sustainability

MDPI

Article

Improving Collaborative Intrusion Detection System Using Blockchain and Pluggable Authentication Modules for Sustainable Smart City

Rajeev Kumar Gupta [1], Vedant Chawla [2], Rajesh Kumar Pateriya [2], Piyush Kumar Shukla [3], Saoucene Mahfoudh [4] and Syed Bilal Hussain Shah [4,*]

[1] Computer Science and Engineering Department, Pandit Deendayal Energy University, Gandhinagar 382007, India
[2] Computer Science and Engineering Department, Maulana Azad National Institute of Technology, Bhopal 462003, India
[3] Computer Science & Engineering Department, University Institute of Technology, Rajiv Gandhi Proudyogiki Vishwavidyalaya (Technological University of Madhya Pradesh), Bhopal 462033, India
[4] School of Engineering, Computing and Informatics, Dar Al-Hekma University, Jeddah 22246, Saudi Arabia
* Correspondence: sshah@dah.edu.sa

Abstract: The threat of cyber-attacks is ever increasing in today's society. There is a clear need for better and more effective defensive tools. Intrusion detection can be defined as the detection of anomalous behavior either in the host or in the network. An intrusion detection system can be used to identify the anomalous behavior of the system. The two major tasks of intrusion detection are to monitor data and raise an alert to the system administrators when an intrusion takes place. The current intrusion detection system is incapable of tackling sophisticated attacks which take place on the entire network containing large number of nodes while maintaining a low number of login attempts on each node in the system. A collaborative intrusion detection system (CIDS) was designed to remove the inefficiency of the current intrusion detection system which failed to detect coordinated distributed attacks. The main problem in the CIDS is the concept of trust. Hosts in the network need to trust the data sent by other peers in the network. To bring in the concept of trust and implement the proof-of-concept, blockchain was used. Pluggable authentication modules (PAM) were also used to track login activity securely before an intruder could modify the login activity. To implement blockchain, an Ethereum-based private blockchain was used.

Keywords: sustainable smart city; intrusion detection system; collaborative intrusion detection system; authentication; blockchain

Citation: Gupta, R.K.; Chawla, V.; Pateriya, R.K.; Shukla, P.K.; Mahfoudh, S.; Shah, S.B.H. Improving Collaborative Intrusion Detection System Using Blockchain and Pluggable Authentication Modules for Sustainable Smart City. *Sustainability* **2023**, *15*, 2133. https://doi.org/10.3390/su15032133

Academic Editors: Dhananjay Singh, Paulo J. Sequeira Gonçalves, Pradeep Kumar Singh, Pradip Sharma and Pao-Ann Hsiung

Received: 21 November 2022
Revised: 27 December 2022
Accepted: 16 January 2023
Published: 23 January 2023

1. Introduction

According to the report of the Indian Computer Emergency Response Team (2021), more than 26,100 websites were victims of cyber-attacks in India in the year 2020 alone. This clearly indicates the need for better and more effective defensive tools. The role of blockchain in smart, sustainable cities is vital because it helps to foster the kind of trust necessary for smart cities. Blockchain should serve as the cornerstone for the development of a smart city and is a crucial assurance for the proper design and execution of the management strategy and planning scheme. A smart city naturally combines smart energy, smart transportation, smart government, and other services under the same umbrella. Decentralization and the availability of clear data place strict constraints on the big data service platform. Finding the problematic node among the hundreds of millions of nodes in a network is a time-consuming operation if a network encounters a problem or is the target of an attack. Most current Internet-of-Things networks are centralized. A huge server or centralized cloud is connected to hundreds of millions of nodes, which causes

bottlenecks in the price and computer storage capacity. Blockchain-distributed technology can guarantee that even if one or more nodes are hacked, the total network data remains trustworthy and secure. Distributed computing makes use of point-to-point computing to handle the hundreds of billions of transactions that the Internet-of-Things generates. This significantly lowers the cost of computing and storage by utilizing the computing and storage capabilities of a large number of idle devices deployed in unused locations.

In order to increase the degree of security for secure transmission and safe storage, additional protection mechanisms need to be implemented due to the privacy of numerous people involved. Blockchain has shown to be secure, dependable, and suitable for this purpose. The disaster recovery system cannot be enhanced due to the high expense of creating a data center and data storage. Therefore, a key issue at hand is how to lower storage costs while enhancing disaster recovery capabilities. Blockchain, which connects distributed and centralized services, can successfully stop an attack on the vital network infrastructure. The main objective of intrusion detection is to observe anomalous behavior either in a network or in a host. In the current scenario, the current IDS is not sophisticated enough to detect the wide variety of threats. Collaborative intrusion detection has some of the capabilities to at least detect some of those threats and send them for further processing. Based on the deployed location, IDSs can be categorized as a host-based intrusion detection system (HIDS) and network-based intrusion detection system (NIDS). A HIDS monitors the characteristics of a particular node and the system events in a node for malicious activities, whereas a NIDS monitors the network by placing packet sniffers in the network at various points. These packet sniffers pick up the data and send the data to analysis units who compare the present state of the system with that of an anomaly.

Based on the approach of the detection, an IDS can again be classified into two types: signature-based IDSs and anomaly-based IDSs. Signature-based detection detects an attack by comparing stored signatures with the observed system or network events for possible occurrences. A signature (also known as a rule) is a pattern that describes a known attack or exploit. Anomaly-based detection works by detecting large deviations between its pre-built normal profile and the observed events, and hence detects suspicious activity. A normal profile is frequently generated by observing the features of ordinary activity over time, and it might represent the regular behavior of users, network connections, and programs [1]. If an abnormal circumstance is discovered, an alert may be triggered. The main disadvantage of IDS is that it cannot detect sophisticated attacks which take place on the entire network of nodes cumulatively as they monitor only a single node or a single network. For example, if we have a series of stand-alone IDSs, they are incapable of detecting a distributed attack which takes place across multiple hosts in a network. This is because they do not have the ability to co-relate the events which take place. To address this weakness, the concept of CIDSs was introduced.

