Blockchain-Based Architecture for Secured Cyber-Attack Features Exchange

Oluwaseyi Ajayi and Tarek Saadawi

Department of Electrical Engineering, City College of New York, New York, NY 10031

Oajayi000@citymail.cuny.edu *saadawi@ccny.cuny.edu*

***Abstract*— Despite the increased accuracy of intrusion detection systems (IDS) in identifying cyberattacks in computer networks and devices connected to the internet, distributed or coordinated attacks can still go undetected or not detected on time. The single vantage point limits the ability of these IDSs to detect such attacks. Due to this reason, there is a need for attack characteristics' exchange among different IDS nodes. Researchers proposed a cooperative intrusion detection system to share these attack characteristics effectively. This approach was useful; however, the security of the shared data cannot be guaranteed. More specifically, maintaining the integrity and consistency of shared data becomes a significant concern. In this paper, we propose a blockchain-based solution that ensures the integrity and consistency of attack characteristics shared in a cooperative intrusion detection system. The proposed architecture achieves this by detecting and preventing fake features injection and compromised IDS nodes. It also facilitates scalable attack features exchange among IDS nodes, ensures heterogeneous IDS nodes participation, and it is robust to public IDS nodes joining and leaving the network. We evaluate the security analysis and latency. The result shows that the proposed approach detects and prevents compromised IDS nodes, malicious features injection, manipulation, or deletion, and it is also scalable with low latency.**

Keywords* — *Blockchain, Cyberattack, Compromised nodes, Features, Intrusion Detection System, Salability, Latency, Security.

I. Introduction

The increase in the use of the internet has made data storage and exchange easily achievable. However, the vulnerabilities of these data to cyberattacks increase tremendously. The authors in [1] proposed firewall, data encryption, and user authentication for keeping the unauthorized user from assessing stored data, but malicious intruders still find ways to subvert these protection systems and gain access to the unauthorized data. Further researches put forward intrusion detection systems (IDS) to identify malicious intruders in computer networks and devices connected to the internet [2,3]. These intrusion detection systems can either be classified based on their locations in the network: Host-based detection system (HIDS) and network-based detection system (NIDS) [4] or their detection approaches: signature-based and anomaly-based [3].

Intrusion detection systems have proven to be useful in identifying malicious activities; however, their single viewpoints limit the ability to detect distributed or coordinated cyberattacks. The vantage point has made it possible for some attacks to go undetected or not detected on time. Due to the escape of some attacks, there is a need for IDSs to exchange attack features with the view of detecting new attacks promptly. Apart from this, a zero-day attack (an attack without a known signature) experienced in an organization's IDS located, say in London, the United Kingdom might be different from that experienced in another organization's IDS located, say Washington DC, United States or another company located in the same region. Therefore, if IDS nodes exchange this threat information, more malicious activities can be stopped by coordinating efforts of participating IDS. A cooperative intrusion detection system was proposed to improve the detecting power of single IDS [5-7]. In a cooperative intrusion detection system, IDS nodes exchange attack features with the view of promptly detecting an attack that has previously been detected by other IDS nodes. Users adopted cooperative intrusion detection system due to its better performance; however, some of the major problems threatening the approach are: (i) data manipulation during exchange, (ii) fake data injection to the database, (iii) data deletion if no one monitors the database activities and (iv) inability to guarantee the consistency of the shared data.

The existing cooperative intrusion detection is divided into four main stages [8]. The main vulnerable stages are data storage and distribution stages (Fig. 1). Most of the existing approaches to secure stored data either engage a centralized approach (which makes the network vulnerable to single-point-of-failure and man-in-the-middle attacks [9, 10,11]) or uses a decentralized approach in which the integrity and consistency of the shared data cannot be guaranteed [12,13].

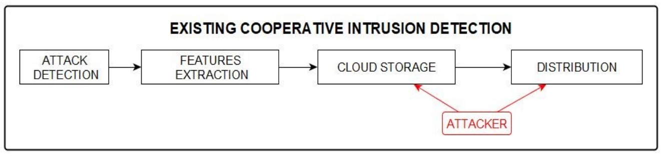

Fig. 1. Cyber-attack targets of existing cooperative intrusion detection

We propose an approach that leverages distributive ledger technology, data immutability, and tamper-proof abilities of blockchain technology to detect and block malicious activities. The proposed approach extracts cyberattack features, stores, and securely distribute among participating nodes in real-time (Fig. 2). We define attack features as characteristics of attacks, retrieve from attacks traffic detected by any IDS.

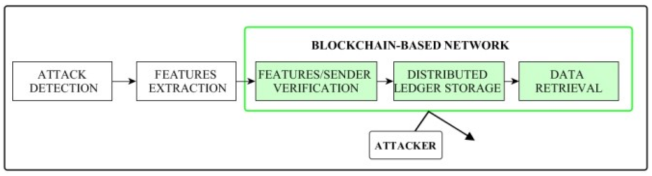

Fig. 2. The proposed blockchain-based solution.

