In [29], it also Provide more details on the history and characteristics of the consensus dilemma. The proof-of-work algorithm, which Bitcoin uses, is the most widely used illustration of a consensus algorithm in the block-chain world [27].

The concept behind proof-of-work is that a participant verifies its identity by offering some evidence that it completed work. In the case of Bitcoin, each participant aims to find a hash value that is lower than a threshold set by the network as the difficulty level [29]. This is an illustration of a computational challenge where a brute force guess-and-check strategy is the most effective

way to solve the puzzle [30]. The next block cannot be produced by a single participant with an advantage thanks to the mining process [27]. Participant authentication and prior knowledge are thus not necessary.

Because a node would have to recreate the entire chain before producing a new block, this technique also makes it theoretically impossible to change a block or set of transactions. According to Nakamoto [27], the likelihood of a successful alteration declines exponentially as the size of the block-chain increases. Nevertheless, the 51% assault, when one coalition controls more than half of the available mining power, makes proof-of-work vulnerable [30], allows one coalition to add blocks to the blockchain. Ethereum, a different cryptocurrency company, adopted the proof-of-stake consensus process to counter this.

In proof-of-stake, the next block is proposed and voted on by a group of validators who have an economic stake in the network [20]. By choosing validators for block production in a pseudo-random manner, the method prevents participants from knowing in advance when they will generate blocks. The participant's cryptocurrency holdings, or stake, determine their chances of getting chosen as a validator [30] [31]. The Nothing-at-Stake problem [20] is an example of a potential issue with this approach; however, Ethereum is actively working on a mechanism to prevent the 51% assault.

2.3.4 Transactions

Data is simply transferred from one party to another during a transaction. Anything can be this info. These data, in the context of cryptocurrencies, would be financial or contractual information [27] [32]. This information may be patient medical records or a transfer of medical equipment in the medical profession [33]. Data is versatile; therefore, Blockchains can be applied in many different industries. A transaction in the context of CIDS could include alert data [24]. Every transaction that has occurred is documented in the Blockchain ledger. Figure 5 shows the general strategy for a cryptocurrency transaction.

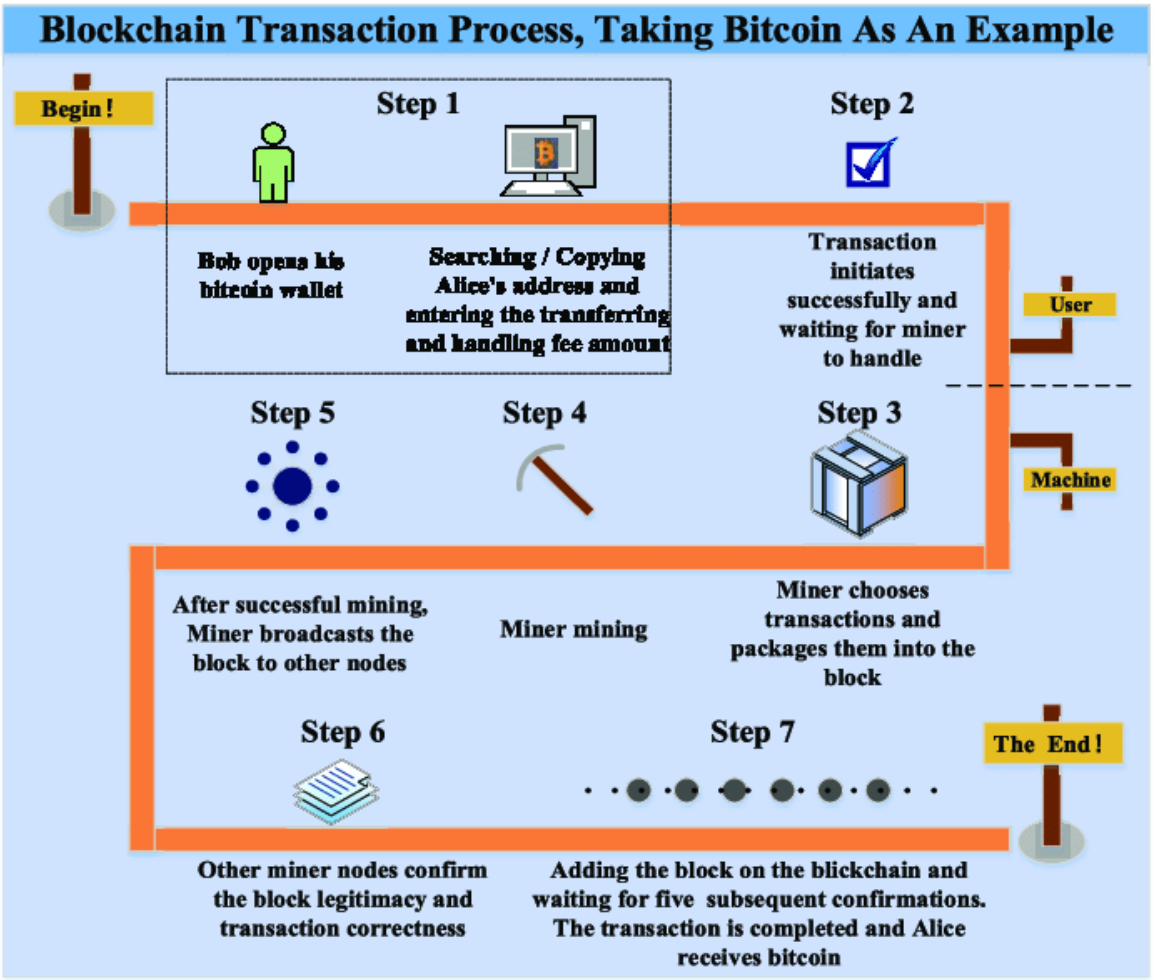


Figure 5: Block chain Transaction Process [34].

Two crucial elements are illustrated by the procedure represented in Figure 5. Each transaction is permanent and transparent, and each participant owns a copy of the ledger of all transactions [27]. Each block is also decided upon using a consensus mechanism.

This provides a trust framework for all participants in the network. The process workflow for Bitcoin is presented as Algorithm 1.

Result: All Transactions Codified

1) All nodes receive broadcasts of new transactions.

2) Every node builds a block out of fresh transactions.

3) Each node strives to come up with a challenging proof-of-work for its block.

4) A node broadcasts the block to all other nodes after locating a proof-of-work.

5. Nodes will only accept a block if all of its transactions are genuine and have not yet been spent.

Algorithm 1: Bitcoin Transaction Codification Workflow [27].

This process flow facilitates the trust framework for all participants in the network. The ability to ensure that all participants have knowledge of all transactions and have a stake in approving those transactions is critical to using Blockchain for a CIDS.

2.4 Related works

The following is a table listing similar works on Blockchain Activity Data for Use in Collaborative Intrusion Detection System, along with information on the methods and algorithms used, IDs type and metrics and performances.

