Sigma Journal of Engineering and Natural Sciences, Vol. 42, No. 5, pp. XX-XX, October, 2024
Yildiz Technical University Press, Istanbul, Turkey

A novel machine learning-based artificial intelligence approach for log analysis using blockchain technology

Rizwan Ur RAHMAN[1] [0000-0002-5801-6625], Pavan KUMAR [2 *] [0000-0001-5340-7777],
Gaurav Pramod KACHARE[1] [0000-0003-2452-8179], Meeraj Mahendra GAWDE[1] [0000-0001-6715-4838],
Tenzin TSUNDUE[1] [0000-0003-1837-744X], and Deepak Singh TOMAR[3] [0000-0001-9025-1679]

ABSTRACT

Cybercrime is one of the fastest-growing crimes worldwide. It is observed that every seven seconds, cyber attackers penetrate cyber systems. While detecting an anomaly or attack, the log system is one of the crucial components of any system storing and managing all the events. It has always been challenging to detect an anomaly in logs. This is because of continuous and ever-changing log events and their mutability property. In this paper, we develop a machine learning-based artificial intelligence approach to address this issue of log analysis by proposing two modules. The first one is anomaly detection using different machine learning models. The second one is a distributed immutable storage system for securely storing the logs. In addition, we present a descriptive and user-friendly web application by integrating all modules using HTML, CSS, and Flask Framework on the Heroku cloud environment. The results demonstrate that the proposed hybrid machine learning models are capable of achieving 99.7% accuracy in detecting network anomalies.

Keywords: *Machine Learning; Logs; Distributed System; Immutable Storage; Blockchain; KNN; Isolation Forest*

1. INTRODUCTION

As both people and technology improve, there are more requests being made of the server than ever before, which increases the generation of Log data. On the other hand, there are more cyberattacks happening now, and log analysis and anomaly detection are becoming increasingly important [1, 2]. With security aspects, log entries are an essential part where developers must be aware of users injecting malicious content into logs called log injection. Security Logging and monitoring Failures are so at risk that they made Open Web Application Security Project Top 10 2022 [3, 4]. Typically, organizations that routinely check and analyze logs are better able to spot mistakes. The organization may even be able to identify issues with a sophisticated log analysis tool, considerably reducing the time and expense of remedy. The log also assists the security engineers in reviewing the events that preceded the error, which may make it simpler to troubleshoot and fix the problem going forward [5]. Additionally, for big businesses like Meta and Google, where the volume of data is so enormous that even with the presence of specialists, manual presence has not been sufficient, machine learning has come to the rescue with automatic anomaly detection that requires little to no human involvement. However, it has been difficult to change every few years because there isn't a sufficient standard up-to-date actual log labeled data set for reference and log message [7, 8].

This paper was recommended for publication in revised form by Regional Editor Omid Mahian
[1] School of Computing Science and Engineering, VIT Bhopal University, Kothrikalan, Sehore-466116, INDIA.
rizwan.ur@vitbhopal.ac.in (R.U.R.); gaurav.pramod2018@vitbhopal.ac.in (G.P.K.)
mahendra.gawde2018@vitbhopal.ac.in (M.M.G.); tenzin.tsundue2018@vitbhopal.ac.in (T.T.)
[2] School of Advanced Science and Languages, VIT Bhopal University, Sehore-466116, INDIA. pavankmaths@gmail.com (P.K.)

This paper was recommended for publication in revised form by Regional Editor Ahmet Selim Dalkilic
[1] School of Computing Science and Engineering, VIT Bhopal University, Kothrikalan, Sehore-466116, India
[2] School of Advanced Science and Languages, VIT Bhopal University, Sehore-466116, India
[3] Department of Computer Science and Engineering, Maulana Azad National Institute of Technology, Bhopal-462003, India
** E-mail address: pavankmaths@gmail.com*
Orcid id: https://orcid.org/0000-0002-5801-6625 Rizwan Ur Rahman, 0000-0001-5340-7777 Pavan Kumar,
0000-0003-2452-8179 Gaurav Pramod Kachare, 0000-0001-6715-4838 Meeraj Mahendra Gawde,
0000-0003-1837-744X Tenzin Tsundue, 0000-0001-9025-1679 Deepak Singh Tomar
Manuscript Received 27 May 2023, Revised 18 August 2023, Accepted 22 November 2023

[3] Department of Computer Science and Engineering, Maulana Azad National Institute of Technology, Bhopal-462003, INDIA.
deepaksinghtomar@manit.ac.in
* Correspondence: pavankmaths@gmail.com
Manuscript Received 27 May 2023, Accepted 22 October 2023

According to security analysts, there will be a cybersecurity attack occurring every seven seconds around the globe by the end of 2023 [9]. The key idea of this paper is to develop suitable solution that will act as both a proactive and reactive approach to tackle the cyber security related risks and issues occurring in organizations. This solution processes the most minor yet crucial component of data generated from any activity performed within an organization's systems Logs. Building different test beds for producing random log files has also been a significant achievement in the cyberspace for research reference. All types of machine learning models, from clustering to neural networks, have been evaluated and contrasted using data sets that have previously been collected [10]. The realness of the data set has proven difficult because the majority of conventional data sets are dated.

This paper proposes Blockchain for log storage and data sets from the most realistic test beds with suitable anomaly detection algorithms. To execute it, we develop a model for detecting threats and vulnerabilities at different levels in systems. These are server logs and network devices logs using multiple machine learning algorithms such as KNN and Isolation Forest to analyze logs and prevent any forgery in a log of the system. In order to maintain the indignity of logs we are also propose distributed immutable storage.

As cyber threats are becoming more sophisticated our existing monitoring systems are not enough to detect the threats. The best examples of that are APT (advanced persistent threats) a sophisticated, sustained cyberattack in which an intruder establishes an undetected presence in a network in order to steal sensitive data over a prolonged period of time. To cover up tracks most of the time these malwares try to update or delete systems and network logs. Making them undetected for a longer period of time. So, we are proposing solutions on these problems by securing most crucial logs then by providing a machine learning based detection mechanism to detect and alert about threats, with a neat and optimized user interface**.**

The use of blockchain technology for log storage is described in the study. This innovative method improves the immutability and security of log data. The study maintains the integrity and reliability of log records by utilizing a decentralized and tamper-proof blockchain ledger. Our Proposed Model also addresses the detection of threats and vulnerabilities at different levels within systems. Our proposed work's novelty lies in its holistic approach to log management and security.

It combines blockchain technology, machine learning algorithms, and realistic test bed scenarios to detect and prevent threats and vulnerabilities in a multi-level system. Distributed immutable storage is also used to guarantee the long-term security and integrity of log data. Together, their contributions increase cybersecurity and log management.

Our proposed approach can be enhanced to the following Applications of Log Analysis using Blockchain with Artificial Intelligence:

a. Fraud Detection and Prevention: Blockchain technology can be used to keep track of transaction records in the banking and e-commerce sectors. Then, these records can be examined by AI models to spot suspicious or fraudulent activity. Real-time flagging of suspicious transactions can help stop financial fraud and ensure secure transactions.

b. Healthcare Data Management: Blockchain technology can be used by healthcare organizations to safely log patient information, medical history, and access logs. AI can help with data analysis for medical research, trend detection in patient care, and privacy compliance. Additionally, it can aid in identifying unwanted access to private medical records.

c. IoT Security: Blockchain can secure device communication logs in the Internet of Things (IoT), and AI can check these logs for anomalous device behavior to assist stop security breaches and safeguard crucial infrastructure.

d. Authentication and Access Control: User identification and access control logs can be securely managed using blockchain technology. By regularly examining user activity patterns to spot unwanted access attempts, AI algorithms can improve security.

The remainder of this paper is organized as: Section 2 discusses the background and overview of Blockchain and related terms in different headings such as Blockchain, Distributed System, Merkle Tree, and Logs. Section 3 enhances the

related work of log analysis and presents the state-of-the-art comparison of solutions for Log Analysis. We develop the proposed model in Section 4. The evaluation of the proposed model is presented in detailed in four different headings: Apache Logs Analysis, Network Log Analysis, Blockchain, and Demonstration of Web App. Section 5 gives detailed result analysis. The last section includes the conclusions.

2. RELATED WORK

LogRobust anomaly detection based on log analysis is proposed by [11] to mitigate the problem of anomaly detection on unstable logs by finding the critical semantics information and processing through Bi-LSTM model collecting from the Hadoop and Microsoft System. Authors in [12] proposed anomaly log detection with automatic neighbor selection for supervised KNN algorithm mitigating the problem of proper Kneighbors selection for large-scale data with different distribution. A similar logs min hash algorithm is used for grouping, and an MVP tree is used for each bucket. Authors in [13] have reviewed the robustness of the LSTM network for the detection and classification of anomalous events in sensor files. In comparison, RNN could only compete in foreseen anomalies events with LSTM. This paper concludes that LSTM performs far better than RNN in log-based anomaly detection an anomaly.
Authors in [14] has considered log files as a standard plain text file to apply NLP to extract information about the log events, including the most popular embedding technique, which is based upon Google's word2vec algorithm. This could perform online detection of any anomaly. Their future work includes the collation of their work with an unsupervised algorithm. Authors have proposed ADA which acronym of adaptive deep log anomaly detector [15]. It is unsupervised deep neural network framework that makes use of LSTM networks with the dynamic threshold. This framework has proven to increase the f1 score significantly compared to existing methods with decreased storage.
To prevent log manipulation, authors suggested a cloud blockchain-based log system and included a tamper detection tool called Logchain [16]. Authors described offering blockchain-based log storage as a service in [17]. utilizing the Ethereum public blockchain network for block validation and merging current public and private blockchain services. Using blockchain technology for enterprise use, authors of [18] worked on Identity and Access Management (IAM) logs for the Internet of Things (IoT). enables the temper-proof IAM and keeps the users' and transactions' solubility; this makes it the best option for security audits and monitoring in businesses.
Authors proposed in [19] Framework for a Secure Log Storage system using blockchain technology. It uses temper proof and decentralization properties of blockchain by utilizing the Ethereum public blockchain for indexing and hashing log files. Medusa [20] is a blockchain-powered immutable storage system for storing logs. This paper mainly focuses on storing weblogs in a blockchain, and it yields much higher throughput and significantly lower latency.

In [21] authors developed a novel approach which keeps complete track of the underlying system events and also monitors dependencies and event occurrences and therefore learns the natural system behavior over the period of time and reports all the malicious actions that differ from the created system model.

A probabilistic technique was put forth by authors in [22] that shows the percentage frequency of occurrence of an event while still taking into account a false alarm rate that is acceptable. This paper provides [23] a scalable and successful process and event analysis technique that includes parallel algorithms to enable quick event correlation for enormous amounts of process data. Over large event datasets, the MapReduce framework suggests a two-stage process for identifying probable event connections and verifying them.

Authors in [24] proposed a unique algorithm based on the principles of filtering and verification, which is also called RF-GraP, which provides a very effective correlation in a distributed systems environment. Authors in [25] provide a detailed explanation of methods for correlating different security events and integrating diverse data sources for defense against cyber-physical threats. Authors in [26] set test beds with actual realistic generating log directory files for six days for evaluation purposes when testing for Intrusion Detection System.

