2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)

Secured Cyber-Attack Signatures Distribution using Blockchain Technology.

Oluwaseyi Ajayi, Melvin Cherian and Tarek Saadawi

Department Of Electrical Engineering City University New York , City College, New York, NY 10031

Oluwaseyi.j.ajayi@gmail.com

***Abstract*—The proliferation of cloud database has increased its vulnerability to cyberattacks. Despite several proposed methods of securing databases, malicious intruders find ways to exploit their vulnerabilities and gain access to data. This is because cyberattacks are becoming more sophisticated and harder to detect. As a result, it is becoming very difficult for a single or isolated intrusion detection system (IDS) node to detect all attacks. With the adoption of cooperative intrusion detection system, all attacks can be detected by an IDS node with the help of other IDS nodes. In cooperative intrusion detection, IDS nodes exchange attack signatures with the view of promptly detecting any attack that has been detected by other IDS. Therefore, the security of the database that houses these shared attack signatures becomes a huge problem. More specifically, detecting and/or preventing malicious signature injection, manipulation or deletion becomes important. This paper proposed an architecture that securely stores and distribute these attack signatures in real time for the purpose of prompt detection. Our proposed architecture leverages the distributed ledger technology, data immutability and tamper-proof abilities of blockchain technology. The performance of our system was examined by using the latency of the blockchain network.**

Keywords—cooperative intrusion detection; cyberattack; signature; blockchain; latency

I. INTRODUCTION

The rapid increase in the use of internet has led to its popularity in recent years. As a result of this, companies rely on internet for storage of their data because it is secured and easily accessible. Owing to this, malicious intruders exploit vulnerabilities of this internet to gain unauthorized access to data stored. Firewalls, user authentication and data encryption[1] were proposed to restrict unauthorized users from gaining access to databases. Further researches put forward intrusion detection system (IDS)[2,3] to detect any malicious activities in computer networks and database housing data. This intrusion detection can be classified according to their location in the network: host-based (HIDS) or network-based (NIDS)[4] or according to their detection approaches: signature-based or anomaly-based IDS[5]. Although these intrusion detection systems have shown their capability of protecting computer networks, however, a single or isolated IDS may be easily bypassed by advanced attacks as the malicious activities gets more complex[6]. In addition, the timely detection of attacks is very important as this may cause huge damage to computer networks if they are not detected on time. To enhance the detection capabilities, cooperative intrusion detection was proposed[7-9]. In this type of detection mechanism, different IDS nodes exchange attack signatures with the view of detecting any attack that has previously been detected by other IDS nodes. This detection mechanism was widely adopted due to its better performance. However, attack signatures being shared are vulnerable to (i) data manipulation during transmission, (ii) fake data injection if database is opened to public (iii) data deletion if database activities are not monitored. Owning to this, security of database and its transmission media are of great concern.

Companies involved in collaborative intrusion detection believes that their data is secured as far as it is encrypted. Although encryption guarantees confidentiality of such data, but their consistency and integrity are not guaranteed. A research proposed the use of a message authentication code algorithm (MAC) as way to secure data integrity [10]. In this method, data file is downloaded, and hash value is checked. Although this method works but downloading and checking hash of files are overwhelming process which requires high bandwidth. Another research computes the hash value of every data by using hash tree. This is also not practical because computing the hash value of huge data requires high computing power and consumes high bandwidth. Some databases employed third party auditor. In this method a person that has skills and experience carry all auditing processes such as checking data integrity. This method achieves desired result but there is need for third party which can expose the data to "man-in-the-middle" attack [10]. Above solutions are not practical especially for large data files, hence, reason for our proposed solution. Our approach leverages distributed ledger technology, data immutability and tamper-proof abilities of blockchain to solve these problems.