Table 2: Comparison of Related Works

Ref.	Method Used	Algorithms Used	IDS Type	Metrics & Performance
[35]	Collaborated signature-based IDS.	Collaborative Intrusion Detection Networks	Signature-based intrusion detection	Simulated CIDN, a practical CIDN (66.7%)
[36]	collaborated signature-based IDS.	Collaborative Intrusion Detection Networks	Snort Based Collaborative Intrusion Detection System	SDN based environment evaluation (96%)
[37]	combination of a proof-of-stake and proof-of-work protocols	Collaborative Intrusion Detection Networks	trust-based blockchain	Trustworthiness, Detection Accuracy
[38]	security-based distributed intrusion detection, privacy-based blockchain with smart contracts	bidirectional long short-term memory (BiLSTM), DNN	CIDS	Accuracy (99.41%), Detection Rate, and processing time
[39]	hierarchical off-line	Distributed Time-Delay Artificial Neural Network	Anomaly network intrusion detection system	Accuracy 97.24%

The research summarized in Table 2 exhibits several significant research gaps. The first study does not address the methods to construct a more effective and robust collaborative anomaly-based IDS. Instead, it solely incorporates a proof-of-concept blockchain in its current work, leaving room for further exploration in improving the system's performance. The second study neglects the consideration of scalability and distribution, which are pivotal aspects for real-world implementation. The absence of simulation and privacy-related concerns weakens the practicality

and security evaluation of third system. Fourth study, the research overlooks the evaluation of scalability and utility using non-real-world datasets, limiting the generalizability of its findings. Lastly, the scalability and performance analysis lack assessment under high traffic volume and large-scale environments, which are crucial factors for determining its effectiveness in real-world scenarios. Addressing these research gaps would enhance the development of a more effective, scalable and practical anomaly network intrusion detection system based on the DTDNN approach.

Chapter Three

The Proposed System Model

The use of various Blockchain machine learning models to solve issues and make data-driven decisions has emerged as the most crucial agenda item. Three groups comprise the most popular methods for creating machine learning models. In the first, unsupervised as well as supervised learning is the only machine learning algorithm that is used. The second strategy is hybrid in nature, drawing on both supervised and unsupervised learning algorithms. The goal of this strategy is to have both algorithms work in concert with one another to enhance the model's performance on a given task. The third method is known as ensemble learning, and it involves using a variety of ML algorithms-either supervised or unsupervised learning algorithms-to construct an ML model.

This paper proposes a Blockchain machine learning model that combines supervised and unsupervised learning techniques. Additionally, feature selection approaches are used to improve the ML models' performance. This strategy aims to build a hybrid machine learning model that maximizes intrusion detection in computer networks by using supervised learning to detect known attacks and unsupervised learning to detect unknown attacks. The input features will be clustered into normal and attack groups by the K-means technique. The classifier algorithm will take advantage of the clustered input feature. The classifier method will use the clustered value as an input feature, enabling the classifier to identify additional abnormalities.

3.1 System Model

The system model, which represents the entire process of converting low-level data into high-level knowledge, formed the basis for this study and was implemented in the Python programming language. The process of building the models is iterative and experimental, allowing the models that best adapt to the situation to be selected. Preprocessing, train-test split, validation split, training with default parameter, and grid search phases are all included in the system design. The preprocessing stage, which is divided into smaller tasks, includes feature selection, normalization, vector assembling, and feature selection to select pertinent features, one hot encoding to convert string indices into 0s and 1s, and string indexing to convert string data to numeric. The entire preprocessed data set is divided into training and testing sets during the train-test split phase for training machine learning algorithms and evaluating models’ performance.

The training set is then divided into the real training and validation sets during the validation split step. During the training phase with default parameters, the machine learning algorithm generates a trained model by utilizing training data with default settings. Evaluation result 1 is produced after the trained model's performance is assessed using distinct preprocessed testing data. The models are then assessed using the validation set. The final trained model is constructed using the best parameters that were chosen, and it is subsequently assessed on the testing set.

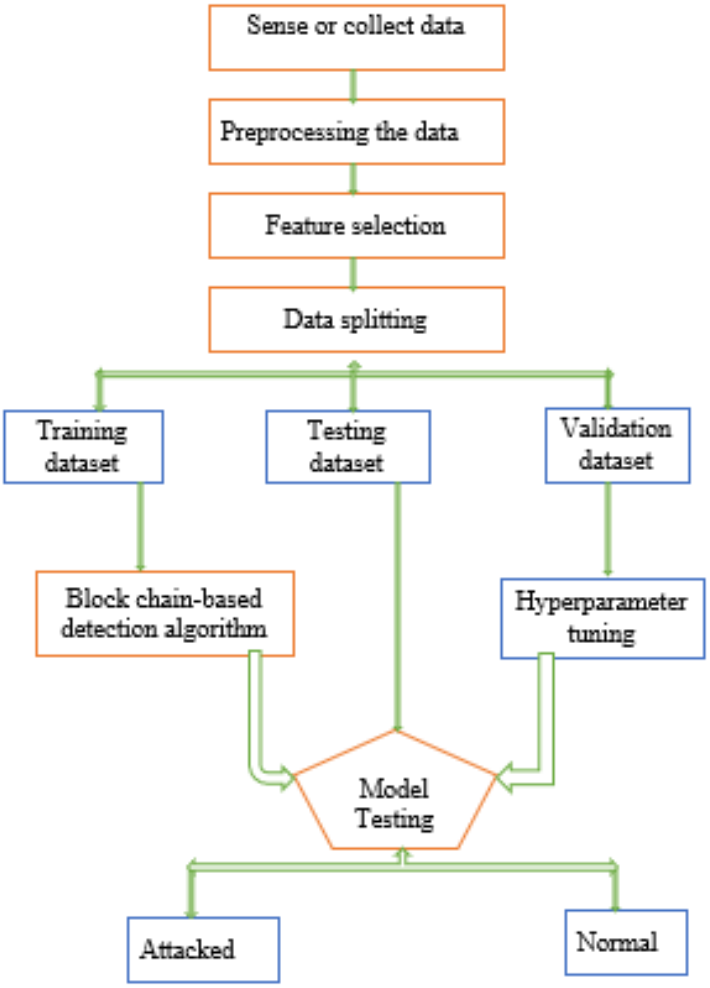


Figure 6: Proposed System Model

3.1.1 Data collection and preparation

A vast amount of network traffic data is required to create an intrusion classification model in order to give the classifiers adequate information to train effectively. Because of this, a network intrusion detection dataset that is accessible to the public is gathered from the Internet. Not all of

the publicly accessible datasets for network intrusion simulation are useful for creating efficient intrusion detection systems (IDSs). The KDD CUP 99 dataset, for instance, is the most popular and publicly available network intrusion detection evaluation dataset and may be accessed at [40]. However, utilizing this dataset will lower the quality of system performance because it contains redundant and unnecessary records in the training and testing set. This is the dataset's main weakness. The first 21,699 records are collected for semi-supervised modelling. 21,699 records are used for this study after preprocessing. Fifty-five percent of these records have no labels, and the remaining forty-five percent have labels. For the semi-supervised modelling of this study, the initial distribution of records in the Kddcup.data-10-perecnt, which was previously stated in this section, determined the distribution of labelled and unlabeled records.

Table 3 displays the distributions of the original data, which comprise both labelled and unlabeled data:

Table 3: The Original Dataset's Distribution

Dataset Label Number of Records Collected
Normal 4,646
Attacks 5,016
Unlabeled 11,926
Total 21,699

The process of data mining begins with the removal of duplicated and irrelevant data. A class's proportion of data to its size is included in the data creation percentage. Just 21,533 datasets remain after superfluous and irrelevant data have been removed in order to carry out this investigation.

3.1.1.1 Data distribution techniques

Different data processing is done while the system model is implemented. The data splitting was 80 % for training data, 10% for testing data and 10% for validation of the actual collected data out of the unbalanced data, after the following data distribution has compared using stratified 5-fold technique as follows.