The literature review includes a variety of methods for anomaly detection and log analysis using machine learning methods. Bi-LSTM and KNN have both been investigated by researchers with an emphasis on handling unstable logs and large-scale data distribution. For online anomaly detection, Natural Language Processing methods like word

embeddings are used. Performance in log anomaly detection has significantly improved thanks to deep learning techniques, specifically ADA using LSTM networks.

In summary, [12] addresses the difficulty of choosing appropriate neighbors for large-scale data by introducing a supervised KNN approach with automatic neighbor selection for anomaly log detection. According to [13], LSTM performs better than RNN in detecting abnormal events in sensor files than log-based anomaly detection. [11] suggests using a Bi-LSTM-based method to examine logs from Microsoft System and Hadoop for anomaly detection. In contrast to current techniques, [15] offers ADA, an adaptive deep log anomaly detector employing LSTM networks with dynamic thresholds, which dramatically raises the F1 score

We have done systematic comparison on the following parameters: R1- Anomaly Detection, R2- Server Log Analysis, R3- Network Log Analysis, R4-Resource Optimization, R5- Data Confidentiality Consideration, R6- Distributed File System. Table 1 provides the most recent comparison of Log Analysis solutions.

Table 1: Most Recent Survey and Comparison of Solutions for Log Analysis.

Author	Key Contributions	R1	R2	R3	R4	R5	R6
Xu et al [11]	LogRobust is able to identify and handle unstable logs	Yes	Yes	No	No	No	No
Bingming et al [12]	Efficient KNN neighbour search for anomaly detection	Yes	Yes	Yes	Yes	No	No
Vinayakumar et al [13]	Evaluated simple LSTM model for anomaly detection	Yes	Yes	No	No	Yes	No
Christophe et al [14]	Mined log as normal text using NLP	Yes	Yes	Yes	Yes	No	No
Yali et al [15]	framework for lightweight anomaly detection.	Yes	No	Yes	Yes	No	No
Friedberg et al [21]	White listing based anomaly detection approach with event correlation	No	Yes	No	Yes	No	Yes
Ambre et al [22]	Frequency based event correlation system for insider threat detection	Yes	No	Yes	No	No	No
Reguieg et al [23]	Scalable event correlation system with MapReduce framework	Yes	Yes	No	No	Yes	Yes
Oliver et al [24]	Filtering-and-Validation based event correlation over distributed system	Yes	No	Yes	No	No	No
Kotenko et al [25]	Heterogeneous data integration approach for event correlation	Yes	Yes	No	Yes	No	No
William et al [26]	Cloud based logging system using blockchain	No	Yes	Yes	No	No	Yes
John et al [27]	Log storage as service in public blockchain network	No	Yes	No	Yes	No	Yes
Nuss et al [18]	Enterprise level security audit system for IAM logs	No	Yes	Yes	No	Yes	No
Huang et al [19]	Tamper-proof decentralized log storage system	No	Yes	No	No	Yes	Yes
Wang et al [20]	Immutable storage system for web log data	Yes	No	No	Yes	No	No
Proposed Model	**Layered log analysis using ML & distributed immutable storage**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

3. BACKGROUND AND RELATED TERMS

In this section, some required terms are introduced in order further understand the log analysis using distributed immutable storage. These terms are as follows: Blockchain, Distributed System, Proof-of-Work, Merkle Tree, Public Key Cryptography, Logs, Server Logs, Network Devices Log, and KNN Classifiers.

3.1 Blockchain

The word "Blockchain" describes a network of linked records that are highly resistant to alteration and are protected by encryption techniques. In 2008, blockchain technology was first made public. A complicated data structure using an asymmetric encryption algorithm and hash functions was detailed in the well-known research paper on the Bitcoin cryptocurrency, Bitcoin: a peer-to-peer electronic cash system.

3.2 Distributed System

Distributed computing is a collection of independent systems or nodes interconnected in a network and can share idle resources currently not utilized by any other method. There are several disadvantages to this architecture. There is an increased amount of computational power involved. It gets increased whenever a new node or server is added; there is a greater demand for storage, and the maintenance of this vast system is not easy. Interplanetary File System (IPFS) is a peer-to-peer protocol for a distributed cloud platform. It uses Blockchain to ensure decentralized architecture and no single point of failure. IPFS offers a separate naming system called IPNS for storing and accessing information quickly.

3.3 Proof-of-Work

Proof-of-Work is the first consensus algorithm that was implemented in the blockchain network. Cur- rently, it is used by several blockchain technologies and add blocks. It has a decentralized ledger that holds information related to blocks and transactions. Proof-of-Work (abbreviated as PoW) to verify a transaction and add a new block to the Blockchain, consensus on the network is employed. With proof of work, miners compete to validate a transaction and get rewarded. A crucial part of the mining process is called Proof of Work, when miners locate a legitimate block by resolving a mathematical conundrum. Miners must continuously alter the nonce value until the generated hash is higher than a predetermined threshold since the block header contains a 32-bit nonce field.

3.4 Merkle Tree

Blockchain data is securely encoded and hashed using the hash tree, often known as the Merkle tree. It permits effective mapping of large amounts of data, quicker blockchain data validation, and quicker transfer of large amounts of data across nodes in a peer-to-peer blockchain network. We can quickly check the data consistency or where the change in data has occurred by crosschecking the root hash value without traversing the whole tree repeatedly. The entire set of data is fingerprinted using the root hash. The information kept in a Merkle tree's leaf nodes is hashed before the hash values of both leaf nodes are combined. This continues until the tree's root is reached. Merkle trees can be utilized in applications for distributed systems where copies or similar data are stored in many locations. At the moment, blockchain and bitcoin both use it.

3.5 Public Key Cryptography

Public key cryptography is asymmetric encryption in which data is transferred using two different keys, one for encryption and another for decryption. Here, the first two keys are generated: the public and private keys. The receiver’s public key is used for data encryption, and then the receiver can use his private key to decrypt and hence receive the confidential data. There are five steps in the entire procedure: Key generation, Key exchange, encryption, data transfer, and decryption.

3.6 Logs

Logs is defined as automated computer-generated data files that track activities performed by systems like operating systems, application servers, database servers, network devices, and many more during interaction with any user or another system. These activities could be logging, retrieving or inserting data, deleting or modifying data, etc. The primary purpose of these logs is to keep all the records of such activities that are or could be responsible for detecting any security issues in the systems in the future like its source, time, or medium [27].

3.6.1 Server Logs

It is simply a text file automatically generated within the system server that contains the records of all activities performed within it by users or another system like logging, input requests, and its response by who, whom, and when in a given period. Types of servers could be an application server, database server, network server, etc.

3.6.2 Network Devices Log

This category includes logs from network devices such as routers, switches, load balancers, intrusion detection systems, and more. These logs offer in-depth details on what is occurring. It keeps track of activities including program startup and shutdown, commands run, information about logging in and out, connections made, and more. This extensive history of operations is a perfect example of why log data should be referred to as a network's footprints [28].

3.7 Machine Learning for Log Analysis

The term machine learning was first used in 1959 by Arthur Samuel. He was a pioneer in the field of computer games and artificial intelligence. Machine learning is an application of artificial intelligence (AI) technology that gives systems the ability to automate the process of learning and to develop from experience without being coded separately. Machine learning concentrates on developing programs that can access and analyze a chunk of data and then use it for themselves. The learning process starts with collecting big data or observations, for example, specific information or instructions, for finding and creating patterns in the data frame and making better decisions in the future based on the samples we provide.

Below algorithms were used in our paper for log analysis and anomaly detection. Isolation Forest Isolation Forest is an unsupervised machine learning algorithm that mainly detects anomalies instead of outlining regular data points. This algorithm is built based on decision trees like other ensemble tree methods. Outliers or anomalies are the data points that deviate significantly from the other observations. Compared to routine observations and value discrepancies, outliers are less common. Isolation forest has many use-cases, for example, network anomaly detection, fraud detection, and sudden change in sales. Isolation by picking a split between the feature's maximum and minimum values at random, forest isolating outliers. KNN algorithm for classification is a supervised learning algorithm that predicts the values based on the nearest already trained points with similar features. The algorithm has relatively high accuracy, and with more data points, the classifier constantly evolves but becomes computationally expensive due to high storage requirements [29,30].

3.8 Flask

Flask is a framework for developing web applications using python, which comes with a debugger and provided server. It is lightweight as the framework does not contain external libraries with only necessary features. The template is a folder containing static files responsible for passing dynamic data from the back-end code. Moreover, the flask does send the essential data by rendering templates using the jinja package. With full completion of the flask web app, there are lots of options to operate in the cloud, and one of the reliable options is the Heroku app, which is a platform as a service (PaaS) company letting developers run their applications in the cloud for free with some restrictions.

4. PROPOSED HYBRID MODEL FOR LOG ANALYSIS

To tackle the increasing problems raised due to attackers clearing their tracks and making the systems and servers seem normal after every successful cyber-attack, we propose a model with an immutable log storage system and root cause analysis. In this proposed model, Blockchain-based log monitoring system with the features of detecting anomalies on logs which will keep the logs tamper-proof and detect any anomalies in the present and future unseen logs is presented. The Architectural Diagram is shown in Figure 1