CIDSs were introduced to address the weakness of IDSs which can be seen during distributed attacks. CIDSs generally consists of several monitor units and analysis units. The monitor units jot down the information and send it to the analysis units, which process the information and make decisions based on it. Based on the architectural differences, CIDSs again can be classified into three categories as shown in Figure 1, namely: centralized, decentralized, and distributed [2]. A centralized CIDS is the most basic version and the simplest one. However, it is prone to a single point of failure (SPoF) and performance bottleneck in cases of network overload. In distributed CIDSs, the SPoF disadvantage is somewhat removed but it still has disadvantages. In this, information is lost at each level of the hierarchy and hence is somewhat unreliable. In a decentralized CIDS, each node behaves as both monitor and analysis units. It is a P2P architecture which facilitates data sharing, correlation, and aggregation of data across all nodes. However, CIDS also has some disadvantages. The network cost incurred is very high as all nodes are in constant communication with each other. Furthermore, the idea of trust is very important among these nodes. To remove the trust issue among the nodes, the concept of blockchain was

introduced. This problem, along with CIDS, is discussed in a detailed manner further in the later stages. Figure 1 illustrates the architecture of CIDS.

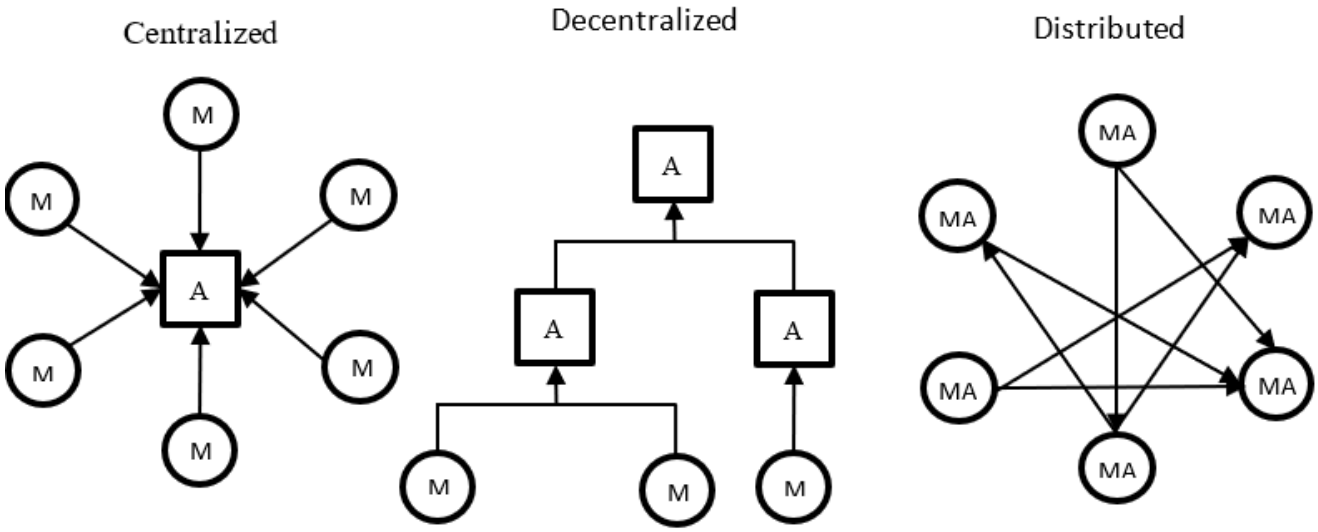

Figure 1. Overview of a CIDS architecture.

The main contributions of this paper are:

Proposed system will be able to detect coordinated distributed attacks.

Hosts in the network need to trust the data sent by other peers in the network. To bring in the concept of trust and implement the proof-of-concept, blockchain was used.

Pluggable authentication modules (PAM) were also used to track login activity securely before an intruder could modify the login activity.

To implement blockchain, an Ethereum-based private blockchain was used

This paper is organized as follows: Section 2 discusses the basics of blockchain along with the different components of blockchain, Section 3 discusses different existing intrusion detection systems, and Section 4 explains the proposed improved collaborative IDS which uses blockchain and pluggable authentication modules. Section 5 discusses the result analysis and Section 6 summarizes the entire work and gives direction for future work.

2. Blockchain

Blockchain can be defined as a distributed peer-to-peer network of blocks. Each block is linked to the previous block using a cryptographic hash. Blockchain technology has been applied to several fields such as healthcare, education, energy, etc. There are three types of blockchain ledgers which are currently in use: public, consortium, and private. Public blockchains (such as Ethereum) are accessible to anyone with internet access and anyone can read the blockchain and maintain the blockchain ledger, i.e., there is no membership mechanism in place. The consortium blockchains (such as the Hyperledger Fabric) are maintained by an established body which grants access to others and has a pre-defined consortium of peers maintaining the chain. Private blockchains are maintained by one entity that provides access to others and there is no consensus process.

2.1. Block Structure

The most basic definition of blockchain is that it is a chain of blocks with each block connected to the one before it with the help of a mathematical relationship. The block in itself is a container of data. The main premise underlying blockchain is that each block contains a unique self-identifying hash that ensures the chain's integrity. The hash of the block index, data, timestamp, and, of course, the hash of the previous block hash, make up this self-identifying hash. It also contains a record of the transactions, called a ledger, which took place during the time of blockchain production. As each block references the

one before it, there is a record of all transactions that took place prior to the current block's generation. The Figure 2 shows the structure of the block chain generation.

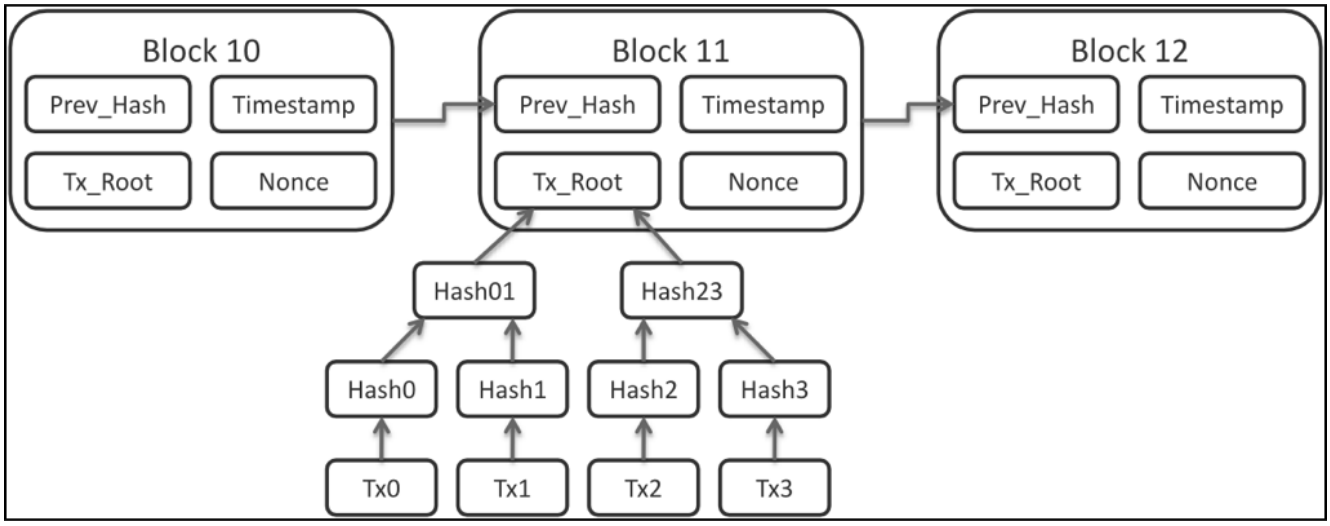

Figure 2. Structure of a blockchain [3].