Table 4: Distribution of gathered records according to dataset label.

Data Distribution Based on Training, Validation, and Testing	Training Dataset	Testing Dataset	Validation Dataset
(50%, 5%, 45%)	(50%)	(5%)	(45%)
(70%, 10%, 30%)	(70%)	(10%)	(30%)
(60%, 20%, 20%)	(60%)	(20%)	(20%)
(80%, 10%, 10%)	(80%)	(10%)	(10%)
(70%, 10%, 20%)	(70%)	(10%)	(20%)

To show the difference on the accuracy during different data distribution ratio between training dataset, testing dataset and validation dataset during training, is evaluated as described above.

3.1.2 Data preprocessing

The primary necessity for data preprocessing stems from the possibility that superfluous data and unimportant aspects could be confusing to the classification algorithm, which could result in erroneous and less broadly applicable models. The data gathered from various sources is typically too big for a given task, contains erroneous and/or missing values, and is not formatted properly for a computer. Prior to being fed into an ML algorithm, machines require certain representations of the input data, known as features, for both testing and training. Preprocessing operations such string indexing, one-hot encoding, feature selection, normalization, and vector assembly are carried out on the training and testing data in order to prepare the data in that format.

3.1.2.1 Normalization and Data Cleaning

Features that are roughly regularly distributed or on a comparable scale help many machine learning algorithms operate more quickly or efficiently. Numerous characteristics with values ranging across multiple ranges may be present in a dataset. Large integer numbers may make up some characteristics, while small floating-point values may make up others. Furthermore, some machine learning techniques, such neural networks, are not able to handle huge numbers—that is, values greater than the starting values that the network's weights took. Consequently, in order to facilitate network learning, the algorithm's input data should contain tiny values (between 0 and 1), all features should have values within the same range, and each feature should be separately normalized to have a mean of 0 and a standard deviation of 1.

Furthermore, certain missing values may be present in the input data that a machine learning system receives. Both during training and testing, the classifier may become confused by these

missing data. Missing input values are an issue for most machine learning methods. Therefore, it is preferable to substitute some suitable values for each missing value. For example, it is safe to enter missing values as 0 when using neural network techniques. The network will learn via data exposure that the value 0 indicates missing data and will begin to ignore the value.

3.1.2.2 Data splitting

Any model building process's primary goal is to create a generalizable model using the data that is already available that will function well on data that is not yet known. However, we require a distinct set of unseen data in order to evaluate a model's performance on unseen data. We accomplish this by dividing the data we have on hand into training and testing sets. A classifier uses the training set to learn how each category looks by predicting things based on the input data and then self-correcting when its predictions turn out to be incorrect. The testing set is used to determine the model's accuracy, or predictive capability.

It is only used to compare the results produced by the final model with the actual labels of the dataset. It is important that the training set and testing set are independent of each other and do not overlap. This is because if we use the testing set as part of our training data, then the classifier's generalizability will be low since it has already seen the testing examples before and learned from them. We have to keep this testing set separate from the training process and use it only to evaluate the model. The size of the split can depend on the size and specifics of the dataset. Out of the preprocessed dataset, 80% is used for training and the remaining 20% equally is used for testing and validation.

While creating models by dividing data into training and testing sets is effective, other datasets must be taken into account while fine-tuning models. This dataset, also known as the development set or validation set, aids in the model's validation. The validation set provides us with an approximation of the model's performance when it comes to fine-tuning the hyperparameters. It is a sample of data from the accessible dataset that is not present in the training data. Training and testing sets have already been created from the dataset. The data set was then further divided into a validation set and a testing set of 10% and 10%, respectively.

3.1.3 Model Performance Evaluation

Testing, validation, and training phases are the three stages a model can assess. The test performance reveals the model's real-world effectiveness. Classification metrics are typically used to assess a model on a dataset. Five assessment metrics—the confusion matrix, accuracy,

precision, recall, and f1-score—are utilized to assess the performance of the models in this study based on the literature review.

3.1.4 Activity data selection for CIDS

- Blockchain-based IDS: This approach utilizes blockchain technology to create a decentralized and tamper-resistant system for intrusion detection. It ensures transparency, immutability, and increased resilience against attacks. Smart contracts can automate intrusion response actions.
- Blockchain-based CIDS: Collaborative Intrusion Detection Systems leverage blockchain to facilitate collaboration and information sharing among multiple entities. It forms a collective knowledge base, enhancing detection by leveraging combined resources and knowledge.
- Normal data-based IDS: Traditional IDS analyzes network or system data in real-time to identify threats. It uses rule-based or anomaly-based detection techniques, but may have limitations in scalability, accuracy, and detecting sophisticated attacks.
- Normal data based CIDS: Similar to normal IDS, CIDS emphasizes collaboration and information sharing. It enables entities to exchange data, share analysis results, and collaborate on response actions. This improves accuracy and effectiveness.

Blockchain-based IDS and CIDS offer advantages in terms of decentralization, transparency, and immutability. They provide a tamper-resistant environment and reduce reliance on a central authority. Additionally, smart contracts can automate response actions, enhancing efficiency.

Normal data-based IDS and CIDS are more traditional approaches. While they can be effective, they may have limitations in scalability, accuracy, and the ability to detect sophisticated attacks. However, they are generally more accessible and easier to implement compared to blockchain-based solutions.

Both blockchain-based and normal data-based CIDS emphasize collaboration and information sharing. However, blockchain-based CIDSs leverage blockchain technology to ensure trust, accountability, and transparency among participants.

3.2 Summary

The practice of identifying improper, inaccurate, or unusual activities directed towards computer systems and networking devices is known as intrusion detection. Using the Apache Spark big data processing platform, we suggest a network IDS that combines machine learning approaches with anomaly-based approach. With the obtained dataset, the proposed system can detect intrusions effectively. Using data preprocessing techniques, the dataset is preprocessed before being categorized using the trained models. The dataset is subjected to feature selection and hyperparameter tuning methods, which allow the system to process information more quickly and accurately. The grid search phase is in charge of validating the models' efficacy using validation data—data that was not used during the training phase—by demonstrating that the parameters used in the training phase perform better based on the evaluation metrics. If the results are not satisfactory, the model can be retrained to obtain better hyperparameter values, which are then used to construct the final model.

Lastly, fresh cases are predicted or categorized into their respective categories using the final model. Based on the findings, a choice regarding the system's efficacy and efficiency can be made.

In summary, blockchain-based IDS and CIDS leverage blockchain technology to enhance the security, transparency, and collaboration aspects of intrusion detection. Normal data-based IDS and CIDS refer to traditional intrusion detection systems that rely on analyzing network or system data. Each approach has its own advantages and considerations, and the choice depends on the specific requirements, resources, and context of the deployment.

Chapter Four

Experimentation and Results

The experiments conducted to evaluate the suggested system's feasibility are covered in this chapter. Discussion topics include the development environment and software tools, programming languages and libraries used in the classification process, the dataset utilized implementation specifics, and the outcomes of the various models' performance assessments using different metrics of evaluation.

4.1 Development Environment and Tools

A number of instruments and methods are applied in order to put the suggested system into practice. Every trial for putting our suggested system into practice was run on a laptop outfitted as follows: With an Intel(R) Core i3 CPU running at 2.40GHz and 8 GB of RAM, window 10 is the operating system platform. The suggested system is put into practice using the following instruments.