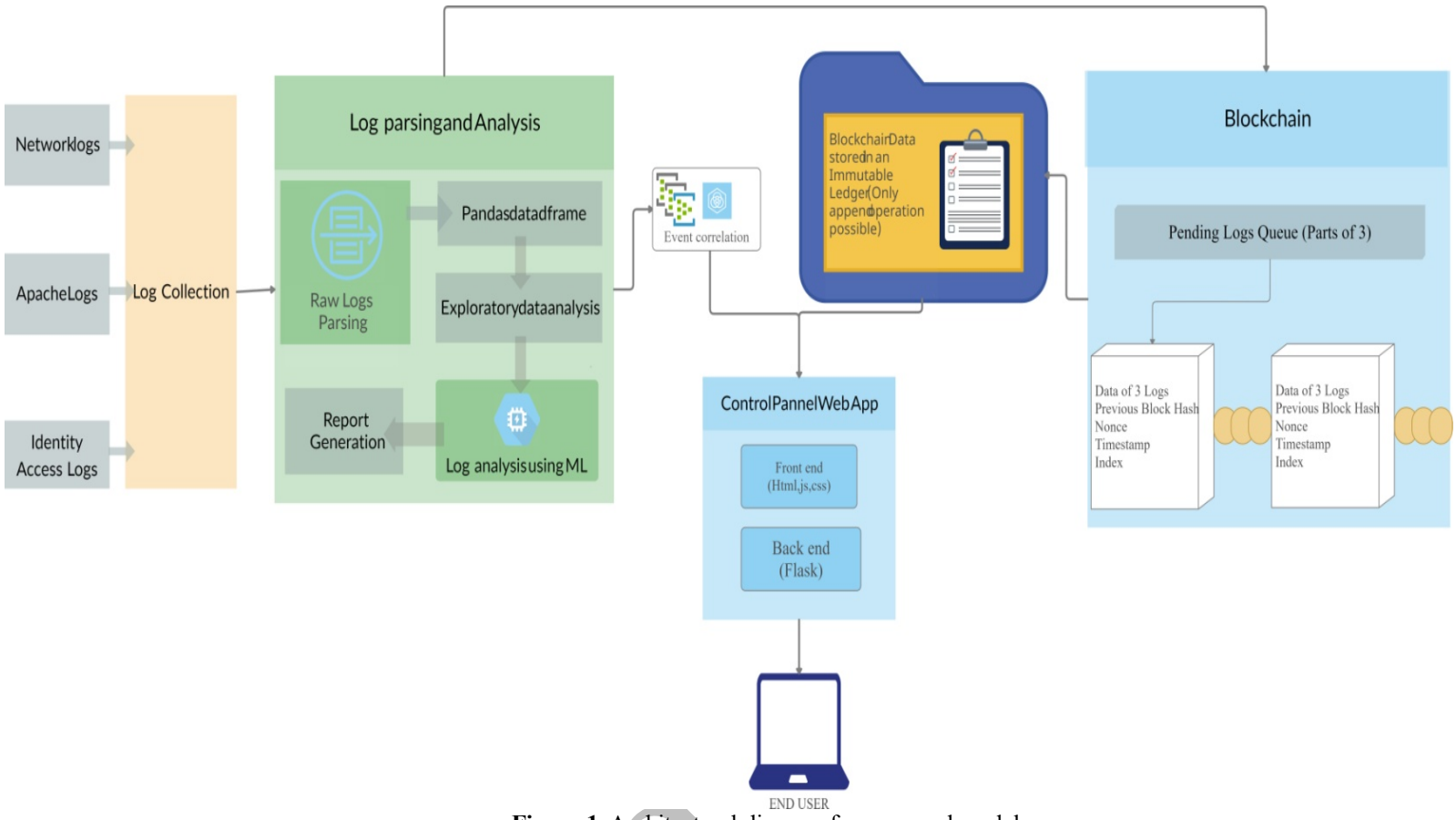


Figure 1. Architectural diagram for proposed model

The proposed model consists of the following subcomponents such as Server Log Analysis, Network Log Analysis, Blockchain, and Distributed File Storage System. Description of each component is as follows:

4.1 Server Log Analysis

Apache Logs are basically information regarding events happening in the particular Apache Web servers. Apache Logs come in two kinds such as Access and Error Logs. The data is classified only to server administrators, and Log data includes IP, timestamp, the request method, status code, referrer, Byte size, User-Agent, etc. With a large number of Server Logs, the administrator gets a grasp of an incoming request and user behaviors which helps in the analysis of possible upcoming attacks, digging for the root cause of past problems, and so on. The Apache Access Logs formatting is shown in Figure 2.

Before

192.168.10.190 - - [29/Feb/2020:00:00:12 +0000] "POST /login.php HTTP/1.1" 302 601 "http://mail.cup.com/login.php" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:73.0) Gecko/20100101 Firefox/73.0"

After Log formatting

ip_addr	ident_check	connecting_user	time_stamp	time_zone	request_method	status_code	byte_size	referer	user_agent
192.168.10.190	-	-	29/Feb/2020:00:00:12	0000	POST	302	601	http://mail.cup.com/login.php	Mozilla/5.0

Figure 2. Apache access logs formatting

We use KNN classifiers and the Isolation Forest algorithm to detect certain anomaly logs which deviate from the base distribution. The algorithms are chosen based on our research and understanding and those algorithms that have performed great results in existing papers.

4.1.1 KNN Classifiers

K-nearest Neighbors algorithm is the most well-known classification algorithm out there. It is simple to implement, and the accuracy drastically increases with more training data sets depending on the value of K. KNN algorithm does not work with missing values, but the data set used here is all labeled and preprocessed.

4.1.2 Hyper-parameters Tunning of KNN

Hyper-parameters regarding the KNN Classifiers includes n-neighbors, weights, and metric. Further- more, we will use the Exhaustive Grid Search technique to optimize hyper-parameters after slicing the best result from initial training and test score between a specified specific range with an n-neighbors range. The KNN algorithm is given in Algorithm 1.

Neighbors: Based on the values we previously computed, choose the optimal k.

Weights: Examine whether giving the data points weights will help the model. While "distance" weighs points according to the inverse of their distances, which means that closer points will have greater weight than further points, "uniform" applies no weight.

Metric: The distance metric to be used will calculate the similarity.

Algorithm 1: KNN Classifier Algorithm

Input: AIT Log Data Set V1.0.
Output: Alert to user through web app.

1: Collect Apache logs from the two servers mail.cup.com and mail.spiral.com
2: Processing the data according to features and label data
3: Taking mail.cup.com server logs as Training set and other as Test set
4: Process the data into selective features with columns and convert to dataframe
5: Generate column with filed name and load dataset in it.
6: **if** Object type Dataset **then**
7: Make additional columns 0 to *n_classes* 1 with feature names as prefix using one hot encoding
8: remove the original feature.
9: remove one of the generated column to avoid multi-linear collinearity
10: repeat process for all necessary labels.
11: **end if**
12: Select KNN classifier.
13: Tune the k Neighbors parameter by selecting certain radius of values
14: Choose the k values with best results
15: Initialize the grid parameters for each parameters including k neighbor's values
16: Perform Exhaustive Grid Search Technique with grid parameters
17: Get the best parameter values from Grid Search
18: Train and Test the data set with KNN
20: **if** malicious activity detected **then**
21: Generate alert.
22: Foreword it to web interface.
23: **end if**

4.1.3 Isolation Forest

Isolation Forest is a concept-based anomaly detection technique of isolation of anomaly points rather than trying to find the perfect description of regular points. The simple fact that anomaly points are far easier to isolate than regular points make this algorithm so effective and practical for real-life data.

4.1.4 Hyper-parameters Tuning of Isolation Forest

Hyper-parameters regarding the Isolation Forest include contamination, max samples, and n estimators. Furthermore, we use the Exhaustive Grid Search technique for hyper-parameters optimization.

Contamination: the proportion of outliers in the dataset. (which we calculated from our labeled dataset).

Max samples: number of samples to pick from original data to train each base estimator.

n estimators: number of trees in the ensemble

Max Features: the number of random samples it will pick from the original data set for creating Isolation trees.

4.2 Network Log Analysis

The process of analyzing network logs includes both identifying any potential malware network infiltration and analyzing the activities of the malware in the network using either of two methods: signature-based or anomaly-based detection. Known malware signatures are utilized in signature-based malware detection to identify the pathogen malware. One of the significant drawbacks of the signature-based method for detecting malware includes not being able to detect new attacks. Malware detection based on anomaly or behaviour is based on a host's harmful activity or action. Anomaly detection is so hard because it is an unsupervised problem. Anomalies are rare and different from each other. So for anomaly analysis here, the isolation forest classifier is used. Isolation Forest starts directly from outliers rather than from regular observations. An anomaly detection system called isolation forest uses isolation, or how remote a data point is from the rest of the data, to find abnormalities.

As with other outlier detection methodologies, an anomaly score is needed for decision-making. Any anomaly detection approach must have an anomaly score. Although the Isolation Tree's maximum height rises in the order of n, the average height increases in the order of log(n), making it challenging to calculate such a score h(x) in order of log(n). Any of the following terms that normalize h(x) are either not bounded or cannot be directly compared. Since iTrees and Binary Search Trees (BST) share the same structure, the calculation of average h(x) for exterior node terminations is equivalent to the BST's ineffective search. We adopt the analysis from BST to determine the average path length of the isolation Tree. Hence the anomaly score of isolation forest is defined as

$$S(x, n) = 2(E(h(x))/c(n)) \quad (1)$$

where n is the number of external nodes, c(n) is the average path length of failed searches in a binary search tree, and h(x) represents the path length of the observation x. The algorithm for Network Log Analysis is given in Algorithm 2 below.

Algorithm 2: Network Log Analysis Methodology

Input: Network Devices Logs,packet capture file (pcap).
Output: Alert to user through web app.
1: Collect pcap file and network logs.
2: Download the KDDTrain+.arff file from the NSL-KDD dataset.
3: Install the necessary packages and the KDDTrain+.arff file.
4: Give the field's name a number.
5: Make a column with the field name, then load the dataset using it.
6: **if** Unprocessed Dataset **then**
7: Label values ranging from 0 to n classes should be encoded.
8: using transform calibrate values.
9: return encoded labels.
10: repeat process for all necessary labels.
11: **end if**
12: Divide the dataset (data and labels) into the ratio of 70:30 for the train dataset and the test dataset.
13: Select Isolation Forest classifier.
14: Train dataset using train dataset.
15: **if** malicious activity detected **then**
16: Generate alert.
17: Foreword it to web interfac.
20: **end if**

4.3 Blockchain

There are several log-based enterprises-level solutions available in the market. This software use machine-learning algorithm to detect anomalies in network and operating systems, then they perform Big Data analysis and create a huge log file for storing the data. Attackers take advantage of this older technology and clear the tracks. Now, the logs generated after detecting anomalies are deleted or altered. Most of these cyber-attacks remain undetected due to companies failing to maintain proper logs after detecting anomalies. In this paper, we develop a blockchain technology solution to store anomaly-based network and error logs in an immutable storage system, maintaining the confidentiality and integrity of the logs.

We have proposed a private blockchain network model, using public and private key pair. This blockchain network works in a restrictive environment like a closed network. The public key is transferred to entities who want to add their data to the blockchain. The private key is used to access the output file from the IPFS file system. The Algorithm for Blockchain Methodology is given in Algorithm 3.

Algorithm 3: Blockchain Methodology

Input: Log data in list format.
Output: Immutable text file to user through web app.
1: Extract log data in form of lists using pandas.
2: Generate public and private key pair.
3: Enter the public key in the web portal as proof-of-work.
4: Generate Merkle Tree using the same log data.
5: Verify the last block hash with merkle tree hash for integrity check.
6: Enter the desired location for storing the immutable output file.
7: The file will be linked to the distributed storage environment for easy access.

4.3.1 Generating Keys

First, we will generate public and private key pairs. The public key will act as proof of work, and it can be transferred in an encrypted format to entities who need to add the logs to the blockchain. The private key will be used to access the immutable log file extracted from the blockchain.

4.3.2 Generating Blocks

Each block in the blockchain will have five properties: index, timestamp, log data, proof of work, and previous block hash. Here each block's log data contains three logs. The data collected from each log is converted into a string and Unicode. Finally, the Unicode value is Hashed using the SHA-256 hashing algorithm. This hash value is stored on the blockchain in a string.

4.3.3 Creating Merkle Tree

The Merkle Tree encodes the log data for easy integrity check and fault detection. The log data in the last block is hashed, then the log data in the second-last block is hashed; after this process, both hash values are combined and hashed using the SHA-256 Hashing Algorithm. This process goes on till we reach the genesis block.
For example, if we take a simple blockchain with three blocks, the first being the genesis block, and the second and third blocks will be actual log data. In the Merkle tree function, we perform hashing on each block individually, then we concatenate both hash values and again perform hashing. Here H represents hashing using the SHA-256 algorithm, GBH is the genesis block hash, PBH is the previous block hash, and CBH is the current block hash.

$$GBH = H(H(PBH) + H(CBH)) \quad (2)$$

4.3.4 Distributed File Storage System

For making the output blockchain file immutable and easily accessible by all the users, it is essential to store the file in a Distributed Environment where only specific individuals can view the log data in the blockchain using their private keys. We use the Inter Planetary File System (IPFS) for creating a decentralized network. The output file extracted from the blockchain is stored in the network node using IPFS Desktop software. The output file is stored on the server as an immutable text file. This text file is made immutable using the *chattr* command locally, and secondly, the IPFS file system makes the output file inherently immutable. Therefore, making it twice immutable and increasing its integrity furthermore, whenever a new block is generated appended, in this text file.