The contributions of our work can be summarized as follows:

- We propose a private-public blockchain-based architecture that facilitates scalable and secured attack features exchange among IDS nodes in computer networks
- The architecture detects and prevents malicious activities on the stored data from both outsider and insider threats.
- The architecture verifies the integrity and consistency of the retrieved features and present in a standard format which encourages heterogeneous IDS nodes participation.
- The architecture permanently stores the verified attack features and shares among IDS nodes using a blockchain network.
- The proposed architecture is robust to public IDS nodes joining and leaving the network in real-time.

The remainder of this paper is organized as follows: Section II discusses the background and related works on cooperative intrusion detection and blockchain technology. Section III describes the proposed architecture. Section IV presents the results. While section V presents the conclusions of this paper and possible future works.

II. BACKGROUND AND RELATED WORKS

First introduced as the technology behind bitcoin in 2008 [14], blockchain was implemented to solve the double-spending problem in a cryptocurrency called bitcoin. Since its inception, diverse areas have seen the application of blockchain technology. e.g. health system [15,16], data integrity security [17], as an intrusion detection system [18 - 20]. Blockchain is an append-only public ledger that records all transactions that have occurred in the network. Every participant in a blockchain network is called nodes. The data in a blockchain is known as a transaction, and it is divided into blocks. Each block is dependent on the previous one (parent block). Every block stores some metadata and hash value of the previous block. So, every block has a pointer to its parent block. Each transaction in the public ledger is verified by the consensus of most of the participants in the system. Once the transaction is verified, it is impossible to mutate/erase the records [14]. Blockchain is broadly divided into two: public and private blockchain[21]. A public blockchain is a permissionless blockchain in which all nodes do verification and validation of transactions. e.g., Bitcoin, Ethereum. While private blockchains are permissioned blockchains where only nodes given permission can join and participate in the network. e.g., Hyperledger.

1. *Blockchain application*

The authors in [18], [19] and [20] proposed the use of blockchain technology in detecting an anomaly. In [18], the authors proposed a blockchain anomaly detection solution (BAD) that focuses on detecting attacks directed at the blockchain network. BAD prevents the insertion of a malicious transaction from spreading further in the blockchain. BAD leverages blockchain metadata named forks to collect potentially malicious activities in the blockchain network. Their works used machine learning to train blockchain nodes to detect malicious activities. In their approach, they considered eclipse attack (an attacker infects node's list of IP addresses, thus forcing the victim's node list of IP addresses to be controlled by that attacker). The analysis of the result showed that BAD was able to detect and stop the spread of attack that uses bitcoin forks to spread malicious codes. However, the solution is specific to attacks directed towards the blockchain network and use bitcoin forks. In another research put forward in [19], the authors proposed collaborative IoT anomaly detection via blockchain solution (CIoTA). CIoTA uses the blockchain concept to perform distributed and collaborative anomaly detection on IoT devices. They used CIoTA to continuously trained anomaly detection models separately and then combine their wisdom to differentiate between rare benign events and malicious activities. The evaluation of the result showed that combined models could detect malware activities easily with zero false positives. The proposed solution uses a collaborative effort of IoT to detect attacks; hence, it does not solve the security concerns facing cooperative intrusion detection, and it is specific to malware attacks.

The authors in [20] proposed a blockchain-based malware detection solution in mobile devices. In their work, they extracted installation package, permission package, and call graph package features for all known malware families for android based mobile devices and uses it to build a feature database. Their result showed that their solution could detect and classify known malware. It also performs malice determination and malware family classification on unknown software with higher accuracy and lower time cost. The solution above is specific to host-based malware attacks on Android-based mobile devices. Hence, it will be difficult to extend it to network-based attacks.

2. *Cooperative intrusion detection*

The authors in [22] proposed a prototype Distributed Intrusion Detection System (DIDS). Their system combines distributed monitoring and data reduction with centralized analysis to monitor a heterogeneous network of computers. The result showed that their prototype demonstrated the viability of distributed architecture in solving the network-user identification problem. However, with the DIDS director responsible for all evaluation, the system is susceptible to single-point-of-failure or man-in-the-middle attacks. Another research put forward in [23] proposed DOMINO (Distributed Overlay for Monitoring Internet Outbreaks). DOMINO is an architecture for a distributed intrusion detection system that fosters collaboration among heterogeneous nodes organized as an overlay network. In their system, they used active-sink nodes that respond to and measure connections to unused IP addresses. This active-sink node enables efficient detection of attacks from spoofed IP sources, reduces false positives, enables attack classification and production of timely blacklists. The result demonstrated the utility of sharing information between multiple nodes in a cooperative infrastructure and active-sink node showed effectiveness in discriminating between types of attacks based on examining payload data. Although their system showed a good result, malicious intruders can hack the database that manages activities, and the integrity of the stored data can be compromised. The authors in [24] proposed a message authentication code (MAC) for detecting any changes in stored data. Although this approach detects any changes in the stored data, however, it is not practical for extensive data because downloading and calculating MAC of large files is overwhelming and time-consuming. Another method described in [24] secures the integrity of cloud data by computing the hash values of every data in the cloud. This solution is lighter than the first approach in [24]; however, it requires more computation power, especially for massive data; hence, it is not practical. The authors in [25] employ the third party to coordinate activities of the database. The problem with this approach is that the data is vulnerable to man-in-the-middle or single-point-of-failure attack.