Apache Spark: The suggested solution is implemented using Apache Spark due to its quickly distributed engine for ML and large-scale data processing applications. An open-source cluster computing platform called Apache Spark is designed to process data quickly [41]. 2009 saw its development at UC Berkeley. Spark offers in-memory storage and processing of data. Apache Spark processes tasks 100 times quicker than Hadoop MapReduce, another big data processing tool [42]. One of the key ideas behind Spark is that it offers an abstraction over the Resilient Distributed Dataset (RDD), a group of items that can be handled in parallel and are divided among worker nodes. The fact that RDDs are fault-tolerant and stored as read-only, immutable RDDs is another important characteristic of RDDs. RDDs are immutable, but they can be changed from one state to another using a variety of Spark functions, including map, reduce, flatMap, filter, take, collect, and count [43]. We have utilized some of Spark's helpful built-in libraries, including MLlib, data frames, and Spark SQL. In order to build a system that provides abstraction on the datasets we wish to deal with, these APIs are designed on top of Spark. Spark SQL transforms an RDD into a data frame and allows the usage of SQL statements within Spark applications, which are comparable to tables in RDBMSs. Similar to tables in RDBMSs, Spark SQL converts an RDD into a data frame and permits the use of SQL commands within Spark applications. The creation of ML applications is made possible by Spark's ML library (Spark

MLlib), which combines data processing and machine learning techniques applied over a node cluster. Users can interact with data using Spark SQL on the spark data frame platform. Numerous computer languages, including Python, R, Scala, and Java, can be used to communicate with Apache Spark. However, the reason Python was chosen for this implementation is that it is an open-source language with a wide range of free machine learning libraries (such as pandas, Scikit-learn, seaborn, matplotlib, etc.), packages, and thorough documentation for various machine learning tasks.

Anaconda and Anaconda Navigator: Anaconda Navigator is a desktop graphical user interface (GUI) that comes with the Anaconda distribution, a free Python distribution designed for scientific computing and created by Continuum Analytics. Even for business use, it is free. Its many preinstalled packages, which are necessary for data science, math, and engineering, are its strongest feature. Without using command-line commands, Anaconda Navigator enables us to effortlessly manage conda packages, environments, and channels in addition to launching apps. It is compatible with Linux, macOS, and Windows [44].

Spyder: It provides a special blend of sophisticated editing, analysis, and debugging all-inclusive development tools with scientific package features including data exploration, interactive execution, deep inspection, and gorgeous visualization [45]. Spyder is employed in its most recent version for this research project. Furthermore, the implementation makes use of the following modules:

- **NumPy**: a library for handling big matrices and multi-dimensional arrays.
- **Matplotlib**: a data visualization charting package (used to create scatter plots, line charts, histograms, and other visualizations).
- **Scikit-learn**: an open-source Python machine learning library that offers infrastructure and helpful auxiliary functions.
- ML: a collection of statistical models and methods.
- Py4j: a library that combines Apache Spark and Python.

Python comes with the first three modules, while Apache Spark comes with the latter two.

4.2 Dataset Used

KDDTrain and KDDTest are the two categories in the NSL-KDD dataset, which was utilized for the experiment. While the KDDTest is a collection of fresh instances that are absent from the training data and aid in assessing a model's performance, the KDDTrain

data is utilized to train classification algorithms. The number of cases in the NSL-KDD training and testing dataset is displayed in Table 5.

Table 5: The NSL-KDD training and testing dataset's number of instances

Category	KDDTrain Instances	KDDTest Instances
Normal	63726	13328
DoS	44,193	9192
Probe	11692	2385
R2L	3536	213
U2R	237	15
Total	123,384	25,133

Table 5 shows that while U2R and R2L classes have tiny instances in both circumstances, DoS and regular classes have big instances in both the training and testing sets. The following is a mapping of the various attack types found in the training and testing sets to the corresponding classes or categories:

4.3 Implementation detail

The experiment's methodology is described in this section. Preprocessing the dataset is the first step taken after data collection to ensure that it is in a format that is appropriate for machine learning. In order to transform the categorical data into the required, numeric form, String Indexer is first applied to the training and testing dataset.

Figure 7 illustrates that the columns labelled "labels2" and "labels5" are composed of strings, making them unusable by machines for training purposes. The binary classes "normal" and "attack" are contained in the "labels2" column, whereas the five-class "labels5" column has the following: "normal," "DoS," "Probe," "R2L," and "U2R." After applying String Indexer to the two-label column, "labels2," normal is changed to 0.0 and attack to 1.0. A new column called "labels2_index," which is the numerical form of the categorical data, is added. In the five-class column, normal, DoS, Probe, R2L, and U2R are changed to 0.0, 1.0, 2.0, 3.0, and 4.0, respectively. A new column called "labels5_index," which is the numeric form for the five class categorical data, is added.

labels2	labels2_index	labels5	labels5_index
normal	0.0	normal	0.0
normal	0.0	normal	0.0
attack	1.0	DoS	1.0
normal	0.0	normal	0.0
normal	0.0	normal	0.0
attack	1.0	DoS	1.0
attack	1.0	DoS	1.0
attack	1.0	DoS	1.0
attack	1.0	DoS	1.0
attack	1.0	DoS	1.0
attack	1.0	DoS	1.0
attack	1.0	DoS	1.0
normal	0.0	normal	0.0
attack	1.0	R2L	3.0
attack	1.0	DoS	1.0
attack	1.0	DoS	1.0
normal	0.0	normal	0.0
attack	1.0	Probe	2.0
normal	0.0	normal	0.0
normal	0.0	normal	0.0

only showing top 20 rows

Figure 7: Data with string indexing sample

Following the indexing of the categorical data, OneHotEncoder is used to replace the indices with 0s and 1s so that algorithms may be applied to these categorical features.

labels2	labels2_index	labels2_index_vec
normal	0.0	(22,[0],[1.0])
normal	0.0	(22,[0],[1.0])
attack	1.0	(22,[1],[1.0])
normal	0.0	(22,[0],[1.0])
normal	0.0	(22,[0],[1.0])
attack	1.0	(22,[1],[1.0])
attack	1.0	(22,[1],[1.0])
attack	1.0	(22,[1],[1.0])
attack	1.0	(22,[1],[1.0])
attack	1.0	(22,[1],[1.0])
attack	1.0	(22,[1],[1.0])
attack	1.0	(22,[1],[1.0])
normal	0.0	(22,[0],[1.0])
attack	9.0	(22,[9],[1.0])
attack	1.0	(22,[1],[1.0])
attack	1.0	(22,[1],[1.0])
normal	0.0	(22,[0],[1.0])
attack	3.0	(22,[3],[1.0])
normal	0.0	(22,[0],[1.0])
normal	0.0	(22,[0],[1.0])

only showing top 20 rows

Figure 8: One-hot encoded data sample

As seen in Figure 8, all input characteristics are combined into a single-vector feature (sequence data) using the VectorAssembler once the categorical data has been transformed into vector form. Figure 9 shows that all of the features have been consolidated into a single column to create a new column known as indexed features. Lastly, an ML algorithm receives the VectorAssembler's final output in order to train the data. Feature selection is the other job completed during dataset preparation. Multiple features with higher AR values are chosen for training and testing models after the 41 features are subjected to the feature selection procedure. The computed attribute ratios for each of the 41 attributes. We tested two distinct scenarios using the four algorithms that we had chosen. Using the testing data, the four algorithms are assessed after being trained with their default parameters in scenario one. In the second experiment, the chosen algorithms are subjected to the hyperparameter tuning methodology (grid search) with a 5-fold cross-validation procedure to determine the optimal parameters. The models are then retrained using these parameters. Lastly, the testing data is used to assess the models. The dataset used for this study contains 148,517 network traffic data points that have been divided into attack and normal classifications.