4.4 Flask Web App

Here in flask-based web app, we have integrated all the results from above all previous analyses and displayed them in the proper user interface that is effective and efficient in further analysis. In this implementation, all the essential details from logs and analysis of logs can be exported report format that maintains the proper chain of custody for forensic analysis. The front-end part is well developed with bootstrap four and stored in the templates folder from where the jinja template (in built-in flask framework) renders it to the user and provides dynamic values to the user as per the code. Also, the graphical results are provided to the user with proper UI to explain the results from different machine learning-algorithms.

Due to its inherent benefits, Flask is an ideal choice for incorporating machine learning models into web applications. It perfectly fits with the Python-centric nature of the majority of machine learning work as a micro web framework. Because of its lightweight design, which reduces needless costs, machine learning models can be served effectively through RESTful APIs or web services. Flask allows for freedom in the design of API endpoints and routing, allowing developers to create applications that are specifically tailored to their needs. A multitude of tools and extensions are made available by the active Flask community, simplifying the integration of machine learning components.

5. DISCUSSIONS AND RESULTS

In this section, we are presenting results for Apache Logs Analysis and Network Log Analysis.

5.1 Apache Logs Analysis

Currents section presents results for Apache Logs analysis and under different headings such as Data Set, Metric, Evaluation, and Isolation Forest.

5.1.1 Data Set

The data set is made at the Austrian Institute of Technology (AIT). In this paper, for anomaly detection, Apache access logs have been used. For training, the mail.cup.com server Apache access logs are utilized, and mail.spiral.com Apache access logs have been used for testing. A thorough explanation of the test beds generation and data set can be found in this paper [16].

The accuracy of a machine learning model may not be a reliable indicator of its performance when the number of samples in the test dataset is almost equal to that in the training dataset. To counter this problem, we have done a more comprehensive evaluation of our proposed model's performance. The description of Apache dataset used after preprocessing is presented in Table 2.

Table 2: Description of Apache dataset used after prepossessing

Total Samples	Total Features	Training	Test
248985	19	148530	100455

We calculate metrics like recall and precision in addition to accuracy. These metrics give a more thorough picture of how each class is performing in our model. To evaluate the model's performance on various data subsets, we used k-fold cross-validation. Cross-validation ensures that the performance measures of our model are reliable and are not too affected by the particular data split. When dividing the data into the training and test sets, stratified sampling was taken into account. This minimizes the possibility of skewed findings by ensuring that both sets preserve the same class distribution. Figure 3 shows the distribution of Apache Access Logs

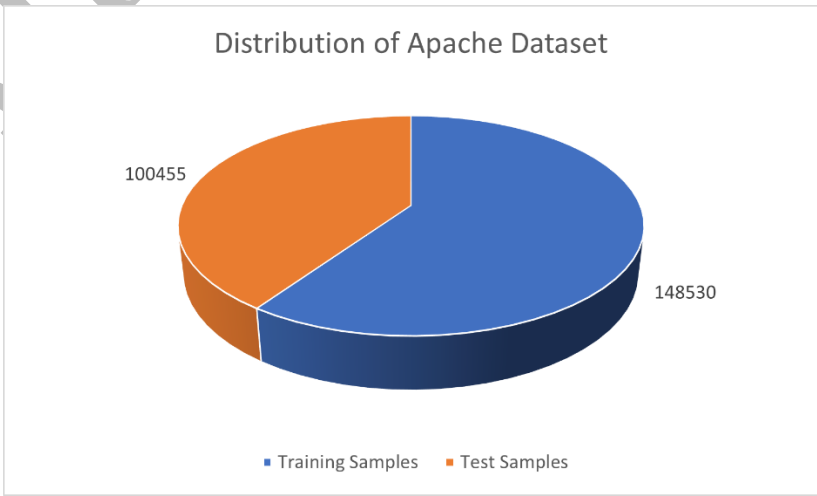


Figure 3. Distribution of apache access logs

5.1.2 Metric

Following metrics are taken into consideration while hyper-parameters are tunning.

True Positives (TP): Instances where the system successfully detects a threat or anomaly are known as true positives (TP). A true positive in the context of log analysis is when the anomaly detection algorithm successfully identifies a log entry as a real security concern or vulnerability.

True Negatives (TN): A true negative in log analysis would be the accurate determination that regular log entries do not indicate any security threat or anomaly.

False Positives (FP): A false positive in log analysis happens when the anomaly detection system incorrectly labels a regular log item as a security risk or vulnerability.

False Negatives (FN): False negatives in the context of log analysis occur when an anomaly detection algorithm neglects to signal a log entry as a security concern or vulnerability when it ought to have.

Accuracy: Measure of total correct classification of all classes in the model. Now accuracy is the most well-known performance metric, and though it also plays an essential role here because the ratio of classes in the dataset we are using is imbalanced (i.e., only 0.06 contamination), other metrics are also being used. The accuracy can be defined by the following mathematical formula

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$$

Precision: Measure of total correctness of the anomaly predictions from the classification result. The higher the precision we obtain, the fewer resources will be needed to find and countermeasure the root cause of the anomaly. The Precision can be defined by the following mathematical formula

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall: Measure the total correct classification of the actual anomaly class done in the result. The optimum classification model will bear great results in all of the mentioned three performance metrics, but that will not be in every case. In most cases and here also, we have to compromise and choose a model based on our needs. Higher in recall metric, less in precision which ultimately means more anomalies will be detected at the cost of more resources. The Recall can be defined by the following mathematical formula

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

5.1.3 Evaluation

KNN: For k neighbor parameter tunning, the training and testing accuracy after a while (at k =10) breaks from inclining and stays at a stagnant state (i.e., slope=0). k neighbor =10 is chosen and carried forward for Exhaustive Grid Search.

Following parameter was used for Exhaustive Grid Search

```
grid params = {'n neighbors' : [10],
'weights' : ['uniform','distance'],
'metric' : ['minkowski','euclidean','manhattan']
}
```

Grid Search will evaluate the model 1 x 2 x 3 = 6 times with different hyperparameters. After performing the technique, it returned with result accuracy of 93.7 percent with {'met- ric':'minkowski', 'n neighbors': 10, 'weights': 'distance'} values along each parameters. Furthermore, four features with >.5 multi collinearity was removed and ran with the same parameters value but received same result. After the Exhaustive Grid Search technique, the results were stored in a data frame. Data Insights showed that most models have high accuracy but low precision and recall. The data frame was sorted based on test accuracy, precision, and recall to ensure the

model was chosen. There were some instances where the model was under fitting and some overfitting. We sorted based on test data set recall and chose the best model (highlighted in the below Figure 3).

Figure 3 shows the performance metric of the model corresponding to the hyper parameters such that of Isolation Forest. Grid Search technique has been implemented and the best combination of hyper parameters has been properly highlighted. Recall metric is chosen because it identifies all actual anomalies (true positives) while minimizing false negatives (missed anomalies). In network security, false negatives can have serious consequences so prioritizing recall was most relevant for the performance metric in this case. Then sorting the best of 20 recall results, overall best performance (other metrics shown in figure) has been chosen for the optimal model. In Figure 3, we can see how the model performs for different hyper parameters that are used in Isolation Forest, which is a method for anomaly detection.

We use Grid Search technique to find the best combination of hyper parameters that maximizes the performance metric. The performance metric we chose is recall, which is the ratio of true positives (anomalies that are correctly identified by the model) to the total number of actual anomalies in the data. Recall is a crucial metric for network security, because we want to make sure that the model does not miss any anomalies that could pose a threat or cause damage. False negatives (anomalies that are missed by the model) are more harmful than false positives (normal data points that are wrongly classified as anomalies) in this case. Therefore, we sorted the results by recall and picked the best one as the optimal model. The figure also shows the other metrics, such as precision and accuracy for both training and testing, for the optimal model. We wanted a model that could not only learn the anomalies and achieve good results on the training data, but also generalize well to unseen data in the testing data.

Brief coding details is as follows:

- **iforest_result**_df is a dataframe containing the results of the Isolation Forest algorithm.
- **sort_values** is a function used to sort the dataframe by a specific column.
- **by='testing_recall'** specifies that the sorting should be done based on the values in the 'testing_recall' column.
- **ascending=False** specifies that the sorting should be done in descending order.
- The table contains various parameters of the algorithm such as n_estimators, max_samples, max_features, bootstrap, n_jobs, and their corresponding values.

5.1.5 Comparison Analysis

Comparison is made between KNN and Isolation Forest Algorithm performance in AIT Log Data Set V1.0. The description Data set is described in Figure 4. Both Training and Testing and each algorithm are done in the same Google Colab environment. The CPU total time is taken as time measure using command. From the below Figure, it can be observed that KNN did outstanding in the Training phase with almost perfect in all three metrics with minimal time effort but failed to replicate its Test data set. The isolation Forest algorithm did super well in the Training dataset and even better on the Test data set. Though KNN took no time during Training the data, Isolation Forest was way faster in predicting the test data. Thus, we could conclude that Isolation Forest performs better on anomaly detection in an accurate data set with a higher result more efficiently. The results of Comparison of KNN and Isolation Forest Algorithm performance in AIT Log Data Set V1.0 is shown in Table 3 below.