2.2. Consensus

Consensus algorithms allow the participants to reach an agreement about the state of the network without the presence of a central authority. Any blockchain model is only as effective as its consensus model. There are two major consensus algorithms in the blockchain world which are the proof-of-work and the proof-of-stake. The proof-of-work algorithm is implemented by Bitcoin, whereas the proof-of-stake algorithm is implemented by Ethereum and is currently in deployment. Proof-of-work is founded on the premise that a participant establishes its identity by demonstrating that it worked. In the case of Bitcoin, each participant's purpose is to find a hash value that is less than a number set by the network as the difficulty level. This is an example of a computational puzzle where a brute-force, guess-and-check method is the most effective way to solve it. This process, known as mining, ensures that no single player has an edge in creating the next block. As a result, miners are not required to provide any authentication or a-priori knowledge. The chances of a block being modified successfully diminishes exponentially with the size of the blockchain. Proof-of-work, on the other hand, is subject to the 51 percent attack, in which a coalition with more than half of the possible mining power can insert blocks into the blockchain. To counter this, Ethereum built a new consensus algorithm called proof-of-stake. Proof-of-stake relies on a group of validators with a financial stake in the network voting and proposing the next block in turn. The method chooses validators for block production in a pseudo-random manner, preventing advance knowledge of when a specific participant would create a block. The quantity of cryptocurrency, or stake, that a participant has determines his or her chances of being chosen as a validator. While there are several drawbacks to this method of implementation, it does address the 51 percent attack problem which the proof-of-work had and is currently being developed by Ethereum. Table 1 illustrates the fields of block structure in blockchains and their uses.

Table 1. Fields in a blockchain [4].

Name	Description
Version	It refers to the protocol's identifying rules.
Timestamp	Nodes can use timestamps to properly change the mining difficulty for each block creation period. Timestamps allow the net- work to calculate how long it takes to extract blocks during a certain time period and alter the mining difficulty parameter accordingly.
Previous Hash	Used to link the previous block with the current block in the chain.
Target	It defines the difficulty level of the consensus algorithm.
Nonce	Nonce is an abbreviation for "number only used once" which is added to a hashed block in a blockchain that when rehashed, fits the difficulty level limitations. In order to receive cryptocurrency, blockchain miners must solve for a nonce.
Merkle Root	A Merkle Root is the hash of all the hashes of all the legitimate transactions that make up a block.
Hash	Hashing transaction occurs by Merkel tree, where each node is related with its parent node; therefore, if the transaction is modified, then it will affect all hash trees from the leaf node to the Merkle Root, respectively.

3. Literature Survey

Kanth et al. [5] implemented a collaborative intrusion detection system capable of recording login activity via a private blockchain-based ledger and hence it is immutable. In initial stages the authors were successfully capable of proving that blockchain-based CIDSs were a viable method to detect doorknob-rattling attacks and hence can prevent any act of an intruder trying to modify the activity records. The author [6] uses CPU utilization as a metric to accurately determine whether an intrusion is taking place or not.

Golomb et al. [7] introduces CIoTA, which is a blockchain-based solution for collaborative anomaly detection across a large number of IoT devices. While staying resilient to adversarial attacks, CIoTA constantly trains an anomaly detection model. CIoTA can also distinguish between uncommon benign events and malevolent activity by harnessing the knowledge of the crowd. One downside of CIoTA is that each IoT model/firmware must have its own chain published. As a result, CIoTA is best suited to large industrial settings and smart cities in its current state. We intend to develop CIoTA in the future to support a variety of frameworks and increase its detection capability, for example, by investigating API flows rather than lower-level control flows. Ide et al. [8] presented a system (CollabDict) based on blockchain and the Gaussian mixture learning algorithm for collaborative anomaly detection. The major challenges which the author faced here were building the consensus, validating the data, and security of the data. However, the performance of the CollabDict is better than the fuses multitask learning algorithm.

Kumari et al. [9] primarily examine the issue of harmful behaviors occurring in blockchain networks, and then attempt to remedy the problem using the clustering protocol. As a result, the authors keep a check on each node's behavior pattern. Had the authors tried to perform manually for each node, it would have been practically impossible to do so for all the nodes. The K-means clustering approach was utilized to perform the clustering. However, with that algorithm, considerable improvisation was required. As a result, an adapted version of the k-means method was used. This in turn made the blockchain safer against any unlawful or unusual activity. However, the major disadvantage is that the authors used the mean value for each cluster, thus an inaccurate cluster head could be selected.

Dey [10] employs game theory and supervised machine learning techniques to identify anomalous player behavior in a blockchain network. The author provides the probability for each attack based on the value of each transaction; however, the implementation was still in its early phases and hence required a lot of improvements in the defense mechanism. Signorini et al. [11] proposed BAD (blockchain anomaly detection). BAD, in particular, enables the detection of abnormal transactions and the prevention of their propagation.

While forks can occur naturally in the blockchain life cycle owing to network delays, they can also be generated purposefully by attackers and used to commit fraud. Malicious acts are dispersed throughout the chain. By gathering data, BAD enables the avoidance of repeated attacks and builds a tamper-proof threat database that is distributed (thus preventing any single point of failure), trusted (the majority of the network collects and verifies any behavioral data), and private.

Kanth et al. [5] implemented a collaborative intrusion detection system capable of recording login activity via a private blockchain-based ledger and hence it is immutable. In initial stages the authors were successfully capable of proving that blockchain-based CIDSs were a viable method to detect doorknob-rattling attacks and hence can prevent any act of an intruder trying to modify the activity records. The author also uses CPU utilization as a metric to accurately determine whether an intrusion is taking place or not.