labels2	labels2_index	labels5	labels5_index	indexed_features
normal	0.0	normal	0.0	(77,[0,1,2,3,4,5,...
normal	0.0	normal	0.0	(77,[1,2,3,4,5,6,...
attack	1.0	DoS	1.0	(77,[0,1,2,3,5,6,...
normal	0.0	normal	0.0	(77,[0,1,2,3,4,5,...
normal	0.0	normal	0.0	(77,[0,1,2,3,4,5,...
attack	1.0	DoS	1.0	(77,[0,1,2,3,5,6,...
attack	1.0	DoS	1.0	(77,[0,1,2,3,5,6,...
attack	1.0	DoS	1.0	(77,[0,1,2,3,5,6,...
attack	1.0	DoS	1.0	(77,[0,1,2,3,5,6,...
attack	1.0	DoS	1.0	(77,[0,1,2,3,5,6,...
attack	1.0	DoS	1.0	(77,[0,1,2,3,5,6,...
attack	1.0	DoS	1.0	(77,[0,1,2,3,5,6,...
normal	0.0	normal	0.0	(77,[0,1,2,3,4,5,...
attack	1.0	R2L	3.0	(77,[0,1,2,3,4,5,...
attack	1.0	DoS	1.0	(77,[0,1,2,3,5,6,...
attack	1.0	DoS	1.0	(77,[0,1,2,3,5,6,...
normal	0.0	normal	0.0	(77,[0,1,2,3,4,5,...
attack	1.0	Probe	2.0	(77,[1,2,3,4,5,6,...
normal	0.0	normal	0.0	(77,[0,1,2,3,4,5,...
normal	0.0	normal	0.0	(77,[0,1,2,3,4,5,...

only showing top 20 rows

Figure 9: Data sample following vector assembly.

As for binary and five-class classification, the experiment is conducted with 123,384 instances in the training data (67,343 normal and 56,041 attack) and 25,133 examples in the test set (13,328 normal and 11,805 attack).

4.4 Evaluation results

The experimental results for the chosen classification algorithms under both scenarios—learning with default parameters and learning with optimal parameters by employing the hyperparameter technique—are evaluated in this section. A 2x2 and 5x5 confusion matrix is used to assess each categorization model. By counting the number of test samples under each of the four categories (TP, TN, FP, FN), the confusion matrix displays the number of accurate and inaccurate predictions generated by the classification model in relation to the target values in the data. The actual results for each model, computed using these categories, are shown in this section. With default settings, the NN achieved 88% accuracy.

With default settings, the NN achieved 88% accuracy. As indicated in Tables 6 and 7, the accuracy for the RF, DT, and LR is found to be 80%, 79%, and 76%, respectively. When compared to the other two, the NN classifier performs the best classifiers in both situations, with the RF coming in second. With default parameters, the NN achieved an overall classification accuracy of 88%; with the hyperparameters approach, it achieved an accuracy of 96.9%. In both cases, RF outperformed DT and LR in terms of classification accuracy when placed next to NN. A hyperparameter tweaking strategy is used to further increase the three classifiers' accuracy. The accuracy, precision, recall, and f1-score of the four classifiers are displayed in Table 6 prior to the use of the hyperparameter tweaking technique. The accuracy of the four classifiers trained on their default parameters is less than the accuracy obtained by training on the best parameters by applying the hyperparameter tuning technique, as shown in Tables 6 and 7.

Table 6: the four classifiers' accuracy prior to applying hyperparameter adjustment.

Classifiers	Metrics			
	Accuracy	Precision	Recall	F1-score
RF	80%	98.8%	53%	69%
LR	76%	78%	52%	69%
DT	79%	97.8%	55.8%	71%
NN	88%	88.7%	88.6%	88%

Table 7: the four classifiers' accuracy following the use of hyperparameter adjustment.

Classifiers	Metrics			
	Accuracy	Precision	Recall	F1-score
RF	96.8%	96.7%	96.6%	96.5%
LR	96.8%	97.8%	95.4%	96.5%
DT	96.2%	96.7%	96.3%	96.2%
NN	96.9%	96.8%	96.7%	96.7%

The four chosen classification methods are used for two types of sample classification: binary, where samples are classified as either normal or attack; and five-class, where samples are classified as either normal, DoS, U2R, Probe, or R2L. The next sections display the categorization outputs from both classifications.

The model has also been evaluated by different data distribution ratios as follows and given different accuracy results accordingly.

Different data processing is done while the system model is implemented, as stated in the previous chapter three. As discussed, a stratified cross-validation type was selected for validation. After partitioning the data into training, validation, and testing sets, Table 8 displays the accuracy captured during training and testingin five (5- fold) different data distributions.

Table 8 Data distribution description with its performance

Data Distribution Based on Training, Validation, and Testing)	Model Training Accuracy	Model Testing Accuracy	Description
(50%,5%,45%)	85.63%	87.22%	Over-fitting, and increasing the validation set can solve this problem
(70%,10%,30%)	87.94%	96.31%	Still, over-fitting is there
(70%,15%,15%)	92.05%	95.29%	Accuracy diminution
(80%,10%,10%)	89.88%	96.90%	Achieves well than others
(70%,10%,20%)	89.65%	88.77%	The model generalizes the training, and high over-fitting is recorded

When the **21,533** data is split into 80%, 10%, and 10%, which is 12.226k, 2.153k, and 2.153k data for training, validation, and testing, the performance is better than other data distributions. Because the documented difference between modeling, training, accuracy, and testing accuracy is smaller in this training distribution than in others, there is less overfitting.

4.4.1 Confusion matrix for binary class classification

The RF confusion matrix for the two-class classification is displayed in Figure 10. It displays the model's TP, FP, FN, and TN findings following testing on the test set. Class Normal indicates negative in this confusion matrix, while class Attack symbolizes positive.