Despite several kinds of research, the available solutions have not addressed the security problems associated with data exchange in a cooperative intrusion detection system. Hence, the motivation for the work. Our proposed architecture uses blockchain technology to guarantee the security of shared attack features among IDS nodes. The novelty of our architecture is that it facilitates scalable attack features exchange, encourages heterogeneous IDS nodes participation, detects and prevents malicious activities on stored data from both insider and outsider threats. It is robust to public IDS nodes joining and leaving the blockchain network in real-time. This novelty distinguishes our work from previous works.

III. THE PROPOSED ARCHITECTURE

The proposed architecture, which is compatible with any blockchain platform, is built on the Ethereum blockchain platform. Ethereum blockchain is an open-source blockchain-based distributed computing featuring smart contracts. A smart contract is an agreement among consortium members, which is stored on the chain and run by all participants [26]. Although the central Ethereum platform is a public blockchain, we configure it to a combination of public and private networks. Fig. 3 shows a pictorial representation of the proposed architecture.

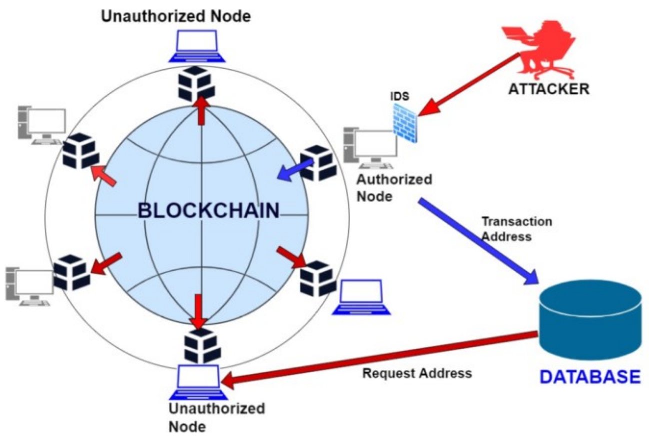

Fig. 3. The Proposed Architecture

The architecture is composed mainly of the following:

- **Authorized Nodes**
  Also known as miners, these nodes prepare, submit, and verify transactions. They also run the consensus algorithm, thus validate transactions/blocks. All authorized nodes update database
- **Unauthorized Nodes**
  These are also known as public nodes. They join the network to retrieve stored attack features. Public nodes are not privileged to prepare, verify, validate, or run consensus algorithm. They do not update the database but can only request the transaction address of the mined blocks.
- **Database**
  Database, which is accessible to all nodes, stores the address of the mined blocks. While all public nodes have read-only access to it, authorized nodes update block information. Any data manipulation in the database results in an inability to access the contents of the blockchain but does not affect data stored in the blockchain network. Such malicious activity can be easily detected.

The proposed architecture is divided into three main stages, as shown below.

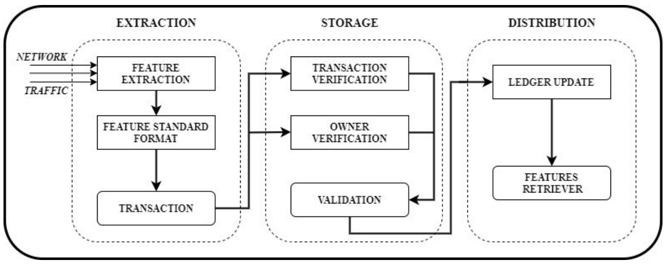

Fig. 4. Building blocks of the proposed architecture

1. *Extraction*

Attack features are characteristics of attack traffic that differentiate them from regular traffic. Anomaly-based IDSs detect malicious incoming traffic patterns based on deviation from typical traffic patterns. The IDSs are trained with the features extracted from regular traffic, then raise alert whenever there is a deviation from the known traffic pattern [3]. In this work, network-based attack features are extracted based on feature names proposed in [27] using network traffic analyzing tools. We extract attack features under two categories: (i) Connection features and (ii) packet features.

i. *Connection features:* These features are obtained from attack network connections. Whenever an attack is detected, a developed script sniffs, captures and analyzes network connections using *tcpdump v. 4.9.2., libpcap v. 1.9.0, tcptrace 6.6.0, and Wireshark v. 3.0.1*. Tcpdump captures and analyzes TCP packets while Wireshark uses libpcap to capture network connections in real-time. Tcptrace is used to analyze the captured attack connections. Some of the features extracted and exchanged from attack connections are shown below in Table I.