Steichen et al. [12] discusses security issues regarding private or consortium blockchains. In this paper, the authors discussed how an attacker can target individual nodes since the number of nodes undertaking blockchain-related tasks is generally restricted. As a result, ChainGuard, which is built as an SDN module and identifies and intercepts excessively large flows at the network level, was proposed in this study. ChainGuard's implementation specifics were discussed and trials were carried out. The tests conducted by the author indicate that ChainGuard can effectively resist DoS and DDoS assaults while allowing a restricted number of packets to cross the SDN network, and hence permits communication between benign blockchain nodes to continue in the case of an attack. Zhu et al. [13] discussed a novel approach to achieving the controllable blockchain CBDM, which is used to obtain storage efficiency in the cloud computing network and to reduce the risk of attacks which are malicious in the blockchain. Though not tested in a real environment, it provides huge scope for development of the prototype.

Hu et al. [14] discussed the multi-microgrid system which it creates a collaborative intrusion detection (CID) paradigm based on blockchain technology. It stores the CID goal in a blockchain and uses a consensus mechanism to create a multi-microgrid system correlation model. It also reduces the false-negative rate and considerably improves the DPoS consensus algorithm by continuously using multiple patterns. However, the major drawback here is that this method does not provide a higher level of true-positive rates and is also limited to fewer types of attacks. N. Alexopoulos et al. [15] uses blockchain technology to improve CIDSs and also provides a combined architecture based on the CIDS and blockchain. This paper proposes a model which considerably reduces the overhead and volume of the blockchain. However, the author has provided a prototype, or a higher view, and the model was not tested in the real environment.

Li et al. [16] focused on signature-based collection in their study and proposed a CBSigIDS, a general framework for a collaborative blockchained, signature-based IDS that used blockchains to help gradually share and construct a trusted signature database, inspired by previous blockchain applications. It improved the effectiveness of signature-based IDSs. However, a major drawback is that it was prone to advanced attacks, and the need for verification and updates in the blockchain resulted in the diminishing performance of the overall network.

In recent years, many other IDSs have been proposed [17–21]. These IDSs can be used in any domain to identify intrusions or abnormalities, which can then lead to the development of a secure solution for smart cities. In smart cities, everything is connected to the internet so a smart IDS can play a significant role to provide the security for this created network. Aloqaily et al. [22] proposed an IDS for securing transportation. This IDS will help in vehicular service management to secure the network from attacks and ensure the quality-of-service availability. Elrawy et al. [23] discussed the role of IDSs and the IoT in the smart environment. This article first discusses various existing works that have contributed to the smart environment using IoT sensors, and then discusses the existing IDSs used to provide security in an IoT context. Elsaeidy et al. [24] introduced a smart IDS to prevent distributed denial of service (DDoS) attacks in smart cities. This article used

the restricted Boltzmann machines (RBMs) technique to design the IDS. Saba et al. [25] proposed an ensemble-based IDS for smart city hospitals.

The current IDS is not sophisticated enough to detect distributed, parallel attacks that take place throughout the nodes in the network instead of a single node. In the case of CIDSs, the ability to correlate events is crucial. The events occurring across all nodes in the network must be aggregated for further processing and raising of alerts. The concept of trust is crucial among the nodes. While the discussed approaches have their advantages, there was clearly a lack of scalable architecture in the case of the CIDS. The main aim of this paper is to demonstrate an approach through which a scalable architecture could be developed, and trust could be established among the nodes in the CIDS architecture. Blockchain was proposed to solve the trust issue in the case of the CIDS.

4. Proposed Methodology and Implementation

In the case of the doorknob-rattling scenario, there is a clear need for a CIDS as demonstrated by Alexopolous et al. [15]. The doorknob-rattling scenario can be further explained. In this case, suppose 50 stand-alone nodes in the network are using IDSs and tracking login attempts. The threshold for an individual machine could be set to 4. Instead of making 4 incorrect attempts on each node, the attacker would make a series of 2 incorrect attempts until he successfully logs on to a particular node. To utilize this, the attacker uses the common list of user-ids and passwords available. If the system is using an IDS, these activities would go unnoticed. However, in the case of a CIDS, these activities would clearly be noticed, and a clear spike would be seen as in Figure 3.

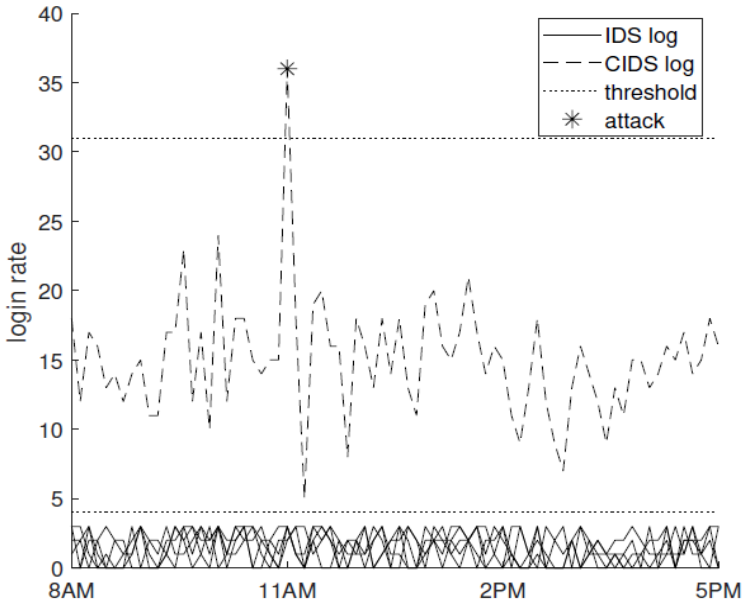

Figure 3. Doorknob-rattling attack [5].

Table 2 indicates the parameters necessary for the building of a CIDS system. While these are necessary, some of them are complementary to each other. That is, while satisfying one of the requirements, there would be a high chance of violating the other. For example, accountability means disclosing some of the information about the node in the system while that would clearly defy the rules of privacy. Hence, there are clear trade-offs between one and another.

Table 2. Requirements of a CIDS system [26].

Accountability	Nodes must be responsible for the actions taken by them.
Integrity	Data cannot be manipulated once entered into the system.
Resilience	The system should be free from SPoF.
Consensus	Nodes in the system must trust the data sent by other nodes.
Scalability	The system must be able to scale as the number of nodes increases.
Overhead	The overhead cost must be minimized to achieve scalability.
Privacy	Privacy must be a concern for the participants in the system.