Blockchain was first introduced in 2009 as technology behind bitcoin[11]. The author implemented a solution to double spending problem in cryptocurrency called bitcoin. Since its inception, blockchain has been applied to different areas by several researchers e.g in healthcare system[12,13], securing data integrity[10], as IDS [14,15,16] etc. In its simplest form, blockchain is an append-only, public ledger which records all the transactions that has taken placed in the blockchain network. Every participant in the blockchain network are called nodes. Each transaction in the public ledger is verified by consensus (an agreement among all participating nodes) of most of the participants in the system. Once the transaction is verified, it is impossible to mutate / erase the records. The blockchain contains a certain and verifiable record of every single transaction ever made [11]. The data in the blockchain (i.e. transactions) is divided into blocks. Each block is dependent on the previous one (parent block). Every block stores some metadata and the hash value of the previous block. So, every block has a pointer to its parent block. Blockchain is broadly divided into two: public and private blockchain[17]. In public blockchain, all nodes verify and validate transactions. They are also known as permissionless blockchain. Examples are Bitcoin, Ethereum etc. while in private blockchain, only nodes given permission can join and participate in the network. It is usually controlled by the node that starts the network. It is usually built for certain purposes. Example is Hyperledger.

978-1-7281-1664-8/19/$31.00 ©2019 IEEE
DOI 10.1109/CSE/EUC.2019.00095

IEEE computer society

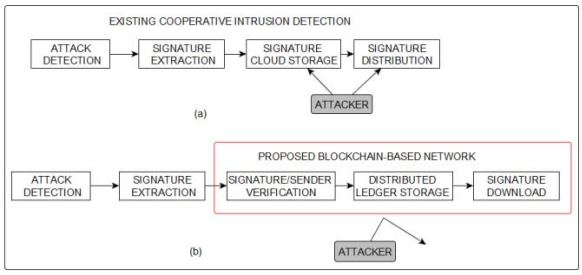

Fig.1 (a) cyber-attack targets of existing cooperative intrusion detection (b) Blockchain-based cooperative intrusion detection

Basically, there are four steps involved in existing cooperative intrusion detection system (Fig.1a). The main targets of cyber-attackers are storage and distribution steps (Fig.1a). Majority of the available solutions to secure these cyberattack targets either engage centralized approach which makes the network vulnerable to single point of failure attacks or uses decentralized approach in which both database and communication medium cannot ascertain the consistency and integrity of distributed data. Also, existing solutions do not accommodate nodes running different IDS. In this work, we are proposing an architecture which securely store and distribute attack signatures using blockchain technology. The purpose of our approach is to provide security to signature storage and distribution as shown in Fig.1b. In addition, the architecture accounts for IDS nodes running different IDS. This is the motivation to this work.

The contributions of our work can be summarized as follows:

- To develop a private-public blockchain-based architecture that retrieve attack signature from any signature-based IDS e.g. Snort[24], automatically converts to a common format compatible with other signature-based IDS e.g. Bro[25], Suricata[26] etc. This accounts for nodes running different IDSs.
- This architecture presents a standard format for storing and distributing signature-based IDS attack signatures in order to accommodate nodes running diverse IDS
- The blockchain architecture allows public node to securely join and assess stored attack signatures in real time without permission.

The remainder of this paper is organized as follows: background and related works on cooperative intrusion detection and blockchain applications are discussed in Section II. Section III describes the proposed architecture. Section IV presents performance metrics of our architecture. Section V presents the conclusions of this paper and possible future works and section VI is acknowledgment.

II. BACKGROUND

A. Cooperative intrusion detection

The authors in [7] proposed Cooperative Intrusion Detection System (CoIDS) which uses a cooperative approach for intrusion detection . In their method, individual intrusion detection components work cooperatively to perform concerted detection. The result showed that their system is efficient and effective to prevent viruses spreading in chain way. However, with the introduction of intrusion detection manager (IDM), who maintains and update data including cooperative protocols, rules and logs, the need to trust IDM makes the network vulnerable to attacks such as man-in-the-middle and single point of failure. In [8] the authors proposed cooperative intrusion detection framework in cloud computing to reduce the impact of denial of service attacks (DoS) and distributed denial of service attacks (DDoS). In their system, each IDS has a cooperative agent that computes and determine whether to accept the alerts sent from other IDS. The result showed that their proposed system only increases little computation effort compared with pure snort-based IDS but prevents the system from single point of failure attack. Although their system showed a promising result, but the method fails to consider situation when each cooperative agent uses different IDSs. Also, their system is susceptible to intrusion detection activities such as data hijacking via medium of transmission.