Table 3: Comparison of KNN and Isolation Forest Algorithm performance in AIT Log Data Set V1.0

Algorithm	**For Training Data**			**For Test Data**			**Training Time**	**Testing Time**
	Accuracy	**Precision**	**Recall**	**Accuracy**	**Precision**	**Recall**		
KNN	0.99	0.99	0.99	0.94	0.99	0.95	46.7ms	4min 41s
Isolation Forest	0.99	0.85	0.95	0.98	0.9	0.91	2.9s	1.32s

```
iforest_result_df.sort_values(by='testing_recall', ascending=False).head(20)
```

	contamination	n_estimators	max_samples	max_features	bootstrap	n_jobs	training_accuracy	training_precision	training_recall	testing_accuracy	testing_precision	testing_recall
413	0.06	50	200	10	True	40	0.983639	0.750806	0.948273	0.975947	0.762273	0.923332
318	0.06	10	200	10	False	30	0.981599	0.725735	0.944956	0.971995	0.726452	0.923030
405	0.06	50	200	3	True	40	0.981741	0.726349	0.948273	0.980875	0.812542	0.923030
334	0.06	10	700	10	False	30	0.987053	0.800179	0.946313	0.979760	0.802211	0.920012
365	0.06	30	200	10	True	40	0.990089	0.847970	0.947972	0.988611	0.908860	0.919559
381	0.06	30	700	10	True	40	0.896922	0.007379	0.009802	0.952243	0.588434	0.918352
428	0.06	50	700	10	True	30	0.898430	0.004919	0.006334	0.941451	0.532610	0.918201
383	0.06	30	700	10	False	40	0.898894	0.004959	0.006334	0.948629	0.568492	0.918201
382	0.06	30	700	10	False	30	0.900847	0.006341	0.007842	0.951635	0.585000	0.918201
399	0.06	50	100	10	False	40	0.897009	0.004913	0.006485	0.953557	0.596099	0.917899
296	0.06	10	100	5	True	30	0.908307	0.005939	0.006334	0.957688	0.621398	0.917748
328	0.06	10	700	5	True	30	0.898894	0.004842	0.006183	0.948599	0.568371	0.917748
430	0.06	50	700	10	False	30	0.901998	0.007947	0.009652	0.957449	0.619951	0.917296
380	0.06	30	700	10	True	30	0.904287	0.006763	0.007842	0.955558	0.608187	0.917145
319	0.06	10	200	10	False	40	0.905419	0.006914	0.007842	0.952581	0.590534	0.916994
366	0.06	30	200	10	False	30	0.902443	0.006406	0.007691	0.952531	0.590335	0.916239
297	0.06	10	100	5	True	40	0.898087	0.004889	0.006334	0.950162	0.577069	0.915334
317	0.06	10	200	10	True	40	0.926822	0.004443	0.002865	0.971228	0.723927	0.911410
95	0.02	30	700	10	False	40	0.944428	0.009668	0.002413	0.984031	0.855968	0.911259
309	0.06	10	200	3	True	40	0.922977	0.005958	0.004373	0.944288	0.546623	0.911259

Figure 4. Result from exhaustive grid search technique of isolation forest

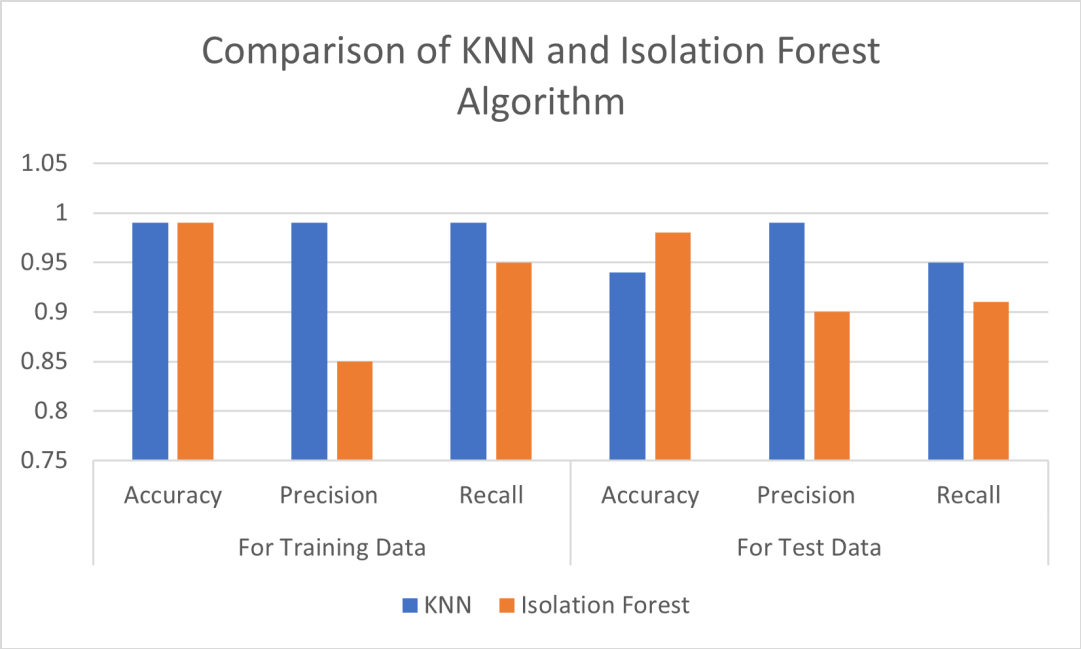


Figure 5. Comparison of KNN and isolation forest algorithm

Figure 5 shows the comparison of KNN and Isolation Forest Algorithm. Both algorithms produced great levels of recall, precision, and accuracy for the training set of data, with KNN and Isolation Forest exceeding 99% accuracy. This demonstrates that both models had outstanding performance in classifying occurrences in the training dataset. Furthermore, KNN demonstrated somewhat higher precision (99%) than Isolation Forest (85%), indicating that KNN made fewer erroneous positive predictions during training. Isolation Forest generalizes well to new, unseen data because it demonstrated a little greater accuracy on test data (98%) compared to KNN (94%). With KNN obtaining a precision of 99% and Isolation Forest at 90%, both algorithms showed great precision on the test data, demonstrating their capacity to produce precise positive predictions. However, compared to Isolation Forest, KNN had a somewhat greater recall (95%) rate. Both Figures 5 and 6 are histograms where default parameters used are n_estimators and max_samples. n_estimators is the number of trees to be used in the forest. Since Random Forest is an ensemble method comprising of creating multiple decision trees, this parameter is used to control the number of trees to be used in the process.

The max_features on the other hand, determines the maximum number of features to consider while looking for a split. So, X-axis in Figure 4 is isolation forest scores and Y-axis is max count. We use the decision function of isolation forest to provide a score to the normal training set, and then examine the results in a plot. As decision function is a measure of how simple a point is to describe, we would like to separate out simple points from complicated points by picking a numerical cut-off that gives clear separation.

5.2 Network Log Analysis

Hereafter examining our data to see that most of the traffic is normal, as expected, but a small amount is abnormal. The problem is highly imbalanced. Consequently, this problem is a promising candi- date for an anomaly detection approach, transforming all non-normal traffic into a single class named anomalous. As discussed earlier, for training the isolation forest model, we have used default parame- ters, n_estimators, max_samples, etc., as 'bootstrap': False, 'contamination': 0.04825763270848995, 'max_features': 1.0, 'max_samples': 'auto', 'n_estimators': 100, 'n_jobs': None, 'random_state': None, 'verbose': 0, 'warm_start': False

The c(n) denotes the average path length, which is defined as

$$c(n) = 2H(n-1) - (2(n-1)/n) \quad (3)$$

Here, we used the decision function of the isolation forest to provide a score for the normal and abnormal (anomalous) training set and then examined the results. The Figure 6 shows the Normal Observations, while the Figure 7 shows the Abnormal Observations.