The main challenge here was to develop a CIDS framework which would decrease the overhead costs and would be scalable in cases where the size of the network increases. Blockchain was used to implement the proof-of-concept described earlier. A private Ethereum-based ledger was used in our case. In order to log the successful attempts made, pluggable authentication modules were used. The pluggable authentication modules were used in order to securely log the attempts made in the system so that these attempts could be transferred to the blockchain as environment variables. In order to use this, the pam_exec.so was used to run a shell script login_success.sh and the pam_exec file passed the login information to the shell script to the shell script as environment variables. These variables were then sent to the log files which were stored automatically. In order to send data to blockchain, cron utility software was used in Linux which scheduled the transfer of data from the log files directly to the blockchain by running a python script at a continuous interval of 5 minutes. If the login was successful, data from the logs would immediately be sent to the blockchain. This is because a scenario was imagined where the attacker would gain access to these log files and could tamper or remove them in order to remove the proof of his presence in the node. In order to simulate an attack, continuous login attempts were made from different machines using SSH (secure shell) to check whether the attempts were being logged or not. To make continuous attacks from another host, cronjobs were again used to call shell script files at an interval of 5 min. This shell script would make a series of wrong attempts trying to log in as different users in the target machine. This is undertaken to reduce the overhead cost incurred every time a transaction is made on the blockchain.

In order to make the system more secure, another parameter was used. This was measuring the CPU utilization of the target machine. There may be a case where an attacker, after gaining access to the system, would try to run malicious programs. In order to detect this, CPU utilization was also logged and stored in log files. To measure CPU utilization, the command used was top | head -3 | tail -1. This command would give the CPU utilization at that given instant. Our system was designed in such a way that if the CPU utilization would exceed a given threshold, this would be logged on the log files. After being logged, these files would be sent to the blockchain at an interval of every 5 min. The spike in the gas cost which can be seen through the Ganache UI would warn the system administrator of the possible attacks which might be taking place in the node.

5. Results Discussion

5.1. CPU Utilization

A simple experiment was performed to confirm that our CIDS setup would be able to precisely record CPU utilization information. The system would record CPU utilization every minute and record the result. If the CPU utilization exceeded a particular threshold, for example, 50%, it would get recorded in a log file named cpu.log. To spike the CPU utilization, another program (prime) was running in the background. The purpose of prime was to spike the CPU utilization, utilizing 10 threads so that the results could be logged onto the cpu.log file. Based on the usage of the current system, a threshold was set to 50% to trigger the sending of data to the log files. This threshold could be modified according to

the usage of the system and based on user needs. Figure 4 illustrates the cpu.log recording all CPU utilization.

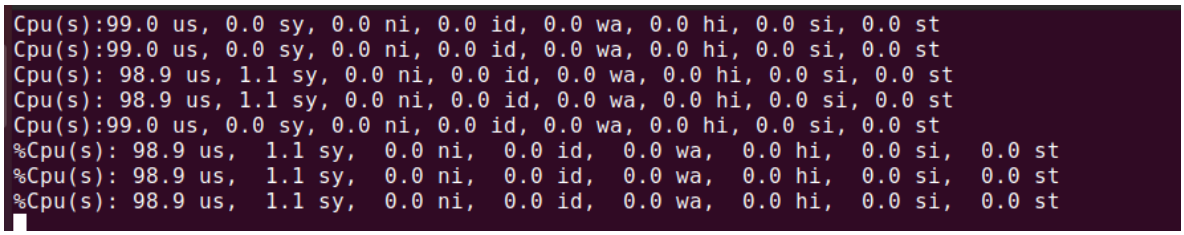

Figure 4. Snapshot from cpu.log recording all CPU utilization.

5.2. Login Attempts

After the CIDS was set up, the main objective was to capture different authentication requests, including the login attempts. Figure 5 shows the output of the log file when a person tries to become a super-user. This is determined by the $PAM_RUSER field because both sudo and su can change the context of the user.

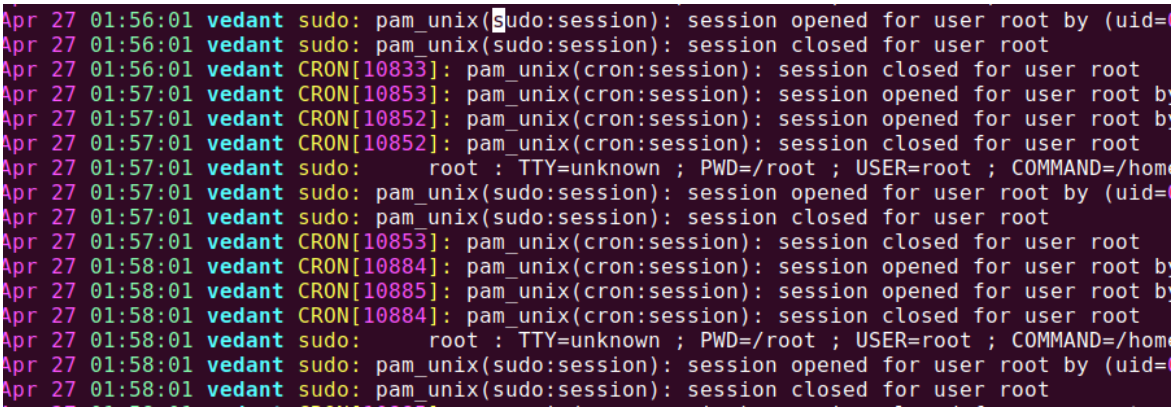

Figure 5. Output of auth.log when someone tries to become a super-user.

Table 3 illustrates the results when the user tries to become a super-user using the sudo and su commands.

Table 3. The attempts when the user tries to become a super-user using the *sudo* and *su* commands.

$PAM_USER	$PAM_TYPE	$PAM_SERVICE	$PAM_RUSER	Date	Success/Failure
vedant	auth	sudo	vedant	Tue 27 May 01:56:00	Success
vedant	auth	su	vedant	Tue 27 May 01:59:00	Success

Table 4 shows the use of an external host by making use of the ssh command to log onto the CIDS remotely. This example makes use of the $PAM_RHOST field, containing information about requesting hosts. In the current example, an external agent (192.168.87.4) successfully logged into the CIDS node as '*vedant*' via the *ssh* vedant@192.168.87.3 command. This is a crucial use case as the doorknob-rattling attack typically involves remote users attempting to penetrate the target network [27].

Table 4. Login attempt evidence.