Table I: Attack Connection Features

S/N	Feature Name	Definition

1	Source Port	Port from which an attack is launched.
2	Destination Port	Target port in the target network.
3	Source IP	The IP address of the attack node.
4	Destination IP	Target IP address in the target network
5	Source Bytes	The total number of bytes sent from attack nodes during the attack period.
6	Destination Bytes	The total number of bytes sent from the target network to attack nodes during the attack period.
7	Source Packets	The total number of packets sent from attack nodes during the attack period.
8	Connection	The total number of connections initiated with the target network by attack node.
9	Duration	Total time elapsed during an attack.
10	Packets/seconds	The number of packets sent by an attack node within 1 second.
11	Source Host count	The total number of attack nodes connecting to the target network.
12	Destination Host Count	The total number of target nodes in the target network.
13	Throughput	The rate at which attack nodes send bytes to the target node. (measured in kbps).
14	Service Count	The total number of ports connected to attack nodes during the attack period.
15	Same service count	The total number of connections to the same port number during the attack period.
16	Different Host rate	Percentage of attack nodes attacking different target nodes.
17	Same service rate	Percentage of attack nodes attacking the same port during the attack period.
18	Same Host rate	Percentage of attack nodes attacking the same target node during the attack period.

ii. *Packet features*: These attack features are obtained by sniffing and analyzing attack packets. During attack detection, a script that uses *Scapy v 2.4.0* analyzes ingress packets**.** Scapy decodes traffic packets and matches request with replies. Table II shows some of the packet features extracted.

Table II. Attack Packet Features

S/N	Feature Name	Definition
1	Land	'1' if the source and destination IP and ports are the same; otherwise '0'.
2	Type of service	Class of traffic assigned to attack packet
3	Protocol	Higher layer protocol used in the data portion of the attack packet
4	Ip flags	How packet should be routed or processed by higher layer
5	TCP Flags	Defines the type of packet sent by attack node
6	Urgent	Indicates priority of handling packets by the router
7	Time to Live	Time left for a packet to be discarded
8	Checksum	Error checking in the packet header
9	Wrong Fragment	'1' if the checksum is 'incorrect'; otherwise '0.'

Based on the features in Tables I and II, a transaction, which agrees with a standard format, is prepared, signed, and submitted to the blockchain network. To verify the authenticity of the submitted data, the owner submits its verification information. Examples of such verification information are Transaction account, MAC address, IP address.

2. Storage:

The conformity of the submitted transaction with standard format is verified. The architecture also verifies the privilege of transaction owners to submit transactions and the cost of mining such transactions. If these verification steps are successful, the transaction is pushed for validation (i.e., attached to the blockchain). The storage stage is divided into the following steps:

i. *Transaction and owner's verification:* This step ensures that all malicious transactions or activities on the submitted transaction by either insider or outsider threats are blocked. (i.e., it ensures that no public node submits transaction and prevents compromised authorized nodes from participating). The smart contract monitors the cost of mining transactions and keeps track of all nodes that participate in mining a transaction. Algorithm 1 describes the verification process. For verification step to be successful, no feature fields must be missing (i.e., all feature field must have values), verification information must be in their respective sets, transaction owner must not mine its transaction, the cost of mining transaction must not exceed the threshold, and sender's public key must verify the private key. If any of these conditions fail, smart contract returns fail, and the transaction is dropped.

Algorithm 1: Verification

```
Procedure: Verification (Transaction, V.I)
Inputs: Transaction, Verification Information (V.I)

If (Transaction agrees with Standard Format) and
  (Transaction owner does not mine) and
  (V.I in respective V.I sets) and
  (public key verifies private key):

      Return Success
      Push transaction to format creation step
else:
      Return fail
      Drop transaction
end if
end procedure
```

ii. *Validation*: Blockchain consensus protocols handle the validation of transactions. In this work, we combine both Proof-of-Work (PoW) and Proof-of-Stake (PoS). The pending transaction is built into a block and the block is broadcasted into the blockchain network for validation. Every node receives a broadcasted block, but only authorized nodes (miners) work to validate the block. Each block contains a unique code called hash; it also contains a hash of the previous block. Data from previous blocks are encrypted or hashed into a series of numbers and letters.

The authorized nodes work to get the target hash in order to validate a block. A target hash is a number that a hashed block header must be less than or equal to for a new block to be awarded. The miners achieve this target hash by using an iterative process such as POW which requires consensus from all authorized nodes. The characteristics of proof-of-work are computationally difficult to compute and easy to verify. We set an upper bound of stake for every transaction to ensure fair competition among miners (i.e., to discourage authorized nodes with lager stake from always emerge as the miner). The process of guessing the hash starts in the block header. It contains a block version number, a timestamp, the hash used in the previous block, the hash of the Merkle Root, the nonce, and the target hash. Successfully mining a block requires an authorized node to be the first to guess the nonce, which is a random string of numbers and broadcast to other nodes. Other authorized nodes verify the correctness of the nonce value by appending this number to the hashed contents of the block and then rehashed it. If the new hash meets the requirements outlined in the target, then the block is added to the blockchain. The transaction is permanently stored on the blockchain network, and it is impossible to mutate/erase the block.

3. *Distribution*

After a successful validation process, the transaction address is issued to the owner (sender). The blockchain is updated, and the transaction is ready to be retrieved. Steps involved in the secure distribution of mined features are as follows:

i. *Ledger updating:* The newly added block reflects on the ledger which is possessed by every nodes in the network. The transaction address is sent to the database by the transaction owner. This database is opened to the public so that everyone can have access to this information.

ii. *Features Retrieval:* All blockchain nodes receive the notification of the newly added block but do not have access to the content of the block. The transaction address obtained from the database is used to retrieve information stored in the new block. Nodes extract the stored attack features and use them to train their intrusion detection systems.