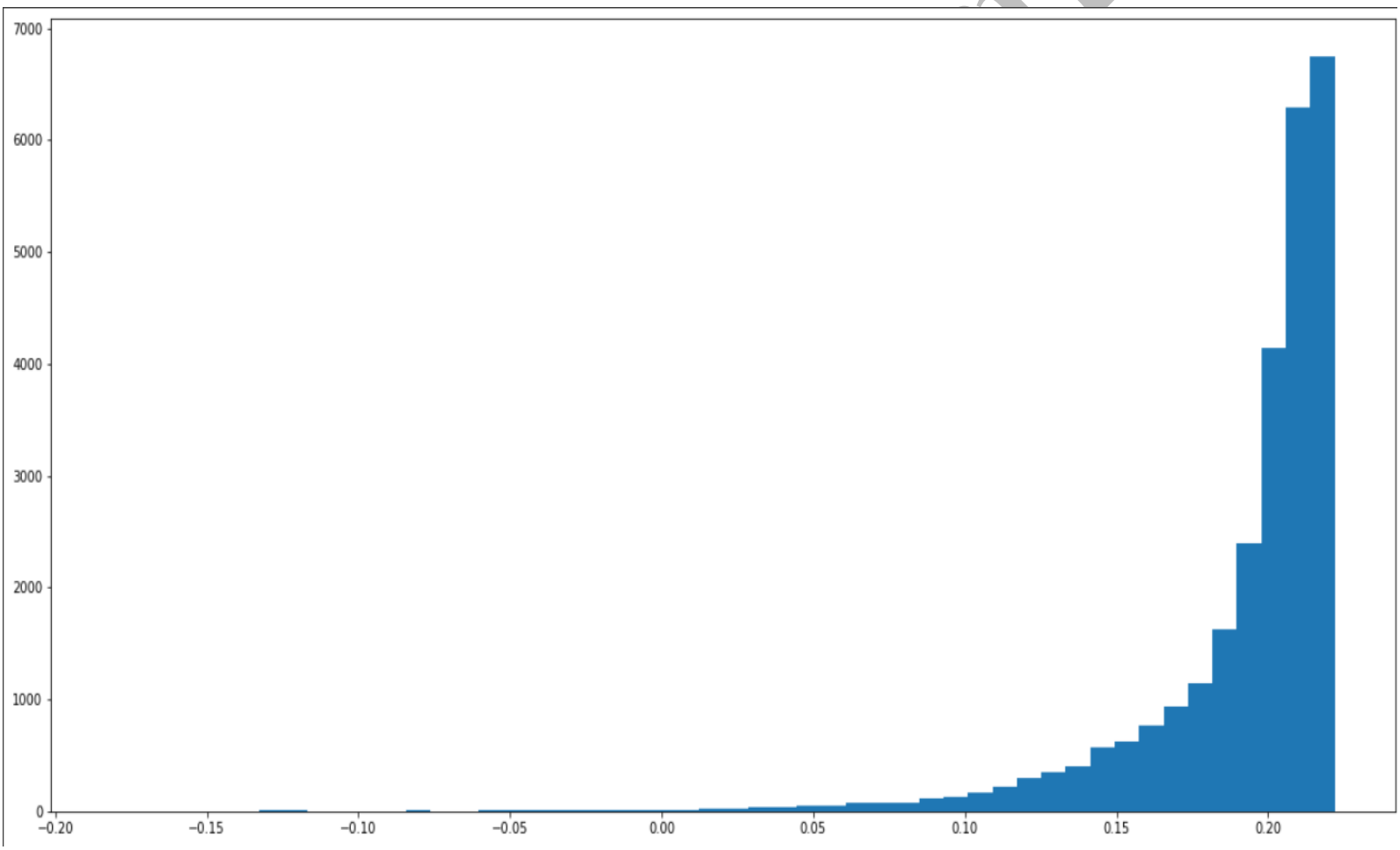


Figure 6. Normal observations

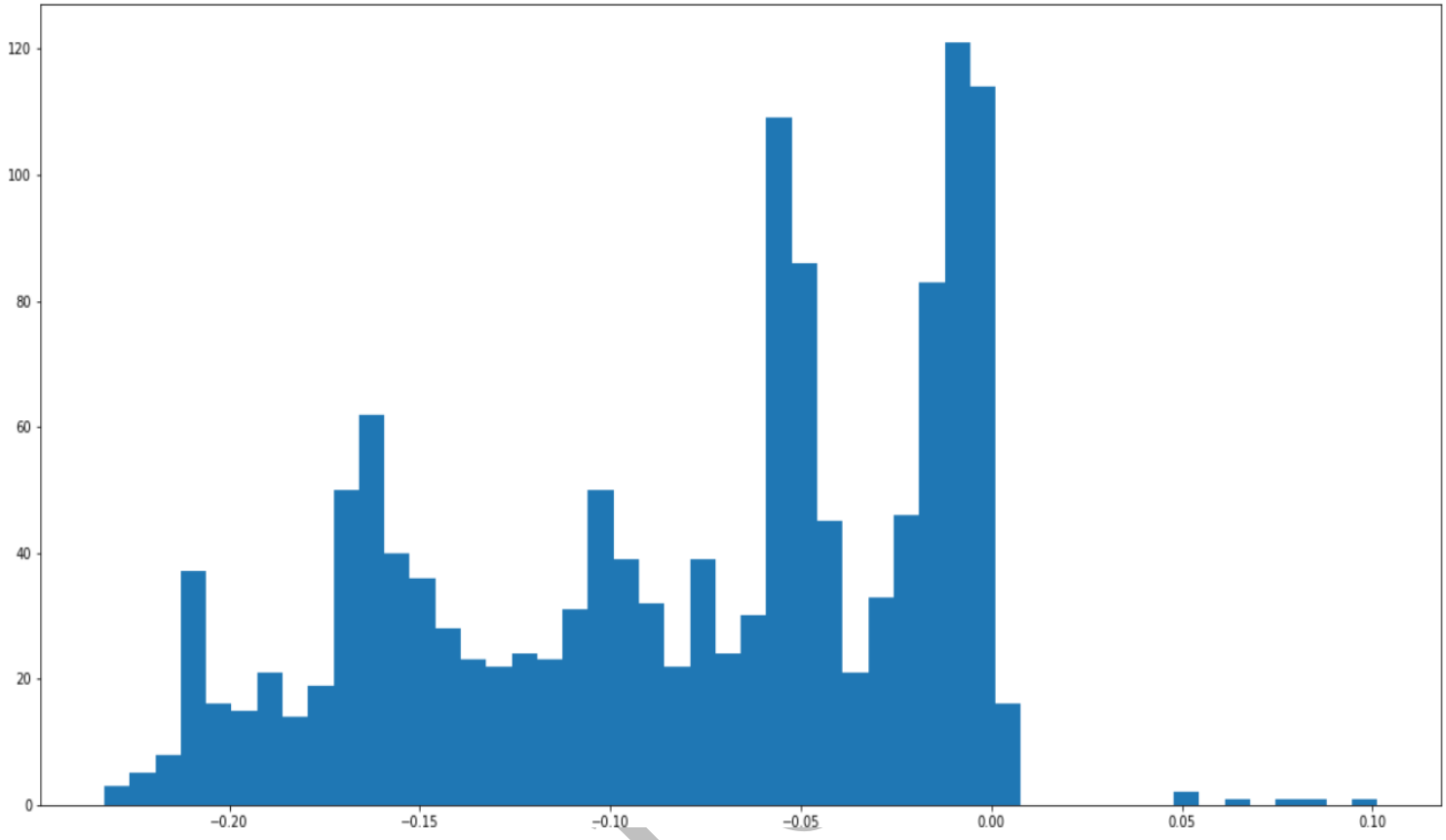


Figure 7. Abnormal observations

Knowing that the decision function measures how simple the point is to describe, we would like to separate simple points from complicated points by picking a numerical cut-off that gives clear separation. Using a visual examination of anomaly scores s, we can make the following assessment:

1. If an instance return value of s very close to 0.01, then it is definitely an anomaly.
2. If an instance has as much greater than 0.01, then it is quite safe to be called as a normal instance,
3. If an instance return s = 0.1, then the entire sample does not really have any distinct anomaly present.

Therefore, from above observation it is confirmed that the cutoff value chosen should be 0.01.

Table 4: Distribution of Anomalies

	Anomaly	Normal
Testing data	597	11775
Results	595	23

Finally, we can use our model to make predictions and provide an assessment of its performance. As given in Table 4, the model could pick up 595 anomalies out of 597 without triggering too many false positives that are instances of regular traffic, which are 23, so the absolute accuracy of the given model is 99.67% which is very high, considering the uniqueness of anomalies. Isolation Forest techniques have been shown to be quite good in detecting anomalies and are computationally considerably more efficient. Despite its benefits, some drawbacks include the fact that the final anomaly score depends on the contamination parameter supplied during model

training. This suggests that in order to make more accurate predictions, we need to be aware of the proportion of anomalous data up front. Similar to how branching happens, the model has a bias.

5.3 Blockchain

Figure 8 shows an example of network log data stored in the blockchain. When we consider the network logs, each has the following data, IP address, Timestamp, Time zone, the Request method, Status code, Byte size, Referrer, and User-Agent.

```
root@kali:~/Desktop/ITTCP# openssl req -new -x509 -newkey rsa:2048 -keyout Privkey.ou
t -pubkey -out Pubkey.out -days 365 -nodes -sha256
Generating a RSA private key
.....................................................+++++
.................................+++++
writing new private key to 'Privkey.out'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

Figure 8. Generating public and private key pair

The above Figure 8 explains how we generated public and private key pairs using the OpenSSL command on Linux operating system locally. The public key is transferred to entities who want to add their data to the blockchain. The private key is used to access the output file from the IPFS file system. The above Figure 7 depicts a general scenario where the machine learning algorithm detects malicious network log data. The network log data is stored in the transaction part.

```
Genesis block:

[{'index': 1, 'timestamp': 1645641270.1562915, 'transactions': [], 'proof': 100,
'previous_hash': 'Caught with wisdom'}, {'index': 2, 'timestamp':
1645641270.1562965, 'transactions': [{'IP_address': '192.168.10.190', 'Time_stamp':
'29/Feb/2020:00:00:02', 'Time_zone': '0000', 'Request_method': 'GET', 'Status_code':
'200', 'Byte_size': '2532', 'Referer': '-', 'User_agent': 'Mozilla/5.0'}, {'IP_address':
'192.168.10.4', 'Time_stamp': '29/Feb/2020:00:00:09', 'Time_zone': '0000',
'Request_method': 'POST', 'Status_code': '200', 'Byte_size': '402', 'Referer':
'http://mail.cup.com/kronolith/', 'User_agent': 'Mozilla/5.0'}, {'IP_address':
'192.168.10.190', 'Time_stamp': '29/Feb/2020:00:00:12', 'Time_zone': '0000',
'Request_method': 'POST', 'Status_code': '200', 'Byte_size': '402', 'Referer':
'http://mail.cup.com/kronolith/', 'User_agent': 'Mozilla/5.0'}], 'proof': 12345,
'previous_hash':
'95d238c9e86af3cd69b73de243df7d5ade08f4d2ed1c7ac1e139bee2b85088d9'}]