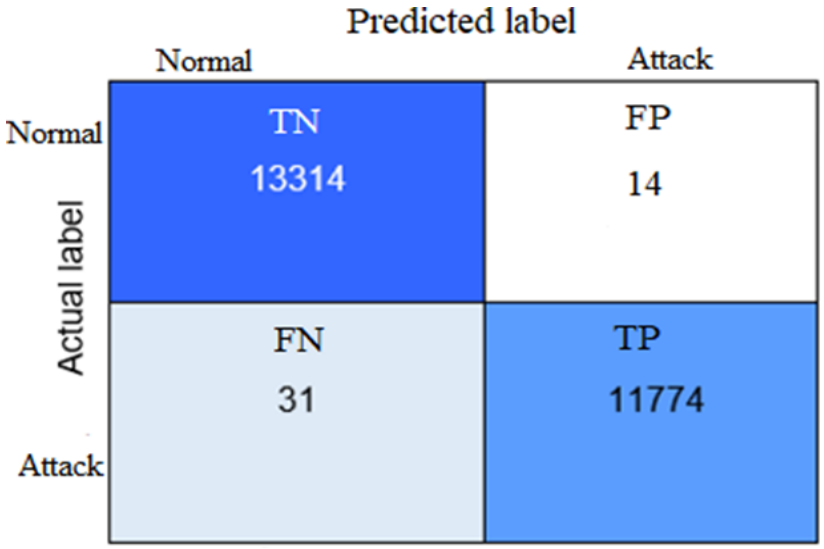
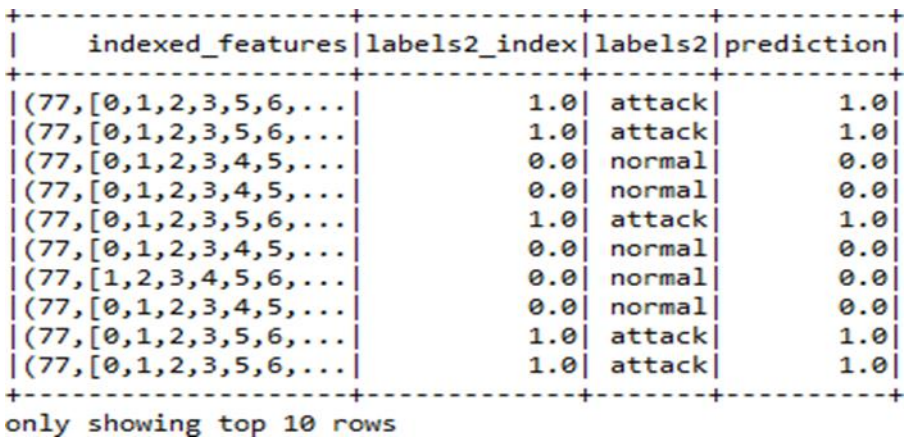


Figure 10: RF's confusion matrix for classifying binary classes.

The anticipated and actual classes of test set samples using an RF classifier for the two-class classification are shown in the confusion matrix in Figure 10. Out of all the positives in the test set, the model accurately identified 11,774 samples as positive, yielding a 99.7% true positive rate for the classifier. 13,314 samples, or 99.89% of the total negative samples in the test set, are accurately identified as negative by the model. The model obtained true positive and true negative outcomes with fewer false positive (14) and false negative (31) results in relation to this count. As a result, the model did well in the binary class.

The DT scored 0.997 precision, 0.993 recall, and 0.992 f1-score; the LR classifier obtained 0.978 accuracy, 0.954 recall, and 0.965 f1-score; and the NN classifier obtained 0.998 precision, 0.997 recall, and 0.997 f1-score.

A sample of test set predictions from the RF model on the two-class is shown in figure 11. It displays both the data's actual label and the model's predicted label. The real labels of the samples are shown in the labels2 _index column, and the model predictions for each record in the test data are displayed in the prediction column. This picture shows that the model successfully identified the first ten rows of samples as belonging to the belongings class.



indexed_features	labels2_index	labels2	prediction
(77,[0,1,2,3,5,6,...]	1.0	attack	1.0
(77,[0,1,2,3,5,6,...]	1.0	attack	1.0
(77,[0,1,2,3,4,5,...]	0.0	normal	0.0
(77,[0,1,2,3,4,5,...]	0.0	normal	0.0
(77,[0,1,2,3,5,6,...]	1.0	attack	1.0
(77,[0,1,2,3,4,5,...]	0.0	normal	0.0
(77,[1,2,3,4,5,6,...]	0.0	normal	0.0
(77,[0,1,2,3,4,5,...]	0.0	normal	0.0
(77,[0,1,2,3,5,6,...]	1.0	attack	1.0
(77,[0,1,2,3,5,6,...]	1.0	attack	1.0

only showing top 10 rows

Figure 11: Example of RF forecasts for test sets

4.4.2 Confusion matrix for Five class classification

Figure 12 shows the classification performance result for RF utilizing the five-class confusion matrix. The table displays the TP, TN, FP, and FN counts for each class (Normal, DoS, Probe, R2L, and U2R). For every class, the model achieved the greatest true positive and true negative results. Regarding the first class, or the Normal class, out of the 13,328 total normal test samples, 13,321 samples are correctly classified as normal; however, one normal sample is incorrectly classified as DoS, three normal samples are incorrectly classified as Probe, two normal samples

are incorrectly classified as R2L, and one normal sample is incorrectly classified as U2R. Out of all the test samples in the DoS class, 3 are incorrectly classified as normal, 9189 are correctly classified as DoS, and no sample is incorrectly classified as Probe, R2L, or U2R. This demonstrates how well the model has done in the DoS lesson.

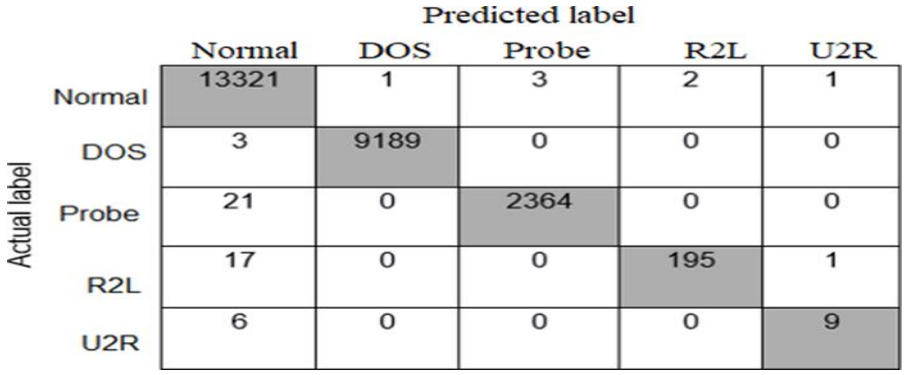


Figure 12: shows the RF confusion matrix for the five-class categorization.

There are 2,364 correctly categorized Probe test samples overall in the Probe class, 21 mistakenly classified normal test samples, and no incorrectly classified DoS, R2L, or U2R test samples. Of the entire test samples for the R2L class, 195 are correctly classified as R2L, 17 are wrongly classed as normal, and 0 samples are incorrectly identified as DoS, Probe, or U2R. Out of all the test samples for the U2R class, nine are correctly classified as U2R, six are wrongly classed as normal, and no samples are incorrectly classified as DoS, Probe, or R2L.

Table 9 displays the four classifiers' detection performance on the five-class dataset. The detection rate and FAR results for the four attack classes—DoS, Probe, R2L, and U2R—are displayed in this table. The findings demonstrate that all classification models—aside from LR—performed admirably in terms of identifying DoS assaults. In contrast to the other three classifiers, the LR demonstrated a lower detection rate and a higher false alarm rate (FAR) for DoS assaults, while failing to identify Probe, R2L, and U2R attacks. The U2R attack had the lowest performance results, despite the fact that all classification algorithms perform well in the detection of DoS attacks.