$PAM_USER	$PAM_TYPE	$PAM_SERVICE	$PAM_RHOST	Date	Success/Failure
vedant	auth	sshd	192.168.87.3	Wed 27 May 01:56:23	Success

Using the current method to record the external login attempts evidenced by Table 4, a simulated doorknob-rattling attack was tried against one of the machines in the network. During the test, the intruder tried using different user accounts on a single machine. The output from the log files is shown in Figure 6. The attacking machine tried to penetrate the machine thrice using a secure shell (SSH) into each of the user accounts. Each of these attempts was recorded and sent to the blockchain as transactions [28–30].

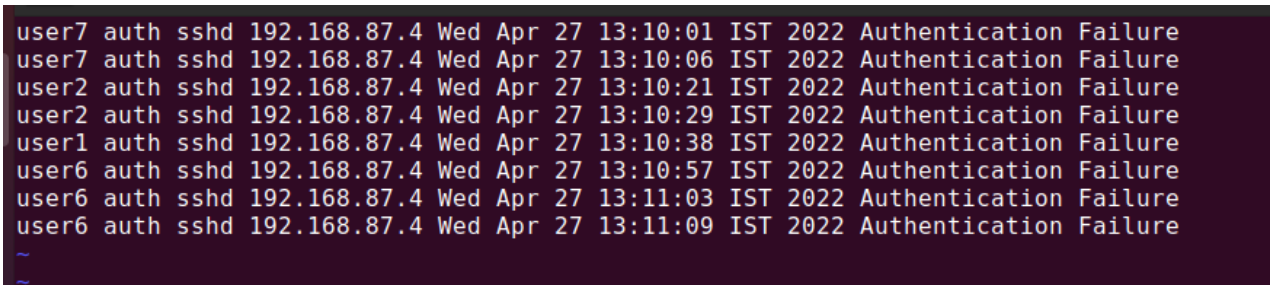

Figure 6. Doorknob-rattling attack in a ledger.

Each transaction had a varying gas cost based on the number of attempts the intruder made to penetrate the machine. All records from the log files were pushed onto the blockchain either at the end of a specified time interval (in the case where all the attempts during the interval were failed login attempts) or were sent immediately to the blockchain if there were any successful logins [31].

A brief summary of the doorknob-rattling attack events in the case of a single attacker is shown in Table 5. The transaction which took place in Ganache can be seen in Figure 7.

Table 5. Summary of doorknob-rattling attack.

User	IP Address of Request	Number of Login Attempts
user1	192.168.87.3	1
user2	192.168.87.3	2
user6	192.168.87.3	3
user7	192.168.87.3	2

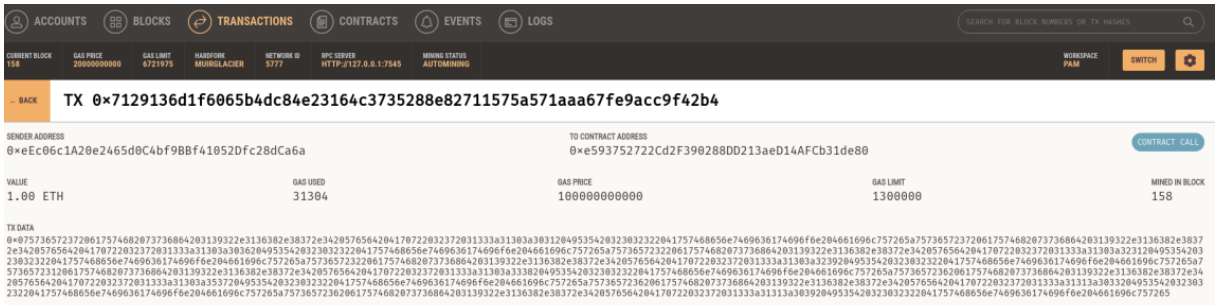

Figure 7. Transactional data which were sent to the blockchain.

There was a total of eight login attempts over four different user accounts. Figure 7 shows the transaction which was submitted to the blockchain and the transaction hash and all the other details.

Figure 8 shows a list of all the blocks which were mined during the entire process. It also shows the gas cost incurred during the entire attack.

BLOCK	MINED ON	GAS USED
158	2022-04-27 13:15:01	31304
157	2022-04-27 13:10:01	28728
156	2022-04-27 13:05:01	45528
155	2022-04-27 13:04:50	23576
154	2022-04-27 13:00:02	50616
153	2022-04-27 11:00:02	22248
152	2022-04-27 10:59:25	21000

Figure 8. List of all the blocks mined and the respective gas costs.

The timestamps of these transactions show that the attacks were permanently recorded in the CIDS distributed ledger. The given sequence of events and protection of related data shows that the nodes can be protected, and the system administrator would be made aware of the possible intrusion immediately. This also proves that Ethereum and Ganache work smoothly with Linux and the integration between them to achieve data ingest for intrusion detection is successful.

5.3. Detecting an Anomaly: Thwarting a Doorknob-Rattling Attack

The main aim of our CIDS architecture was to record data that could be used to detect anomalies. This is because data collection at the end is not enough. It needs to be processed and the analysis should be performed to find potential threats. Subroutines were created which were scheduled to run at specific time intervals using the cron utility software. This made the traffic steady over a period of time. The cron software utility in Linux was used to schedule the bash script anomaly.sh. The anomaly.sh is used to ensure that 20 to 30 login attempts were made from random users on the virtual machine at an interval of every five minutes. This was continued for several iterations. All these attacks were logged into our target machine and data were being sent to the blockchain. The main idea behind this approach was that an increase in the number of transactions would mean higher gas cost. The gas cost for each transaction is further used to analyze whether an attack has actually taken place or not. Table 6 shows the transactions which took place during the attack and the total gas cost incurred. There were three instances where the number of attacks crossed the threshold. The time instances were 10:40, 11:20, and 11:45.

The transactions were then plotted onto a graph with the number of transactions on the primary axis and the number of attempts on the secondary axis. The peaks show that the number of transactions crossed the threshold as shown in Figure 9. Figure 10 shows the bar chart of gas costs in various time frames. Although the inability to detect all the attacks was problematic, this statistical method did correctly identify that there was an anomaly, which would lead a system administrator to further investigate.

Table 6. Transactions and their respective gas costs in an interval of five minutes.