IV. RESULT

We carry out the implementation of the proposed architecture first in the lab and later extends it to google cloud platform. The aim is to compare the behavior of the architecture when the nodes are in closed promixity to each other with when they are located far apart. We set up eight blockchain nodes, one database node, and one attack node for the lab experiment while we employ seven blockchain nodes located at different regions around United States for the cloud experiment. Table III shows the configuration of the nodes used in the lab, while Fig. 5 shows the location of the nodes when deployed to the cloud. We use Solidity *v 0.6.2* implementation for smart contract and *geth v 1.9.0* for Ethereum. A public node becomes a miner after we write its verification information to the smart contract. Hence, we create our miners by adding the verification information to the smart contract.

Table III: Configuration of Lab nodes

Name	**Machine**	**OS**	**RAM**	**Processor**
Node 1	Desktop	18.04	4GB	2.2GHz
Node 2	Laptop	18.04	16GB	2.81Ghz
Node 3	Desktop	16.04	8GB	2.44GHz
Node 4	Laptop	18.04	4GB	2.44GHz
Node 5	Vmware	18.04	4GB	2.2GHz
Node 6	Vmware	18.04	4GB	2.2GHz
Node 7	Vmware	18.04	4GB	2.2GHz
Node 8	Vmware	18.04	4GB	2.2GHz
attacker	Laptop	16.04	4GB	2.2GHz
database	desktop	window	4GB	I5@2.44

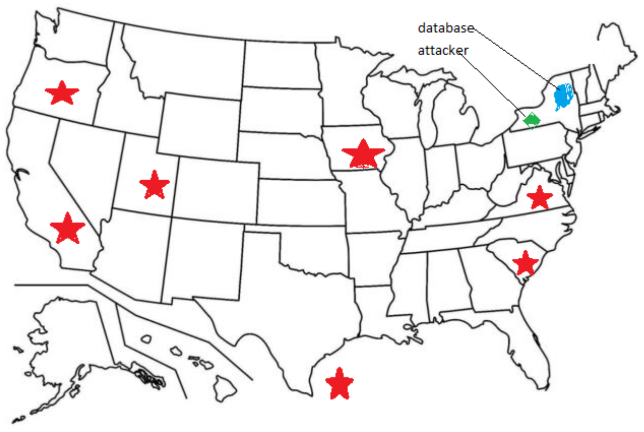

Fig. 5. Location of blockchain nodes around the United States

We install *tcpdump v. 4.9.2., libpcap v. 1.9.0, tcptrace v.6.6.0, Wireshark v. 3.0.1.* and *Scapy v.2.4.0* on all authorized nodes. For the proof of concept, we run connection and packet analyzing scripts and an anomaly-based IDS called Dendritic Cell Algorithm (DCA) [24] on authorized node (node 2). The attacking node launches a Denial of Service (DoS) attack at node 2 and we extracts the features as explained section III. The extracted features are converted to a standard format and submitted to the blockchain network as a transaction. The architecture verifies, validates and distributes these transactions among other nodes (as explained above).

A. *SECURITY ANALYSIS*

1) Outsider Threat Analysis: We analyze the security of the architecture against malicious transaction injection in both experiments. A transaction, prepared by an unauthorized node, is submitted to the blockchain network for verification and validation. Although other authorized nodes work to validate this transaction, we observed that instead of issuing a transaction address to the sender, the owner receives a notification that the transaction has been failed and dropped. The transaction failed because the sender is not privileged to submit the transaction; hence, it fails the verification steps. We investigated further by manually generating the transaction address, then use it to request for the transaction from the blockchain. We observed that the blockchain network did not return any transaction because there is not transaction with such network address.

2) Insider Threat Analysis: Here, we examined the security of the architecture in two common ways an authorized node can be compromised.

a) *A large volume of data*: We implement a case where an authorized node sends a large amount of what appears to be legitimate standard formatted attack features in an attempt to mount a DoS attack on the blockchain network. An authorized node prepare transactions that are massive amount of data and submit to the blockchain network. Although other authorized nodes are working to validate the transaction, we observed that the transactions are not mined because the cost of mining these transactions exceeded the threshold cost. A notification to the owner indicates that the transaction has failed due to its cost. We persistently submit such huge transactions from the same authorized node, and we observed that other miners stop mining after the sender was flagged to be compromised. The smart contract automatically drops all subsequent transactions from the same authorized node.

b) *Fake feature values*: We implement a situation where a compromised authorized node submits what appears to be legitimate standard formatted attack features but with fake data values. The cost of each submitted transaction is within the set range written in the smart contract. It is assumed that an attacker is not likely to hold an authorized node in a compromised state for too long due to frequent security checks by the network administrators. Based on this assumption, an attacker will make all efforts to get its transactions mined as quickly as possible. The result showed that such a transaction is not mined, although other authorized nodes are working to validate the same transaction. The architecture drops the transaction because the owner attempts to mine own transaction, which makes transaction flagged to have been compromised. Based on these results, it shows that the architecture has the capability of verifying the consistency and integrity of submitted transactions, thereby detecting and preventing any malicous activities from both the insider and outsider attackers on the shared data.