The above output is also stored in an immutable file 'output.txt'
```

Figure 9. Blockchain in simple text format

Lastly, we compare the hash value of the last block from the blockchain and the Merkle tree hash value for checking the integrity of the data. After an integrity check, the data is stored in the IPFS file system. The results of Blockchain after applying Merkle Hashing Function are shown in Figure 9.

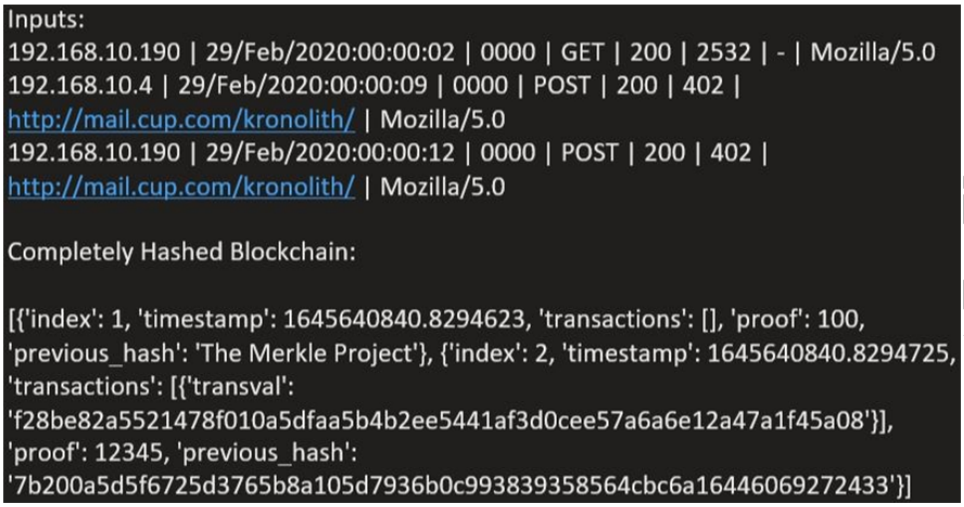


Figure 10. Blockchain after applying merkle hashing function

5.4 Web App

We also have integrated all the machine learning algorithms and blockchain results into a simple, user-friendly, easy-to-use Web Application using HTML, CSS, and Flask. This Web Application has four sections: Info, File Upload, Log Analysis, and View Report. The Info tab gives the user an in-depth idea of the project and how to use the application effectively. The File upload section is where we can upload our log files. The Log Analysis tab contains three more subsections, Error Log analysis, Access Log Analysis, and Network Log Analysis; the user can select and use any section as per the originality of the log file. The Figure 9 shows the descriptive Error Log overview and results

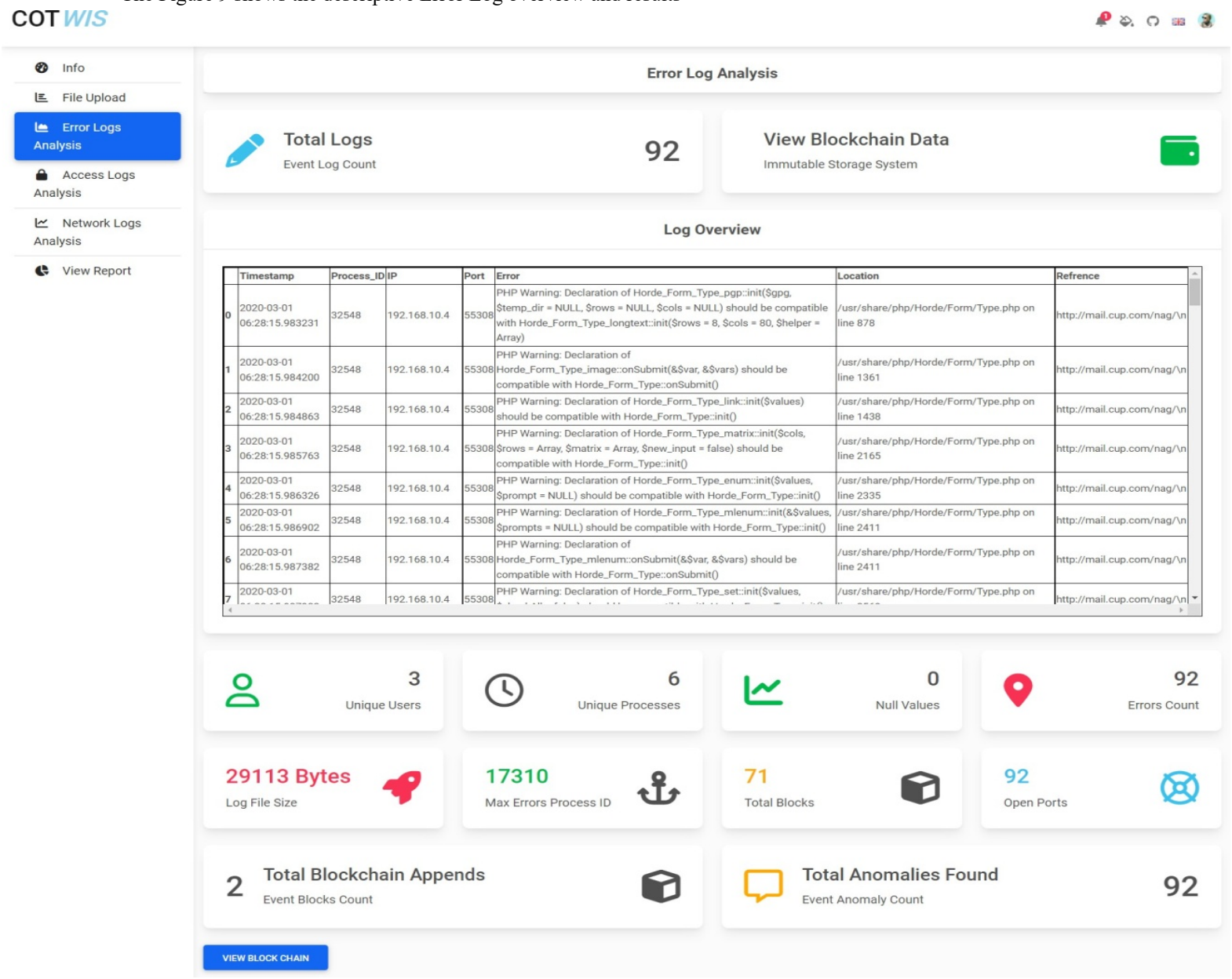


Figure 11. Descriptive error log overview and results

Figure 11 is the screenshot of the Web App when we upload an Error Log file. The Web app provides a descriptive Log overview, Anomaly information, and Blockchain data, representing malicious activities in Pie chart format.

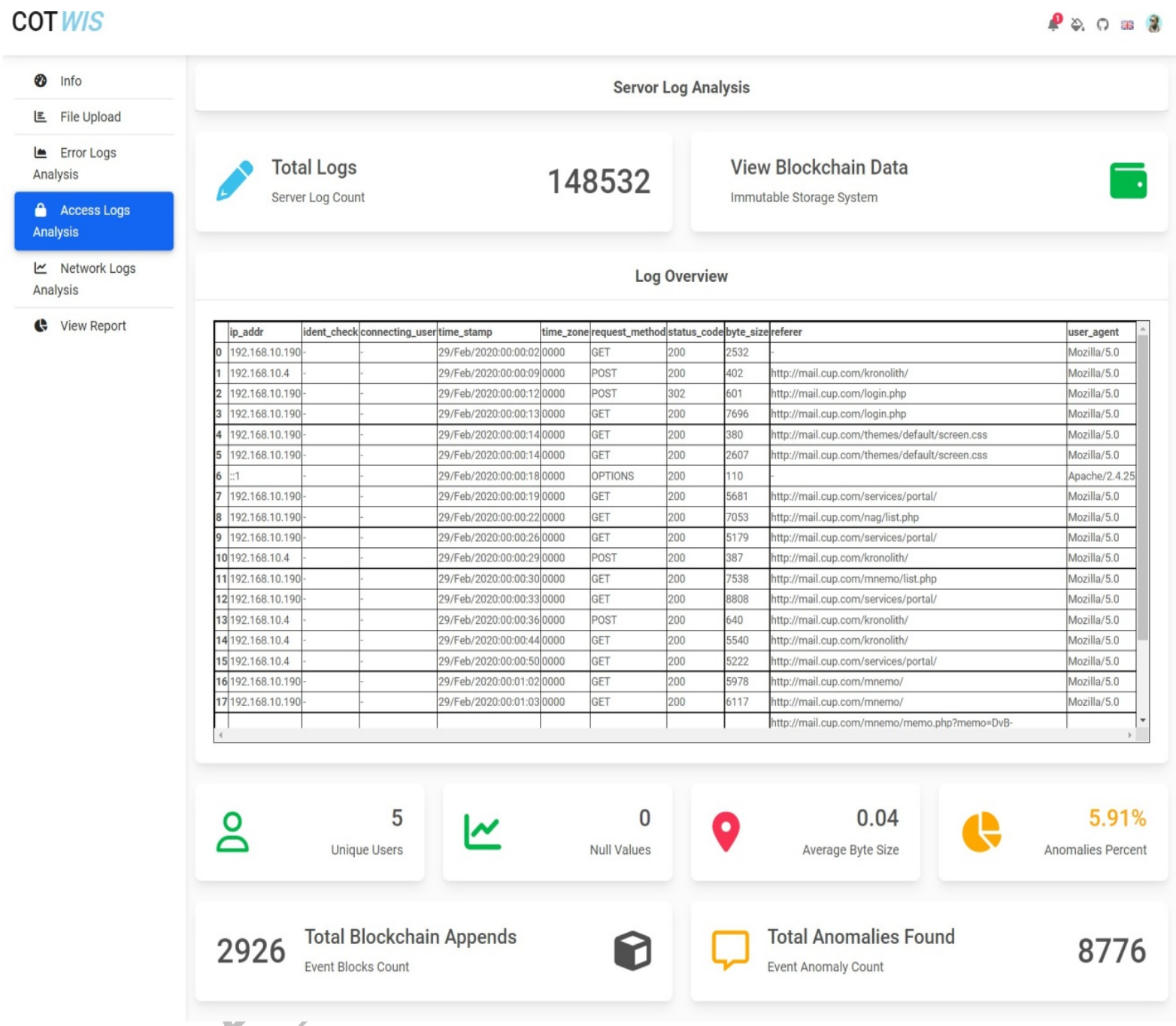
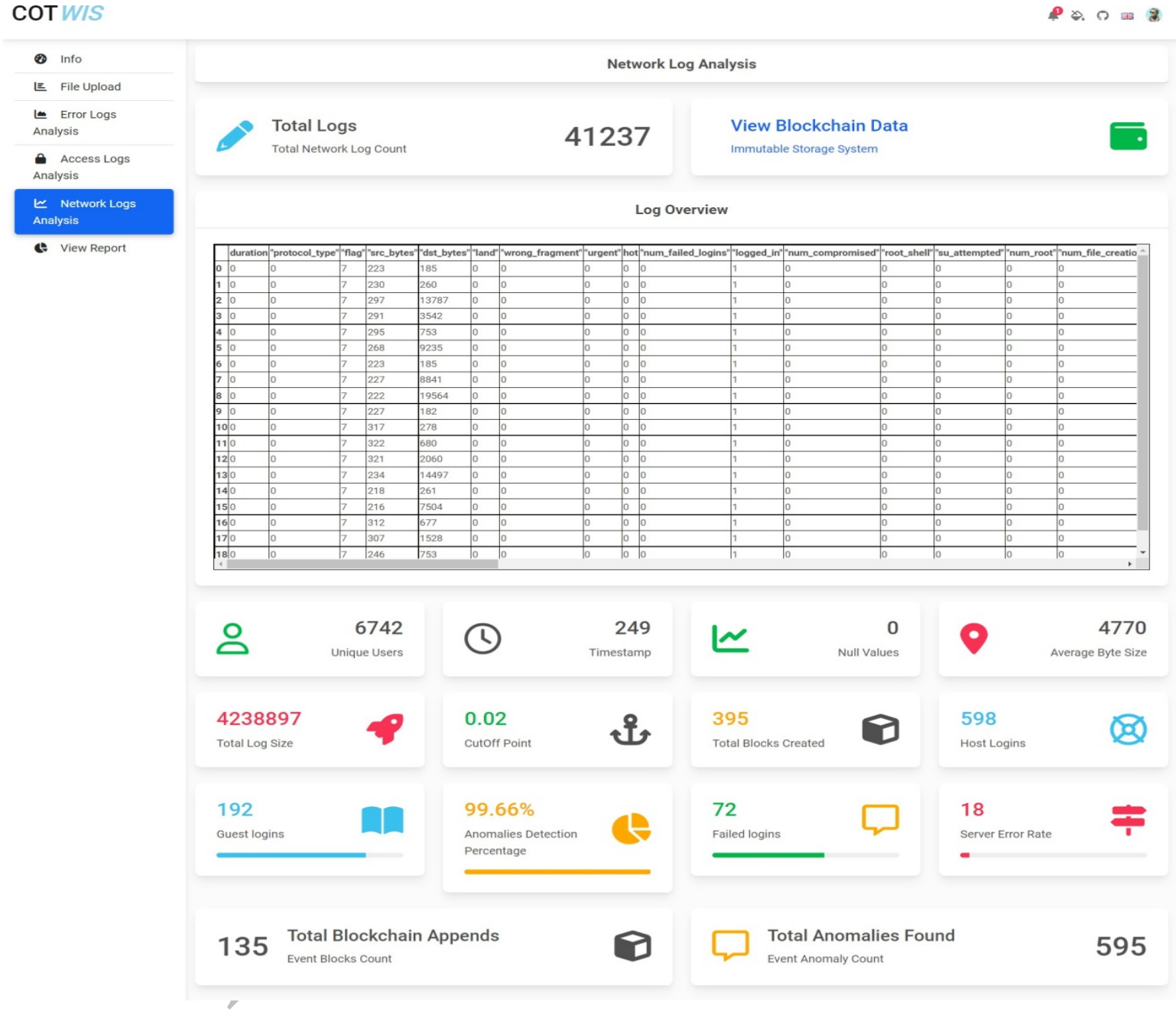


Figure 12. Descriptive access log overview and results

Figure 12 shows the screenshot of the Web App when we upload a Server Access Log file. The Web app provides a descriptive Log overview, Anomaly information, and Blockchain data, with information about malicious activities. Figure 13 shows the screenshot of the Web App when we upload a Network Log file. The Web app provides a descriptive Log overview, Anomaly information, and Blockchain data, with the graphical representation of malicious activities. The whole idea of storing the machine learning result in blockchain is to make the data confidential, tamper-proof, and limit the accessibility of malicious logs.

Several algorithms are proposed for making the malicious log data confidential, but most of them are susceptible to attacks such as Man-in-the-middle, Log4j, Log forging, Cross-site scripting, and injection. The proposed model is more focused on

increasing the log data's privacy and immutability. The proposed model is adept at preventing confidential data from attacks such as identity theft, system hacking, DNS Tunneling, Malware, and SQL injection. The double integrity check and immutable storage help to prevent spoofing attacks.



	duration	"protocol_type"	"flag"	"src_bytes"	"dst_bytes"	"land"	"wrong_fragment"	"urgent"	hot	"num_failed_logins"	"logged_in"	"num_compromised"	"root_shell"	"su_attempted"	"num_root"	"num_file_creatio
0	0	0	7	223	185	0	0	0	0	0	1	0	0	0	0	0
1	0	0	7	230	260	0	0	0	0	0	1	0	0	0	0	0
2	0	0	7	297	13787	0	0	0	0	0	1	0	0	0	0	0
3	0	0	7	291	3542	0	0	0	0	0	1	0	0	0	0	0
4	0	0	7	295	753	0	0	0	0	0	1	0	0	0	0	0
5	0	0	7	268	9235	0	0	0	0	0	1	0	0	0	0	0
6	0	0	7	223	185	0	0	0	0	0	1	0	0	0	0	0
7	0	0	7	227	8841	0	0	0	0	0	1	0	0	0	0	0
8	0	0	7	222	19564	0	0	0	0	0	1	0	0	0	0	0
9	0	0	7	227	182	0	0	0	0	0	1	0	0	0	0	0
10	0	0	7	317	278	0	0	0	0	0	1	0	0	0	0	0
11	0	0	7	322	680	0	0	0	0	0	1	0	0	0	0	0
12	0	0	7	321	2060	0	0	0	0	0	1	0	0	0	0	0
13	0	0	7	234	14497	0	0	0	0	0	1	0	0	0	0	0
14	0	0	7	218	261	0	0	0	0	0	1	0	0	0	0	0
15	0	0	7	216	7504	0	0	0	0	0	1	0	0	0	0	0
16	0	0	7	312	677	0	0	0	0	0	1	0	0	0	0	0
17	0	0	7	307	1528	0	0	0	0	0	1	0	0	0	0	0
18	0	0	7	246	753	0	0	0	0	0	1	0	0	0	0	0

Figure 13. Descriptive network log overview and results

6. CONCLUSIONS

As malware is getting more advanced and persistent, their new obfuscation method is evading detection of malware. With all these advancements in persistent malware attacks, there is a need for a proper detection methodology that can reduce the triage time and hence help isolate the spread of malware and reduce the impact of the infection. To detect these threats effectively, we have analyzed the different types of logs, utilized multiple machine learning algorithms to detect the new threats accurately, and proposed a system to protect the logs from mutation or deletion from attackers and store them in

distributed immutable storage. Future work includes comparing more Machine Learning algorithms, preferably LSTM Neural Network with a high-performance metric with efficient computational resources. Including the Identity and Access Logs for insider threat detection in the network log analysis framework. Comparing specific log data and machine learning algorithms for event correlation and better root cause analysis and creating a public cloud service with ubiquitous access for effective and easy log management and analysis.

Conflict of Interest: The authors declare no conflict of interest.
Funding information: This work received no funding.

APPENDIX

A. Web Application Details

In this section, we provide additional details about the web application developed for this research. The GitHub link contains code for installing the web application and necessary software. The application is accessible at the following URL:
Web Application: *https://github.com/d3M0N-wq/cotwis*

B. Abbreviations

Here are the abbreviations used throughout the paper for clarity and conciseness:
AI: Artificial intelligence
AIT: Austrian Institute of Technology
BST: Binary Search Trees
c(n): Average path length of failed searches in a Binary Search Tree
CBH: Current Block Hash
GBH: Genesis Block Hash
h(x): Path length of the observation x
IDS: Intrusion Detection System
IPFS: Inter Planetary File System
PaaS: Platform as a service
PBH: Previous Block Hash
PoW: Proof-of-Work
S(x, n): Anomaly score

REFERENCES

[1] Simoes, V., Maniar, H., Abubakar, A., & Zhao, T. (2022). Deep Learning for Multiwell Automatic Log Correction. In SPWLA 63rd Annual Logging Symposium. OnePetro. https://doi.org/10.30632/SPWLA-2022-0070
[2] Oliner, A., Ganapathi, A., & Xu, W. (2012). Advances and Challenges in Log Analysis. Communications of the ACM, 55(2), 55-61. https://doi.org/ 10.1145/2076450.2076466
[3] Albahar, M., Alansari, D., & Jurcut, A. (2022). An Empirical Comparison of Pen-Testing Tools for Detecting Web App Vulnerabilities. Electronics, 11(19), 2991. https://doi.org/10.3390/electronics11192991