Table 9: The four methods' detection results

Classifier	Metric	DoS	Probe	R2L	U2R
DT	TP(DR)	99.8%	99.2%	93.8%	84.6%
	FP(FAR)	0.00031	0.00048	0.00036	0.000079
LR	TP(DR)	97.9%	0.0	0.0	0.0
	FP(FAR)	0.157	0.0	0.0	0.0
RF	TP(DR)	99.9%	99.1%	91.5%	60%
	FP(FAR)	0.000062	0.00013	0.00008	0.000039
NN	TP(DR)	99.9%	99.4%	92.3%	61.4%
	FP(FAR)	0.000054	0.00012	0.00006	0.000033

A sample of the test set prediction results for the five-class categorization using the RF is provided in Figure 13. The actual label (class) of the samples is contained in the labels5_index column, while the predicted class of these samples is contained in the prediction column. Figure 13 demonstrates the model's strong performance on the five-class classification.

```
+--------------------+-------------+-------+----------+
|    indexed_features|labels5_index|labels5|prediction|
+--------------------+-------------+-------+----------+
|(77,[0,1,2,3,5,6,...|          1.0|    DoS|       1.0|
|(77,[0,1,2,3,5,6,...|          1.0|    DoS|       1.0|
|(77,[0,1,2,3,4,5,...|          0.0| normal|       0.0|
|(77,[0,1,2,3,4,5,...|          0.0| normal|       0.0|
|(77,[0,1,2,3,5,6,...|          1.0|    DoS|       1.0|
|(77,[0,1,2,3,4,5,...|          0.0| normal|       0.0|
|(77,[1,2,3,4,5,6,...|          0.0| normal|       0.0|
|(77,[0,1,2,3,4,5,...|          0.0| normal|       0.0|
|(77,[0,1,2,3,5,6,...|          1.0|    DoS|       1.0|
|(77,[0,1,2,3,5,6,...|          1.0|    DoS|       1.0|
+--------------------+-------------+-------+----------+
only showing top 10 rows
```

Figure 13: Prediction sample for the five-class classification test set

4.5 Comparison of the algorithm

We do tests using NSL-KDD dataset to compare with the existing anomaly detection methods in references [47, 48, 49]. Network anomaly detection using the NSL-KDD dataset was investigated in reference [49]. In order to eliminate the highly associated attributes from the

dataset, CFS and chi-squared feature selection techniques were used. The k-means approach with information gain feature selection method was suggested in reference [48] as a means of identifying network anomalies. To find pertinent features from the NSL-KDD dataset, the author of reference [47] suggested a supervised feature selection method based on datamining methods. When the detection accuracy of the algorithms was compared, the wrapper-Bayesnet technique produced an accuracy of 95.3% and a FAR of 0.006, whereas our approaches FAR and FN rate were both 0.0%, indicating that it could distinguish between attack and normal data. Compared to the other approaches utilized in the three referenced works, the feature selection method employed in our proposed study is easier to calculate. The studies' detection accuracy is likewise inferior to our method. The detection accuracy of earlier research is contrasted with our suggested study effort in Table 10. These comparison results show that, in terms of network anomaly identification, our method performs better than the research in references [47, 48, 49].

Table 10: Evaluation of Block chain based (our approach) with others.

Approach	**Accuracy**
CFS and chi-squared based [47]	92.13%
k-means with IG based [48]	89.6%
Wrapper-Bayesnet based[49]	95.3%
Blockchain based (our approach)	96.9%

Chapter Five

Conclusions and Future Works

5.1 Conclusions

Network or Internet communications are constantly growing as a result of the quick development of electronic devices and widespread reliance on Internet-based applications for both business and leisure activities. On the other hand, since attackers have occasionally increased, security challenges have emerged as one of the most pressing issues facing organizations. There should be a system in place to safeguard against these security risks due to the growing number of Internet users and sophisticated cyberattacks targeting computer networks. ML approaches are among the technologies utilized by network intrusion detection systems. The primary goal of this research project is to develop an effective model that uses several Blockchain-based machine learning approaches to categories network traffic as normal or attack. These methods are put into practice with Apache Spark, a quick data processing framework with a number of modules for Blockchain-based machine learning applications. Two methods are used to validate the ML approaches. The models are tested on a different test set after being trained using their default settings in the first experiment. In the second experiment, the ML techniques are combined with grid search techniques to do a 5-fold cross-validation and determine the ideal value for each hyperparameter, hence increasing the models' accuracy.

The process of implementation began with preprocessing the dataset. From the preprocessed data, features were chosen, and the classifiers were trained and tested using these features. Four distinct machine learning models—DT, RF, LR, and NN—are developed, trained, tested, and compared depending on the results of their evaluations during the classification process. In order to complete this study, the performance of the suggested models—which come from the University of New Brunswick—is assessed using the NSL-KDD dataset. This dataset is used to implement the models in the Python programming language.

DT scored 96.2%, precision of 96.7%, recall of 96.3%, and f1-sore of 96.2%. LR scored 96.8%, precision of 97.8%, recall of 95.4, and f1-sore of 96.5. RF scored 96.8%, precision of 96.7%, recall of 96.6%, and f1-score of 96.5%. These results are based on the evaluation results of the various models presented in Chapter 5. This leads us to the conclusion that all the models—aside from the LR—specify all normal samples and are effective in identifying attacks. In

general, the NN model performs better across the board in all evaluations, and the LR has the lowest score among the models. The evaluation's findings demonstrate that the suggested method effectively detects network breaches.

5.2 Future works

This study examines many areas that could be used as additional improvement to enhance the system's performance.

- Using the system with various datasets
- Using integrated and hybrid Blockchain-based machine learning methods to detect network breaches from actual networks using collaborative based IDs.
- Establishing a more accurate model that can be utilized in real-time to detect and categorize anomalies with fewer false alarms and in less time. To accomplish this, combine a number of detection techniques and apply them to an anomaly detection module.
- Experimenting with various feature selection strategies by fusing our methodology with other approaches

References

[1] Check Point Software Technologies Ltd. “What is a Cyber Attack?” Internet: www.checkpoint.com/definitions/what-is-cyber-attack/, Feb. 27, 2020 [Oct 27, 2023].

[2] Shreevyas, H.M., Kumar, C.S., Diat-Drdo, P., et.al. “Can Blockchain technology be the future of network intrusion detection systems: A review.” International Journal of Applied Engineering Research, vol. 14, pp. 10179-10187, 2019.

[3] Li, W., Wang, Y., Li, Au., et.al. “Towards blockchained challenge-based collaborative intrusion detection.” in Proc. 17, 2019, pp. 122-139.

[4] Baliga, A. “Understanding blockchain consensus models.” Persistent, vol. 4, pp.14, 2017.

[5] Tug, S., Meng, W. and Wang, Y. “CBSigIDS: towards collaborative blockchained signature-based intrusion detection.” in Proc. IEEE, 2018, pp. 1228-1235.

[6] Alexopoulos, N., Vasilomanolakis, E., Ivánkó, N.R., et.al. “Towards blockchain-based collaborative intrusion detection systems.” in Proc. CRITIS, 2017, pp. 107-118.

[7] Kanth, V.K. “Blockchain for use in collaborative intrusion detection systems”, Doctoral dissertation, Naval Postgraduate School, USA, 2019.

[8] Kim, S., Kim, B. and Kim, H.J. “Intrusion detection and mitigation system using blockchain analysis for bitcoin exchange.” in Proc .CC&IOT, 2018, pp. 40-44.

[9] Di Pietro, R. and Mancini, L.V. eds. “Intrusion detection systems.” Springer Science & Business Media, vol. 38, 2008.

[10] Bace, R.G. and Mell, P. “Intrusion detection systems.” 2001.

[11] Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., et.al. “Anomaly-based network intrusion detection: Techniques, systems and challenges.” Computers & security, Vol. 28, pp. 18-28, 2009.

[12] Jyothsna, Prasad, R. and Prasad, K.M. “A review of anomaly-based intrusion detection systems.”, International Journal of Computer Applications, vol. 28, pp. 26-35, 2011.