B. *PERFORMANCE METRICS*

We obtain the following data for each transaction from every node to analyze the response time.

- *Transaction deployment time (t_1):* This is the time a transaction is submitted to the network. These data are collected directly from the sender console.
- *Execution time (t_3)*: This is the time taken for the content of each transaction to appears in designated files of each node. The time is retrieved by setting on current time on all node consoles.

1) Response Time: This is also known as latency (measured in seconds) of the blockchain network. For each transaction, latency is the difference between the execution time and the deployment time (t_3-t_1). Latency includes verification time, mining time, and time taken for nodes to request transaction address and retrieve mined features. We measure the response time of the architecture in two different scenarios: (i)Closed proximity, i.e., when the nodes are closed to each other (e.g., lab) and (ii) Wide geographical area, i.e., when the nodes are far apart from each other. Fig. 6 shows the average response time of all nodes for both closed proximity and wide-area deployment. The average response time is the addition of all response times for each transaction divided by the number of transactions. We could observe that the difference in the average response time is small for both cases. The slight increase in the response time (Fig. 6A) is due to the computing power of the nodes, which is lower than the cloud nodes employed in Fig. 6B. The result shows that the architecture can facilitate scalable attack features exchange among IDS nodes in computer networks irrespective of the location.

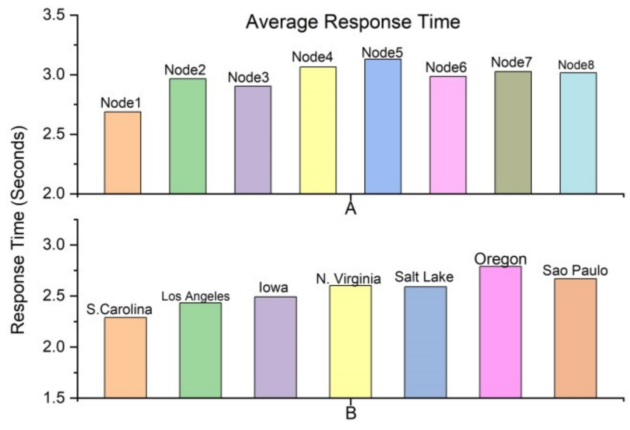

Fig. 6. A)Average response time when performed in the lab. B)Average response time when deployed to locations around the US.

2) *Scalability:* We evaluate the change in the response time of the blockchain network with an increasing number of nodes. We first implement the effect of increasing the number of unauthorized (public) nodes on the response time of two authorized nodes located in South Carolina and Los Angeles. The blockchain network is step up as described above with two authorized nodes and an attack node (located in New York). The features extracted from DoS attacks are prepared as transactions and submitted to the blockchain network. These transactions are verified, validated, and stored on the blockchain network. We record the response time of the two authorized nodes. We increase the number of public nodes joining the network one at a time, repeat the experiment, and record the response time for the two authorized nodes. Fig. 7 shows the response time of the two authorized nodes for an increasing number of public nodes. We observed that increasing the number of public nodes has no effect on the architecture's response time, which implies that the solution is robust to public IDS nodes joining and leaving the network.

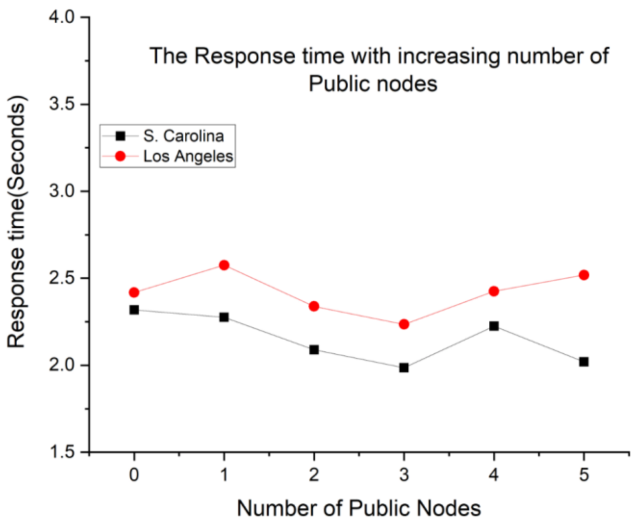

Fig.7. The response time with an increasing number of public nodes

Furthermore, we evaluate the response time with an increasing number of authorized nodes (i.e., miners). We set up a blockchain network, as described above, with seven nodes. Transactions are prepared and submitted to the blockchain network by an authorized node. The submitted transactions are verified, validated, stored, and distributed to all nodes in the network. We repeat the experiment several times, and the average response time of each node is recorded. We increase the number of miners One at a time, repeat the experiment, and record the average response times of each node. Figs. 8, and 9 show the response time of all nodes when the architecture is implemented in closed proximity and large geographical area with an increasing number of authorized nodes. We could observe a slight fall in the response time as more authorized nodes are added to the network. The decrease in response time is due to the availability of more miners to compete for mining, hence, reducing the mining time (which accounts for a large portion of the response time). The result further confirms that the architecture can facilitate scalable and prompt attack features exchange among IDS nodes.