[4] Candel, J. M. O., Gimeno, F. J. M., & Mora Mora, H. (2023). Serverless Security Analysis for IoT Applications. In International Conference on Ubiquitous Computing and Ambient Intelligence (pp. 393-400). Springer. https://doi.org/10.1109/I2CT.2018.8529465
[5] Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z., & Lyu, M. R. (2019). Tools and benchmarks for automated log parsing. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) (pp. 121-130). IEEE. https://doi.org/ 10.1109/ICSE-SEIP.2019.00021
[6] Behera, A., Panigrahi, C. R., & Pati, B. (2022). Unstructured Log Analysis for System Anomaly Detection—A Study. In Advances in Data Science and Management (pp. 497-509). Springer. https://doi.org/10.1007/978-981-16-5685-9_48
[7] Chen, Q. X., & Chang, X. H. (2022). Resilient Filter of Nonlinear Network Systems with Dynamic Event-Triggered Mechanism and Hybrid Cyber Attack. Applied Mathematics and Computation, 434, 127419. https://doi.org/10.1016/j.amc.2022.127419
[8] Rout, B., & Natarajan, B. (2022). Impact of cyber-attacks on distributed compressive sensing based state estimation in power distribution grids. International Journal of Electrical Power & Energy Systems, 142, 108295.

https://doi.org/10.1016/j.ijepes.2022.108295
[9] Ghiasi, M., Niknam, T., Wang, Z., Mehrandezh, M., Dehghani, M., & Ghadimi, N. (2023). A Comprehensive Review Of Cyber-Attacks And Defense Mechanisms For Improving Security In Smart Grid Energy Systems: Past, present and future. Electric Power Systems Research, 215, 108975. https://doi.org/10.1016/j.epsr.2022.108975
[10] Abdullahi, M., Baashar, Y., Alhussian, H., Alwadain, A., Aziz, N., Capretz, L. F., & Abdulkadir, S. J. (2022). Detecting Cybersecurity Attacks in Internet of Things Using Artificial Intelligence Methods: A Systematic Literature Review. Electronics, 11(2), 198. https://doi.org/10.3390/electronics11020198
[11] Zhang, X., Xu, Y., Lin, Q., Qiao, B., Zhang, H., Dang, Y., & Zhang, D. (2019). Robust Log-Based Anomaly Detection on Unstable Log Data. In Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 807-817). https://doi.org/10.1145/3338906.3338931
[12] Wang, B., Ying, S., & Yang, Z. (2020). A Log-Based Anomaly Detection Method with Efficient Neighbor Searching And Automatic K Neighbor Selection. Scientific Programming, 2020, 1-17. https://doi.org/10.1155/2020/4365356
[13] Wang, J., Tang, Y., He, S., Zhao, C., Sharma, P. K., Alfarraj, O., & Tolba, A. (2020). LogEvent2vec: Log Event-to-Vector Based Anomaly Detection for Large-Scale Logs in Internet of Things. Sensors, 20(9), 2451. https://doi.org/10.3390/s20092451
[14] Wang, Z., Tian, J., Fang, H., Chen, L., & Qin, J. (2022). LightLog: A Lightweight Temporal Convolutional Network for Log Anomaly Detection on The Edge. Computer Networks, 203, 108616. https://doi.org/10.1016/j.comnet.2021.108616
[15] Catillo, M., Pecchia, A., & Villano, U. (2022). AutoLog: Anomaly Detection by Deep Auto Encoding of System Logs. Expert Systems with Applications, 191, 116263. https://doi.org/10.1016/j.eswa.2021.116263
[16] W Pourmajidi, W., & Miranskyy, A. (2018). Logchain: Blockchain-Assisted Log Storage. In 2018 IEEE 11th International Conference on Cloud Computing (CLOUD) (pp. 978-982). https://doi.org/10.1109/CLOUD.2018.00150
[17] Pourmajidi, W., Zhang, L., Steinbacher, J., Erwin, T., & Miranskyy, A. (2019). Immutable Log Storage as a Service. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion) (pp. 280-281). IEEE. https://doi.org/10.1109/ICSE-Companion.2019.00114
[18] Oprea, S. V., & Bâra, A. (2021). Machine Learning Classification Algorithms and Anomaly Detection In Conventional Meters And Tunisian Electricity Consumption Large Datasets. Computers & Electrical Engineering, 94, 107329. https://doi.org/10.1016/j.compeleceng.2021.107329
[19] Huang, W. (2019). A blockchain-based framework for secure log storage. In 2019 IEEE 2nd International Conference on Computer and Communication Engineering Technology (CCET) (pp. 96-100). IEEE. https://doi.org/10.1109/CCET48361.2019.8989093
[20] Wang, H., Yang, D., Duan, N., Guo, Y., & Zhang, L. (2018). Medusa: Blockchain Powered Log Storage System. In 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS) (pp. 518-521). IEEE. https://doi.org/10.1109/ICSESS.2018.8663935
[21] Friedberg, I., Skopik, F., Settanni, G., & Fiedler, R. (2015). Combating advanced persistent threats: From network event correlation to incident detection. Computers & Security, 48, 35-57. https://doi.org/10.1016/j.cose.2014.09.006
[22] Ambre, A., & Shekokar, N. (2015). Insider Threat Detection Using Log Analysis and Event Correlation. Procedia Computer Science, 45, 436-445. https://doi.org/10.1016/j.procs.2015.03.175
[23] Reguieg, H., Benatallah, B., Nezhad, H. R. M., & Toumani, F. (2015). Event correlation analytics: scaling process mining using mapreduce-aware event correlation discovery techniques. IEEE Transactions on Services Computing, 8(6), 847-860. https://doi.org/10.1109/TSC.2015.2476463
[24] Bračevac, O., Amin, N., Salvaneschi, G., Erdweg, S., Eugster, P., & Mezini, M. (2018). Versatile Event Correlation with Algebraic Effects. Proceedings of the ACM on Programming Languages, 2(ICFP), 1-31. https://doi.org/10.1145/3236762
[25] Kotenko, I. V., Levshun, D. S., & Chechulin, A. A. (2016). Event Correlation in the Integrated Cyber-Physical Security System. In 2016 XIX IEEE International Conference on Soft Computing and Measurements (SCM) (pp. 484-486). IEEE. https://doi.org/10.1109/SCM.2016.7519820
[26] Landauer, M., Skopik, F., Wurzenberger, M., Hotwagner, W., & Rauber, A. (2020). Have it your way: Generating Customized Log Datasets with a Model-Driven Simulation Testbed. IEEE Transactions on Reliability, 70(1), 402-415. https://doi.org/10.1109/TR.2020.3031317
[27] Rahman, R. U., & Tomar, D. S. (2020). New biostatistics features for detecting web bot activity on web applications. Computers & Security, 97, 102001. https://doi.org/10.1016/j.cose.2020.102001
[28] Breier, J., & Branišová, J. (2017). A dynamic rule creation based anomaly detection method for identifying security breaches in log records. Wireless Personal Communications, 94, 497-511.https://doi.org/10.1007/s11277-015-3128-1

[29] Rahman, R. U., & Tomar, D. S. (2021). Threats of price scraping on e-commerce websites: Attack model and its detection using neural network. Journal of Computer Virology and Hacking Techniques, 17, 75-89. https://doi.org/10.1007/s11416-020-00368-6

[30] Alkawaz, M. H., Steven, S. J., Hajamydeen, A. I., & Ramli, R. (2021). A comprehensive survey on identification and analysis of phishing website based on machine learning methods. In 2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE) (pp. 82-87). IEEE. https://doi.org/10.1109/ISCAIE51753.2021.943179