[13] Raj, R.V., Narayanan, S.A. and Bijlani, K. “Heuristic-based automatic online proctoring system.” in Proc. IEEE, 2015, pp. 458-459.

[14] Kuznetsov, A.A., Smirnov, A.A., Danilenko, D.A, et.al. “The statistical analysis of a network traffic for the intrusion detection and prevention systems.” Telecommunications and Radio Engineering, vol. 74, 2015.

[15] Zamani, M. and Movahedi, M. “Machine learning techniques for intrusion detection.”, arXiv preprint arXiv, 2015.

[16] Vigna, G. and Kemmerer, R.A. “NetSTAT: A network-based intrusion detection system.”, Journal of computer security, vol. 7, pp. 37-71, 1999.

[17] ScienceDirect. “Network Based Intrusion Detection System.” Internet: https://www.sciencedirect.com/topics/computer-science/network-based-intrusion-detection-system, Aug. 12,2023 [Oct. 27, 2023].

[18] Hu, J. “Host-based anomaly intrusion detection.” in Proc. ICS, 2010, pp 235-255.

[19] Ierace, N., Urrutia, C. and Bassett, R. “Intrusion prevention systems.” Ubiquity, vol. 6, pp. 2, 2005.

[20] Vasilomanolakis, E., Karuppayah, S., Mühlhäuser, M, et.al. “Taxonomy and survey of collaborative intrusion detection.” ACM Computing Surveys (CSUR), vol. 47, pp. 1-33, 2015.

[21] Hu, B., Zhou, C., Tian, Y.C, et.al. “A collaborative intrusion detection approach using blockchain for multimicrogrid systems.” IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 49, pp. 1720-1730, 2019.

[22] Zhou, C.V., Leckie, C. and Karunasekera, S. “A survey of coordinated attacks and collaborative intrusion detection.” computers & security, vol. 29, pp. 124-140.

[23] Alexopoulos, N., Vasilomanolakis, E., Ivánkó, N.R, et.al. “Towards blockchain-based collaborative intrusion detection systems.” in Proc. CRITIS, 2018, pp. 107-118.

[24] Zheng, Z., Xie, S., Dai, H.N, et.al. “Blockchain challenges and opportunities: A survey.” International Journal of Web and Grid Services, vol. 14, 2017.

[25] Okada, H., Yamasaki, S. and Bracamonte, V. “Proposed classification of blockchains based on authority and incentive dimensions.” in Proc. IEEE, 2017, pp. 593-597.

[26] S. Nakamoto. “Bitcoin: A peer-to-peer electronic cash system.” Internet: https://bitcoin.org/bitcoin, 2009 [Nov 10, 2023]

[27] Blockgeeks. “What Is Hashing?” Internet: https://blockgeeks.com/guides/what-is-hashing/, Aug. 2017 [Nov 10, 2023].

[28] Persistent Systems. “Understanding blockchain consensus.” Internet: www.persistent.com/whitepaperunderstanding-blockchain-consensus-models/, 2017 [Nov 15, 2023].

[29] University of Central Florida. “Anonymous byzantine consensus from moderately hard puzzles: A model for bitcoin.” Internet: https: //socrates1024.s3.amazonaws.com/consensus, 2014 [Oct. 25, 2023].

[30] The Ethereum Wiki. “Contribute to ethereum/wiki development by creating an account on GitHub.” Internet: https://github.com/ethereum/wiki, Aug. 2019 [Nov. 15, 2023].

[31] B. Vitalik. “A next-generation smart contract and decentralized application platform Ethereum.” Internet: https://github.com/ethereum/wiki/wiki/WhitePaper, 2014 [Nov. 10, 2023].

[32] Azaria, A., Ekblaw, A., Vieira, T, et.al. “Using blockchain for medical data access and permission management.” in Proc. IEEE, 2016, PP. 25-30.

[33] Asyâ, M.F., Budiyono, A. and Widjajarto, A. “Analisa Parameter Ethereum Pada Jaringan Peer to Peer Blockchain di Aplikasi Transfer Koin Terhadap Aspek Processor.” eProceedings of Engineering, vol. 6, 2019.

[34] Asyâ, M.F., Budiyono, A. and Widjajarto, A. “Analisa Parameter Ethereum Pada Jaringan Peer to Peer Blockchain di Aplikasi Transfer Koin Terhadap Aspek Processor.” eProceedings of Engineering, vol. 6, 2019.

[35] Li, W., Tug, S., Meng, W, et.al. “Designing collaborative blockchained signature-based intrusion detection in IoT environments.” Future Generation Computer Systems, vol. 96, pp. 481-489, 2019.

[36] Ujjan, R.M.A., Pervez, Z. and Dahal. “Snort based collaborative intrusion detection system using blockchain in SDN.” in Proc. IEEE, 2019, pp. 1-8.

[37] Kolokotronis, N., Brotsis, S., Germanos, G., et.al. “Blockchain Architectures for Trust-based Collaborative Intrusion Detection” IEEE world congress on services, IEEE world congress on services, vol.2642, pp. 21-28, July 2019.

[38] Alkadi, O., Moustafa, N., Turnbull, B, et.al. “A deep blockchain framework-enabled collaborative intrusion detection for protecting IoT and cloud networks.” IEEE Internet of Things Journal, vol. 8, pp. 9463-9472, 2020.

[39] Ibrahim, L.M. “Anomaly network intrusion detection system based on distributed time-delay neural network (DTDNN).” Journal of Engineering Science and Technology, vol. 5, pp. 457-471, 2010.

[40] Apache Spark. “Databricks” Internet: https://databricks.com/spark/about, Dec. 7, 2019 [Nov. 20, 2023].

[41] Alex Bekker. “Spark vs. Hadoop MapReduce Databricks” Internet: https://www.scnsoft.com/blog/spark-vs-hadoop-mapreduce, Dec. 7, 2019 [Nov. 20, 2023].

[42] Apache Spark. “Lightning-fast Cluster Computing.” Internet: https://svn.apache.org/r [1]epos/asf/spark/site/index.html, Dec. 7, 2019 [Nov. 20, 2023].

[43] M. Beyeler. “Machine Learning for OpenCV.” Packt Publishing Ltd, 2017.

[44] Spyder. “Welcome to Spyder Documentation.” Internet: https://docs.spyder-ide.org/, Dec. 29, 2019 [Nov. 23, 2023].

[45] Gupta, G.P. and Kulariya, M. “A framework for fast and efficient cyber security network intrusion detection using apache spark.” Procedia Computer Science, vol. 93, pp. 824-831, 2016

[46] Effendy, D.A., Kusrini, K. and Sudarmawan, S. “Classification of intrusion detection system (IDS) based on computer network.” in Proc. IEEE, 2017, pp. 90-94.

[47] Martha Teffera. “A Data Mining Approach for Intrusion Detection System Using Wrapper Based Feature Selection Method.” M. Sc Thesis, Addis Ababa University, Ethiopia, 2014.

[48] Li, W., Tug, S., Meng, W, et.al. “Designing collaborative blockchained signature-based intrusion detection in IoT environments.” Future Generation Computer Systems, vol. 96, pp. 481-489.

[49] Ujjan, R.M.A., Pervez, Z. and Dahal, K. “Snort based collaborative intrusion detection system using blockchain in SDN.” in Proc. IEEE, Aug. 2019, PP. 1-8.