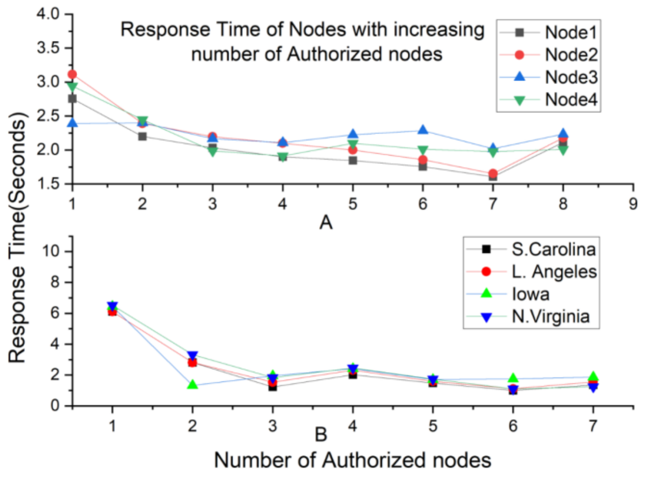

Fig. 8. A) Response time of nodes with an increasing number of authorized nodes for closed proximity implementation (B) Response time of nodes with an increasing number of authorized nodes for wide-area deployment

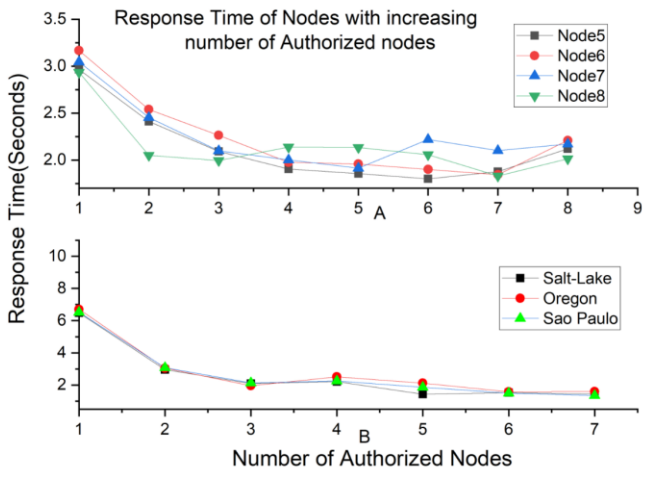

Fig. 9. A) Response time of nodes with an increasing number of authorized nodes for closed proximity implementation (B) Response time of nodes with an increasing number of authorized nodes for wide-area deployment

V. Conclusion

In this paper, we propose a permissionless public-private blockchain-based architecture that detects and prevents malicious activities on the stored data from both outsider and insider threats. The proposed solution, which focuses on network-based attacks, securely extracts, stores, and shares attack features in real-time with the view of enhancing the security of shared data in cooperative intrusion detection. We implement the architecture in closed proximity and a large geographical area, evaluate the security analysis and performance metrics. The result showed that the architecture facilitate scalable and prompt attack features exchange among IDS nodes, resistant to familiar insider and outsider attack threats and robust to public IDS nodes joining and leaving the network. Also, the result showed that the response time of the architecture decrease with an increasing number of miners.

In future we wish to expand our work to accommodate the following :

1. Implementing ways of increasing the throughput of architecture.
2. Develop an algorithm that restricts mining of similar attack features by different nodes

REFERENCES

[1] S. Peddabachigari, A. Abraham, C. Grosan, and J. Thomas, "Modeling intrusion detection system using hybrid intelligent systems," Journal of network and computer applications, vol. 30, no. 1, pp. 114–132, 2007.

[2] O. Igbe, O. Ajayi, and T. Saadawi, "Denial of Service Attack Detection using Dendritic Cell Algorithm" 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON 2017) Oct 19th – 21st 2017, Columbia University, New York, USA.

[3] O. Igbe, O. Ajayi, and T. Saadawi, "Detecting Denial of Service attacks using a combination of Dendritic Cell Algorithm(DCA) and Negative Selection Algorithm(NSA)" 2nd International Conference on Smart Cloud (Smart Cloud 2017) Nov 3rd-5th, 2017, New York, USA.

[4] F. Gong, ''Next-generation intrusion detection systems (IDS),'' McAfee Netw. Security. Technol. Group, Santa Clara, CA, USA, White Paper, 2003

[5] Y. L. Dong, J. Qian, M. L. Shi, "A cooperative intrusion detection system based on autonomous agents," IEEE CCECE 2003, Vol. 2, pp. 861– 863, 2003.

[6] C. C. Lo, C. Huang, J. Ku, A cooperative intrusion detection system framework for cloud computing networks, in: In: Proceedings of the 2010 39th International Conference on Parallel Processing Workshops,ICPPW '10, 2010, pp. 280-284.