In another research [18], the authors proposed a cooperative intrusion detection based on granular computing. In their work, they analyzed four different attacks; probing, distributive denial of service, Remote to local (R2L) and user to Root (U2R). They divided the attacks to one host-one host, one host-many hosts, many hosts-one host and many hosts-many hosts, based on source and destination addresses of the network packages. The result showed that their method can detect slow scanning attacks which cannot be detected by a traditional scanning detector. However, with response unit updating the database, this unit and database can be hacked which may result in data manipulation and deletion. The authors in [19] proposed signature-based multilayer IDS using mobile agents. In their works, they created small multiple databases from huge database of attack signatures for efficient threat detection in computer networks. They also proposed a mechanism to automatically update these small signature databases using mobile agents. The analysis of their results showed that proposed architecture led to a significant decrease in packet drop rate as compared to conventional signature-based database. This resulted in a significant improvement in detection of malicious attacks to the network. Although, this approach reported good results, but malicious activities pose a great threat to the database that houses these signatures

B. Blockchain appliation

The authors in [14], [15] and [16] proposed use of blockchain technology in detecting anomaly attack. In [14], the authors proposed a blockchain anomaly detection solution (BAD) which focus on detecting attacks directed at the blockchain network. BAD prevents insertion of 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). Attacker can manipulate and filter victim's incoming connection. The analysis of result showed that BAD was able to detect and stop the spread of attackers that uses bitcoin forks to spread malicious codes. However, proposed solution is specific to attacks directed towards blockchain network and use bitcoin forks. The authors in [15] 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 use it to build feature database. Their result showed that their solution can detect and classify known malwares. It also performs malice determination and malware family classification on unknown software with higher accuracy and lower time cost. However, their solution is specific to host-based malware attacks on android-based mobile devices. The proposed solution is not applicable to network-based attacks.

The authors in [16] proposed collaborative IoT anomaly detection via blockchain solution (CIoTA). CIoTA uses 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 can detect malware activities easily with zero false positive. However, the solution proposed relies mainly on collaborative effort of IoT to detect attacks (i.e. every node works together to detect an attack) and it is specific to malware attacks

In areas highlighted above, application of blockchain has been proven to be efficient and effective. Although much efforts are being shifted to applying the technology in cyber-security, most of available solutions used private blockchain platforms to either combine wisdom to detect attack[16] or detect specific attacks [14,15]. Also, their cooperative intrusion systems did not account for nodes running different IDSs. In this work, we are proposing a permissionless public-private blockchain-based architecture which securely store and distributes attack signatures in a distributed network in real time. Also, our proposed system also accounts for nodes running different IDS by converting attack signatures to standard format compatible with any signature-based IDS. This is the novelty in our work.

III. THE PROPOSED ARCHITECTURE

The proposed architecture, which was built on Ethereum blockchain platform, combines features of both public and private blockchain to extract, convert, store and distribute cyberattack signatures. Ethereum platform was used because it is popular and has potential in being developed to use in a wide variety of application. Also, Ethereum handles a greater number of concurrent transactions which makes it scalable [23]. It is an open-source blockchain based distributed computing featuring smart contracts. Smart contract is an agreement among the members of consortium which is stored on the chain and run by all participants [21,22, 23]. Although the main Ethereum platform is a public blockchain, we configured the network to a combination of public and private networks. It is a private blockchain because certain nodes can prepare, verify and validate transactions. It is regarded as public blockchain because nodes do not need permission to join or leave the network and obtain the content of the mined blocks. Fig. 2 shows a pictorial representation of the proposed architecture.

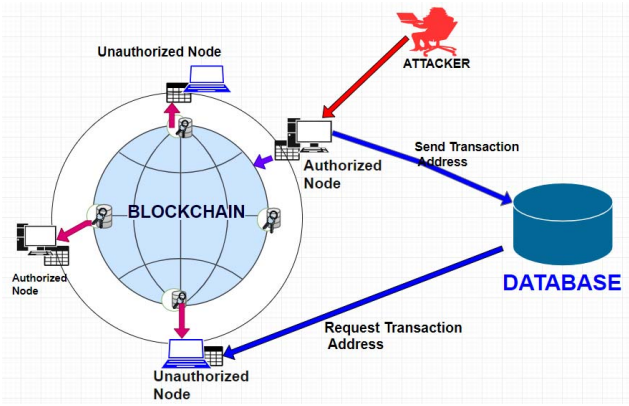

Fig. 2 The Proposed Architecture

The architecture is composed mainly of the following:

- **Authorized Nodes**
  These are nodes that start the blockchain network. They prepare, submit and verify transactions. These nodes also run consensus algorithm, thus validate transactions/blocks. All authorized nodes update database

- **Unauthorized Nodes**
  These are public nodes. They do not need permission to join or leave the blockchain network. They join the network to download cyberattack signatures. They are not privileged to prepare, verify, validate or run consensus algorithm. they cannot update database but request transaction address of mined transactions/blocks.

- **Database**
  The database stores address of transaction, smart contract and their Application Binary Interface (ABIs). The database is accessible to all nodes. Every public node has read-only access to this database. All information is updated by authorized nodes. Any data manipulation in database results in inability to access contents of blockchain but does not affect data stored in the blockchain. Such malicious activity can be easily detected

The proposed architecture is divided into 3 stages as shown below.

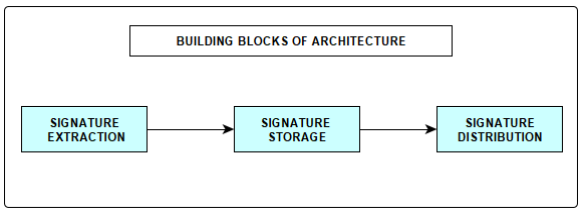

Fig. 3 Building blocks of the proposed architecture

A. Signature Extraction

In our proposed system, extraction of signature is done by authorized nodes that detect attacks. Whenever there is an attack detection, IDS signature that detected that attack is retreived from IDS node (i.e. authorized nodes). Extracted signature is formatted as given in Table I

Table I. Format of the retrieved Signature/Rule

S/N	Type	Signature

- *S/N*: This is serial number of the retrieved signature.
- *Type*: The name of IDS that detects attack. For this proof of concept, we experimented with Snort, Bro and Suricata.
- *Signature*: The actual signature that was retrieved is placed in this column.

The owner signed signature with its private key and submit to the blockchain for verification. In addition to signed signature, owner also submits its MAC address, IP address and Transaction account. Table II shows format of submitted transaction by authorized node.

Table II. Format of Submitted Signature

S/N	Type	$SIGNATURE_{priv\_key}$	MAC address	IP address	Transaction Account

B. Signature Storage

The submitted transaction (i.e. Table II) and owners' privilege are verified, signature is converted to standard format and validated. All verification and signature conversion are handled by smart contract, while validation is handled by blockchain consensus protocol. The accepted format of submitted signature (Table II), transaction account, MAC and IP addresses of all authorized nodes, signature conversion script (Algorithm 2) and signature format creation script (Algorithm3) were all written into the smart contract and mined to the blockchain by authorized nodes. In our architecture, smart contract handles the following functions:

1. *Transaction and owner's verifications:* This step ensures that no unauthorized nodes submit transaction. Algorithm 1 describes how smart contract handles all verifications. For transaction verification to return success and push for signature conversion, submitted transaction must agree with standard format (Table II). Transaction account, MAC and IP addresses of sender must be in their respective sets. Also, private key of sender must be verified by its public key. If any of these fails, the algorithm returns fail and transaction is dropped.

2. *Signature format creation:* This is the novelty in our approach. We experimented with cyber-attack signatures from three common signature-based IDS: Bro, Snort and Suricata. We developed script that convert signatures from one IDS to a common format compatible with other IDSs (Algorithm 3). This script is mined to the blockchain through smart contract. Algorithm 2 describes how smart contract converts attack signatures to standard format. When transaction and sender's verification is successful, *type* field of submitted transaction is examined. If *type* field reads Snort, this shows signature was obtained from Snort IDS, our architecture converts to standard format compatible with Bro and Suricata etc. The Standard format of transaction submitted for validation is shown in Table III.