Time	Number of Transactions	Total Gas Cost
10:20	1	22,280
10:25	1	22,280
10:40	18	44,184
10:45	1	22,280
10:50	4	26,152
11:00	4	23,576
11:15	2	23,576
11:20	8	26,152
11:25	1	22,280
11:30	1	22,280
11:35	26	54,488
11:45	1	22,280
11:50	4	26,152
11:55	1	22,280

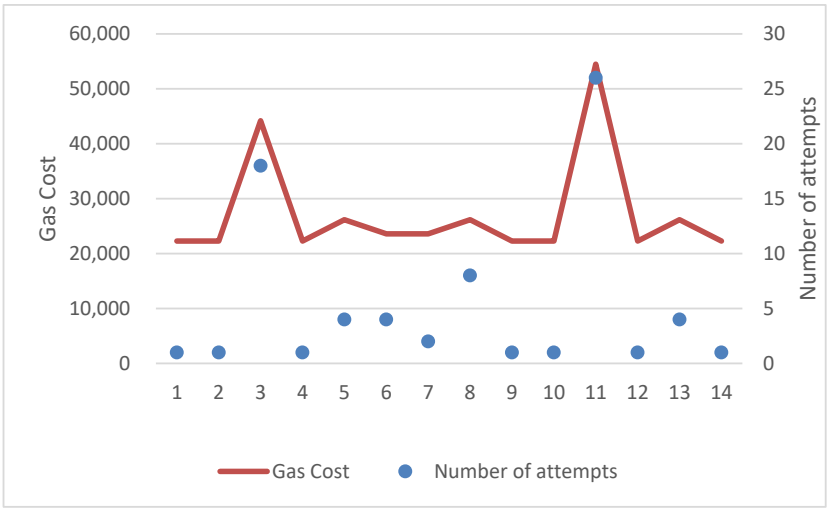

Figure 9. Graph showing number of attempts and corresponding gas costs at an interval of five minutes.

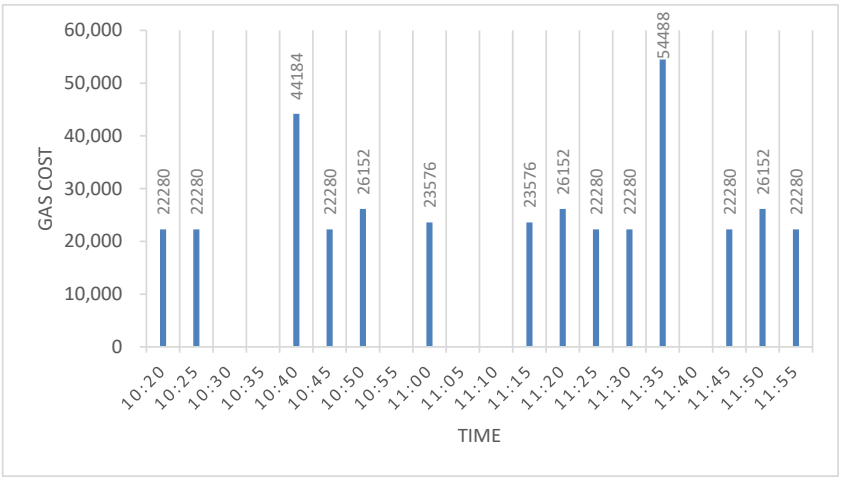

Figure 10. Bar chart of gas costs in various time frames.

6. Conclusions and Future Work

The sharing of information is extremely crucial between the nodes in a CIDS system in order to prevent the system from attacks as a whole. Information sharing is extremely important in a scenario where distributed attacks are taking place increasingly. CIDSs, along with blockchain, appears to be highly suitable for the ingesting of data, especially in the case of building a smart sustainable city. This paper showed that commercial and open-source blockchain technologies may be used to create an information-sharing system that records both doorknob-rattling attacks using pluggable authentication modules and CPU utilization data as blockchain transactions. This also proves that a blockchain system can also be used as a logging mechanism for multiple machines and hence can be used to aggregate data which could be later processed for intrusion detection. This research provides positive indications that blockchain technology could be used on a large scale for solving the intrusion detection problem and building a CIDS at a very large scale.

The most significant contribution made in this paper is that it provides an end-to-end proof-of-concept for CIDS. It also showed at an initial level that attacks or intrusions can be detected using blockchain as a backbone of the CIDS framework. However, there is a need to consider the cost of setting up such a system and how sound it is. The proof-of-concept, which was discussed in the literature, was not implemented at an end-to-end level.

The main aim of this paper was to build an IDS which could be potentially used to detect system abnormalities and intrusions. There are several avenues which are left to explore in this paper for additional work. The main aim going further would be to create a large-scale system which could detect anomalies, block them, and trigger alerts to the system administrator. Further research is also required to see how the overhead cost of running the blockchain client would be taken care of. Currently, Ganache (a private blockchain running at a particular node) is used for testing and carrying out transactions in the blockchain. Public or other test nets could be used to carry out system tests.

Author Contributions: Methodology, V.C.; Formal analysis, S.M.; Investigation, R.K.P.; Resources, R.K.G.; Writing—original draft, S.B.H.S.; Writing—review & editing, P.K.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data that support the findings of this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jose, S. A Survey on Anomaly Based Host Intrusion Detection System. *J. Phys. Conf. Ser.* **2018**, *1000*, 1–11. [CrossRef]
2. Li, W. Surveying Trust-Based Collaborative Intrusion Detection: State-of-the-Art, Challenges and Future Directions. *IEEE Commun. Surveys Tuts* **2021**, 280–305. [CrossRef]
3. "What Is Hashing? Step-by-Step Guide-Under Hood of Blockchain. August. 2017. Available online: https://blockgeeks.com/guides/what-is-hashing/ (accessed on 2 July 2022).
4. Salam, A.-E.; Mohammed, A.; Yousef, S.; Selvakumar, M.; Iznan, H. Intrusion Detection Systems Using Blockchain Technology: A Review, Issues and Challenges. *Comput. Syst. Sci. Eng.* **2021**, *40*, 87–112. [CrossRef]
5. Kanth, V.; Mcabee, A.; Tummala, M.; Mceachen, J. Collaborative Intrusion Detection leveraging Blockchain and Pluggable Authentication Modules. In Proceedings of the 53rd Hawaii International Conference on System Sciences, Maui, HI, USA, 7–10 January 2020.
6. Dreger, H.; Feldmann, A.; Paxson, V.; Sommer, R. Predicting the Resource Consumption of Network Intrusion Detection Systems. International Workshop on Recent Advances in Intrusion Detection. 2008. Available online: https://link.springer.com/chapter/10.1007/978-3-540-87403-4_8 (accessed on 2 July 2022).
7. Golomb, T.; Mirsky, Y.; Elovici, Y. CIoTA: Collaborative IoT anomaly detection via Blockchain. *arXiv* **2018**, arXiv:1803.03807.
8. Idé, T. Collaborative Anomaly Detection on Blockchain from Noisy Sensor Data. In Proceedings of the 2018 IEEE International Conference on Data Mining Workshops (ICDMW), Singapore, 17–20 November 2018; pp. 120–127.
9. Kumari, R.; Catherine, M. Anomaly detection in Blockchain using clustering protocol. *Int. J. Pure Appl. Math.* **2018**, *118*, 391–396.
10. Dey, S. Securing majority-attack in blockchain using machine learning and algorithmic game theory: A proof of work. In Proceedings of the 2018 10th Computer Science and Electronic Engineering (CEEC), Colchester, UK, 19–21 September 2018; pp. 7–10.