[7] Y.-S. Wu, B. Foo, Y. Mei, and S. Bagchi, ''Collaborative intrusion detection system (CIDS): A framework for accurate and efficient IDS,'' in Proc. Annu. Comput. Secur. Appl. Conf. (ACSAC), Dec. 2003, pp. 234–244.

[8] O. Ajayi, M. Cherian and T. Saadawi, "Secured Cyber-Attack Signatures Distribution using Blockchain Technology," 2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), New York, NY, USA, 2019, pp. 482-488.

[9] Y. L. Dong, J. Qian, M. L. Shi, "A cooperative intrusion detection system based on autonomous agents," IEEE CCECE 2003, Vol. 2, pp. 861– 863, 2003.

[10] C. C. Lo, C. Huang, J. Ku, "A cooperative intrusion detection system framework for cloud computing networks," In Proceedings of the 2010 39th International Conference on Parallel Processing Workshops,ICPPW '10, 2010, pp. 280-284.

[11] W. Zhang, S. Teng, H. Zhu, D. Liu, "A Cooperative Intrusion Detection Model Based on Granular Computing and Agent Technologies", *J. International Journal of Agent Technologies and Systems,* vol. 5, no. 3, pp. 54-74, 2013

[12] S.R. Snapp, J. Brentano, GV dias, T.L. Goan, L.T. Heberlein, C. Ho, K.N. Levitt, B. Mukherjee, S.E. Smaha, T. Grance, D.M. Teal, and D. Mansur. DIDS (distributed intrusion detection system) — motivation, architecture, and an early prototype. In Proceedings of the 14th National Computer Security Conference, pages 167–176, October 1991.

[13] M. Uddin, A. Abdul Rehman, N. Uddin, J. Memon, R. Alsaqour, and S. Kazi, "Signature-based Multi-Layer Distributed Intrusion Detection" International Journal of Network Security, Vol.15, No.2, PP.97-105, Mar. 2013

[14] S. Nakamoto (2008) Bitcoin: a peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf

[15] T. Ahram, A. Sargolzaei, S. Sargolzaei, J. Daniels, and B. Amaba. "Blockchain Technology Innovation". 2017 IEEE Technology & Engineering Management Conference (TEMSCON), 2017

[16] Liang, X.; Zhao, J.; Shetty, S.; Liu, J.; Li, D. Integrating blockchain for data sharing and collaboration in mobile healthcare applications. In Proceedings of the 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), Montreal, QC, Canada, 8–13 October 2017

[17] Zikratov, I., Kuzmin, A., Akimenko, V., Niculichev, V., Yalansky, L.: Ensuring data integrity using Blockchain technology. In: Proceeding of the 20th Conference of fruct Association ISSN 2305-7254 IEEE (2017)

[18] M Signorini and M Pontecorvi, W Kanoun, and R Di Pietro, "BAD: a Blockchain Anomaly Detection solution" arXiv:1807.03833v2, [cs. CR] 12 jul 2018

[19] T. Golomb, Y. Mirsky and Y. Elovici " CIoTA: Collaborative IoT Anomaly Detection via Blockchain" arXiv:1803.03807v2, [cs.CY] 09 Apr 2018

[20] Gu, J, B Sun, X Du, J Wang, Y Zhuang and Z Wang (2018). Consortium blockchain-based malware detection in mobile devices. IEEE Access, 6, 12118–12128

[21] Abdullah, N., Hakansson, A., & Moradian, E. (2017). Blockchain based approach to enhance big data authentication in distributed environment. In Ubiquitous and future networks (icufn), 2017 ninth international conference on (pp. 887–892).

[22] S.R. Snapp, J. Brentano, GV dias, T.L. Goan, L.T. Heberlein, C. Ho, K.N. Levitt, B. Mukherjee, S.E. Smaha, T. Grance, D.M. Teal, and D. Mansur. DIDS (distributed intrusion detection system) — motivation, architecture, and an early prototype. In Proceedings of the 14th National Computer Security Conference, pages 167–176, October 1991.

[23] V. Yegneswaran, P. Barford, S. Jha, "Global intrusion detection in the DOMINO overlay system", Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS), pp. 1-17, 2004.

[24] Sultan Aldossary, William Allen. Data Security, Privacy, Availability and Integrity in Cloud Computing: Issues and Current Solutions. (IJACSA) International Journal of Advanced Computer Science and Applications,Vol. 7, No. 4, 2016 pp.485-498

[25] C. Wang, S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," Computers, IEEE Transactions on, vol. 62, no. 2, pp. 362–375, Feb 2013

[26] Ingo Weber, Vincent Gramoli, Mark Staples, Alex Ponomarev, Ralph Holz, An Binh Tran, and Paul Rimba. 2017. On Availability for Blockchain-Based Systems. In SRDS'17: IEEE International Symposium on Reliable Distributed Systems

[27] L. Dhanabal, S.P. Shantharajah, A study on NSL-KDD dataset for intrusion detection system based on classification algorithms, International Journal of Advanced Research in Computer and Communication Engineering 4 (2015) 446–452