Algorithm 1: Transaction Verification

Procedure: Verification (**Transaction, MAC, IP, TA**)
Inputs: Transaction, MAC address (**MAC**),IP address (**IP**) and Transaction Account (**TA**)

```
If (Transaction agrees with Standard format) and
    (IP in IP sets) and ( MAC in MAC sets) and
    (TA in TA sets) and (public key verifies private key):

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

Algorithm 2: Signature Format Creation

Procedure: Standard Format Creation by checking ***type*** field of verified transaction
Input: Verified Transaction
Output: Standard Format

```
If type is Snort:
    Conversion script: converts to format compatible with
    Bro and Suricata
else:
    if type is Bro:
        Conversion script: converts to format compatible with
            Snort and Suricata
    else:
        Conversion script: converts to format compatible with
            Snort and Bro
    endif
endif
Snort: signature ||Bro: Signature || Suricata: signature
end procedure
```

Table III. Format of the mined Transaction

Snort: *signature*	**Bro:** *signature*	**Suricata:** *signature*

Snort: indicates the name of IDS and *signature* is the actual signature

Algorithm 3: Signature Conversion Script

Procedure: Converts IDS signature to Standard Format based on common fields

Inputs: Variables of retrieved IDS (IDS_{var}) and values of retrieved IDS(IDS_{value})

Outputs: Variable of standard format (SF_{var})

Mandatory variable: (protocol, source IP, source port, destination IP, destination port, message, option)