11. Signorini, M.; Pontecorvi, M.; Kanoun, W.; Di-Pietro, R. ADvISE: Anomaly Detection tool for Blockchain Systems. In Proceedings of the 2018 IEEE World Congress on Services (SERVICES), San Francisco, CA, USA, 2–7 July 2018; pp. 65–66.
12. Steichen, M.; Homme, S.; State, R. ChainGuard—A firewall for blockchain applications using SDN with OpenFlow. In Proceedings of the 2017 Principles, Systems and Applications of IP Telecommunications (IPTComm), Chicago, IL, USA, 25–28 September 2017; pp. 1–8.
13. Zhu, L.; Wu, Y.; Gai, K.; Choo, K.R. Controllable and trustworthy blockchain-based cloud data management. *Future Gener. Comput. Syst.* **2019**, *91*, 527–535. [CrossRef]
14. Hu, B.; Zhou, C.; Tian, Y.C.; Qin, Y.; Junping, X. A collaborative intrusion detection approach using Blockchain for multimicrogrid systems. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *49*, 1–11. [CrossRef]
15. Alexopoulos, N.; Vasilomanolakis, E.; Ivánkó, N.R.; Mühlhäuser, M. Towards Blockchain-based collaborative intrusion detection systems. In Proceedings of the Critical Information Infrastructures Security 12th International Conference, CRITIS 2017, Lucca, Italy, 8–13 October 2017; D'Agostino, G., Scala, A., Eds.; Springer: Cham, Switzerland, 2018.
16. Vasilomanolakis, E.; Karuppayah, S.; Mühlhäuser, M.; Fischer, M. Taxonomy and survey of collaborative intrusion detection. *ACM Comput. Surv.* **2015**, *47*, 1–33. [CrossRef]
17. Liang, C.; Shanmugam, B.; Azam, S.; Karim, A.; Islam, A.; Zamani, M.; Kavianpour, S.; Idris, N.B. Intrusion Detection System for the Internet of Things Based on Blockchain and Multi-Agent Systems. *Electronics* **2020**, *9*, 1120. [CrossRef]
18. Ghaleb, F.; Saeed, F.; Al-Sarem, M.; Ali Saleh Al-rimy, B.; Boulila, W.; Eljialy, A.E.M.; Aloufi, K.; Alazab, M. Misbehavior-Aware On-Demand Collaborative Intrusion Detection System Using Distributed Ensemble Learning for VANET. *Electronics* **2020**, *9*, 1411. [CrossRef]
19. Radoglou-Grammatikis, P.I.; Sarigiannidis, P.G.; Efstathopoulos, G.; Panaousis, E.A. A Novel Multivariate Intrusion Detection System for Smart Grid. *Sensors* **2020**, *20*, 5305. [CrossRef] [PubMed]
20. Iwendi, C.; Anajemba, J.H.; Biamba, C.; Ngabo, D. Security of Things Intrusion Detection System for Smart Healthcare. *Electronics* **2021**, *10*, 1375. [CrossRef]
21. Kotecha, K.; Verma, R.; Rao, P.V.; Prasad, P.; Mishra, V.K.; Badal, T.; Jain, D.; Garg, D.; Sharma, S. Enhanced Network Intrusion Detection System. *Sensors* **2021**, *21*, 7835. [CrossRef] [PubMed]
22. Aloqaily, M.; Otoum, S.; Al Ridhawi, I.; Jararweh, Y. An intrusion detection system for connected vehicles in smart cities. *Ad Hoc Netw.* **2019**, *90*, 101842. [CrossRef]
23. Elrawy, M.F.; Awad, A.I.; Hamed, H.F. Intrusion detection systems for IoT-based smart environments: A survey. *J. Cloud Comput.* **2018**, *7*, 1–20. [CrossRef]
24. Elsaeidy, A.; Munasinghe, K.S.; Sharma, D.; Jamalipour, A. Intrusion detection in smart cities using Restricted Boltzmann Machines. *J. Netw. Comput. Appl.* **2019**, *135*, 76–83. [CrossRef]
25. Saba, T. Intrusion Detection in Smart City Hospitals using Ensemble Classifiers. In Proceedings of the 13th International Conference on Developments in eSystems Engineering (DeSE), Liverpool, UK, 14–17 December 2020. [CrossRef]
26. Zhu, B.; Joseph, A.; Sastry, S. A taxonomy of cyber attacks on scada systems. In Proceedings of the 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing, Washington, DC, USA, 19–22 October 2011; pp. 380–388.
27. Debar, H.; Dacier, M.; Wespi, A. Towards a taxonomy of intrusion-detection systems. *Comput. Netw.* **1999**, *31*, 805–822. [CrossRef]
28. Proffitt, T. How Can You Build and Leverage SNORT IDS Metrics to Reduce Risk? *SANS Institute.* 2013. Available online: https://www.sans.org/reading-room/whitepapers/tools/paper/34350 (accessed on 2 July 2022).
29. Hu, J. *Host-Based Anomaly Intrusion Detection*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 235–255. [CrossRef]
30. Khan, A.R.; Kashif, M.; Jhaveri, R.H.; Raut, R.; Saba, T.; Bahaj, S.A. Deep Learning for Intrusion Detection and Security of Internet of Things (IoT): Current Analysis, Challenges, and Possible Solutions. *Secur. Commun. Netw.* **2022**, *2022*, 1–13. [CrossRef]
31. Parwani, D.; Dutta, A.; Shukla, P.K.; Tahiliyani, M. Various Techniques of DDoS Attacks Detection & Prevention at Cloud: A Survey. *J. Comput. Sci. Technol.* **2015**, *8*, 110–120.