```
Read IDS_var
If any mandatory variable not in IDS_var :
    return error
    drop signature
else:
    For IDS_value in IDS_var :
        Read IDS_value
        Assign IDS_value to equivalent SF_var
        Ignore SF_var with no equivalent IDS_value
    end for loop
end if
end procedure
```

Algorithm 3 describes how an attack signature is converted to standard format. Variables of retrieved signature are checked for mandatory variables. If any of the mandatory variables are absent, script returns error and signature is dropped. Otherwise, script reads these values and assign to corresponding standard format variables. Due to different ways of writing rules, we ignore any standard format variables without equivalent values from retrieving signature.

3. *Transaction Validation*: The pending transaction is built into a block by authorized node after successfully converted to standard format. The block is broadcasted into the blockchain network for validation. Every node receives broadcasted block, but only authorized nodes (miners) work to validate the block. Each block contains a unique code called hash; it also contains hash of previous block. Data from previous blocks are encrypted or hashed into a series of numbers and letters. This is done by processing the block input through a mathematical function, which produces an output of a fixed length. The function used to generate the hash is deterministic, meaning that it produces the same result each time the same input is used; makes determining the input difficult and makes small changes to the input result in a very different hash.

   To validate the block, authorized nodes works to get target hash. 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. This is achieved by using an iterative process such as proof-of-work, which requires consensus from all authorized nodes. Proof-of-work was chosen because this is the consensus algorithm run by Ethereum. The characteristics of proof-of-work is it computationally difficult to compute and easy to verify. The process of guessing the hash starts in the block header. It contains 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. If the new hash meets the requirements set forth in the target, then the block is added to the blockchain. it is said to be permanently stored on the network. it is impossible to mutate / erase the block.

C. Signature Distribution

After new block has been chained to the blockchain, transaction address is issued to owner (sender). The blockchain is updated and signature is ready to be downloaded. These are summarized in the following steps:

1. *Blockchain updating:* Current state (i.e. the update of new block) of the blockchain is broadcasted to every node in the blockchain network. Every node (authorized and unauthorized) in the blockchain network receives a copy of the update. Transaction address and Application Binary Interface (ABI) are sent to the database by the owner. This database is made public so that everyone can have access to these information (Fig. 2).

2. *Signature Downloading:* This step is carried out by every node in the network (i.e. authorized and unauthorized). Blockchain nodes request transaction address and ABI from database to download attack signatures mined into the blockchain by other members of the consortium. Nodes extract signatures from downloaded transaction based on IDS utilized.

IV. PERFORMANCE METRICS

Our proposed system was implemented on Ethereum blockchain platform. We use Solidity *v 0.5.4* implementation of Ethereum for our smart contract and *geth v 1.4.18* for Ethereum. Blockchain network was set up in the laboratory with five blockchain nodes, one database node and one attack node as shown in Fig. 2. We implemented four authorized nodes (to ensure consensus of miners during validation stage) and one unauthorized (public) node in the set up. To make a node authorized, we code transaction account, MAC and IP addresses into smart contract and mined it to the blockchain network. Four blockchain nodes run ubuntu 18.04 while one runs ubuntu 16.04. Each node has the following configurations.

Authorized node 1: Desktop, Ubuntu 16.04, 4GB RAM, 2.2GHz processor speed and 300GB hard drive. Authorized node 2: Laptop, Ubuntu 18.04, 16GB RAM, 2.81GHz processor speed, and 2TB hard drive. Authorized node 3: Desktop, Ubuntu 18.04, 8GB RAM, 2.44GHz processor speed, and 1TB hard drive. Authorized node 4: Laptop, Ubuntu 18.04, 4GB RAM, 2.44GHz processor speed, and 500GB hard drive. Unauthorized node: Laptop, Ubuntu 18.04, 4GB RAM, 2.4GHz processor speed, and 300GB hard drive. Database node (desktop) runs window 10 with processor speed of core i5 @ 2.44GHz, 4GB RAM and 500GB hard disk. Attack node: Laptop, Ubuntu 16.04, 4GB RAM, 2.20GHz processor speed, and 300GB hard drive. *Snort v2.9.7* was installed on authorized nodes 1,4 and unauthorized node, *Bro v 2.6.1* was installed on authorized nodes 2, 3 and unauthorized node and *Suricata v 4.1.3* was installed on authorized node 4. Transaction account, IP address and MAC address of all authorized node were written as a smart contract and mined into the blockchain.

In authorized node 2, Denial of Service (DoS) attack rule shown below was set at the local rule files of its snort IDS and snort was started in monitoring mode.

Attack rule*: alert tcp ! $ any any -> $HOME_NET 80 (flags: S; msg:"Possible DoS"; flow: stateless; threshold: type both, track by_src, count 70, seconds 10; sid:10001;rev:1;)*

We launched DoS attack from attack node at authorized node2. Authorized node2 detected this attack, retrieved above signature and submitted it as a transaction to already set up blockchain network. Three other authorized nodes verified and mined transaction to the blockchain network. This was repeated 10 more times. For each transaction, we checked conversion results to standard format described above. The following assumption was made:

1. *None of the authorized nodes is compromised i.e all signature submitted are good signatures.*

To evaluate performance of our system, following data were collected for each transaction.

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

A. Latency: This is response time (measured in seconds) of the blockchain network. For each transaction, latency is the difference between execution time and deployment time (t_3-t_1). Latency time includes signature conversion time, mining time and time taken for nodes to send signatures to their IDS rule files. Fig. 4 shows response time of each node for every transaction. It was observed that the latency of 6th transaction was the smallest while 4th and 8th transaction has the highest latencies for all node. It has been shown that mining time (i.e. time to guess target hash (proof-of-work)) has great influence on latency of blockchain network, hence, the reason for high latency. Also, verification time (format conversion) influnce the network latency. Figs. 5 shows the average response time of each node. The average response time is addition of all response times for each transaction divided by number of transactions. It can be seen that average response time for all nodes is less than 3 seconds

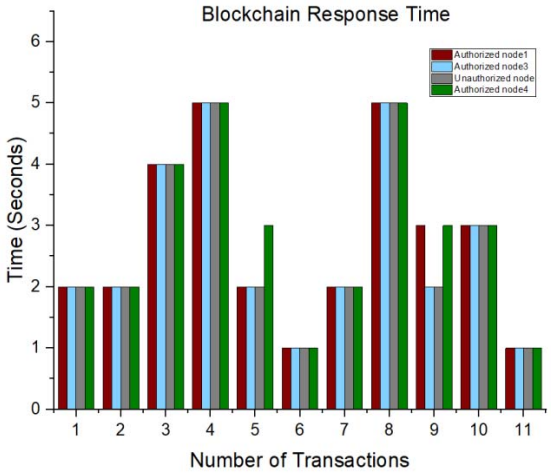

Fig 4. Blockchain Response Time

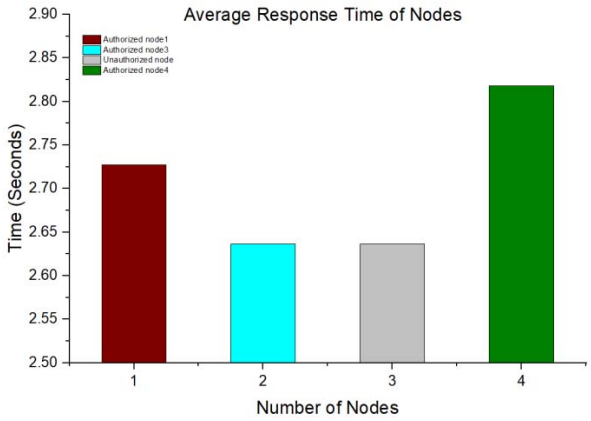

Fig 5.Average response time of each node

V. CONCLUSION

In this paper, we proposed a permissionless public-private blockchain-based architecture that securely share attack

signatures with other public nodes irrespective of IDSs utilized. The proposed solution securely stores and shares attack signatures in real time with the view of combating security concerns associated with cooperative intrusion detection. We focused on nodes that run signature based IDSs. Our proposed solution uses blockchain to convert cyberattack signatures to standard format compatible with other IDSs before sharing with public. We presented a standard format for converted cyberattack signatures. We tested the performance of our architecture using latency. The proposed architecture does not only help in prompt detection of cyberattacks among the cooperative nodes but also combat the security concerns associated with it.

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

1. Optimization of performance using some other time and energy-efficient consensus protocols.
2. Developing a transaction standard format for nodes running different anomaly-based intrusion detection system.
3. Develop an algorithm that restrict mining of similar attack signatures by different nodes
4. Implement scalability of the blockchain network.
5. Implement detection of compromised authorized nodes.

VI. AKNOWLEDGMENT

I want to appreciate other members of the team; Dr. Obinna Igbe and Chantelle Levy, for their supports and efforts to make the work successful.

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. Secur. Technol. Group, Santa Clara, CA, USA, White Paper, 2003

[5] O. Igbe, I. Darwish, and T. Saadawi, "Distributed network intrusion detection systems: An artificial immune system approach," in Connected Health: Applications, Systems and Engineering Technologies (CHASE), 2016 IEEE First International Conference on. IEEE, 2016, pp. 101–106.

[6] C. Duma, M. Karresand, N. Shahmehri, and G. Caronni, ''A trust-aware, P2P-based overlay for intrusion detection,'' in Proc. DEXA Workshop, 2006, pp. 692–697.

[7] 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.

[8]] 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.

[9] 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.

[10] 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)

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

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

[13] 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

[14] 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

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

[16] 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.

[17] 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).

[18] 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

[19] M. Uddin, A. Abdul Rehman, N. Uddin, J. Memon, R. Alsaqour, and S. Kazi, "Signature-based Multi-Layer Distributed Intrusion Detection System using Mobile Agents" International Journal of Network Security, Vol.15, No.1, PP.79-87, Jan. 2013

[20] R. Bye, S.A. Camtepe, and S. Albayrak, "Collaborative intrusion detection framework: characteristics, adversarial opportunities and countermeasures." In Proceedings of CollSec: Usenix Workshop on Collaborative Methods for security and privacy. USENIX, Washington, DC, USA; 2010

[21] 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

[22] S. Pongnumkul, C. Siripanpornchana, and S. Tajchayapong, "Performance analysis of private blockchain platforms in varying workloads," in Proceedings of the 26th International Conference on Comp

[23] Zhang, P., Walker, M., White, J., Schmidt, D.C., and Lenz, G.: 'Metrics for assessing Blockchain-based healthcare decentralized apps', Proceedings of 2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom), October 12-15, 2017, Dalian, China

[24] G.D. Kurundkar, N.A. Naik, and S.D. Khamitkar, "Network Intrusion Detection using SNORT" International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 2, Issue 2,Mar-Apr 2012, pp.1288-1296

[25] V. Paxson. "Bro: a system for detecting network intruders in real time. "Computer Networks, 31(23-24), December 1999.

[26] R. McRee, " Suricata: An Introduction" Information Systems Security Association Journal, USA. August 2010