Consortium Blockchain-Based Architecture for Cyber-attack Signatures and Features Distribution

Oluwaseyi Ajayi
Department of Electrical Engineering, City College of New York,
New York, USA
Oluwaseyi.j.ajayi@gmail.com

Obinna Igbe
Department of Electrical Engineering, City College of New York,
New York, USA
Obiigbe91@gmail.com

Tarek Saadawi
Department of Electrical Engineering, City College of New York,
New York, USA
sadaawi@ccny.cuny.edu

***Abstract*—One of the effective ways of detecting malicious traffic in computer networks is intrusion detection systems (IDS). Though IDS identify malicious activities in a network, it might be difficult to detect distributed or coordinated attacks because they only have single vantage point. To combat this problem, cooperative intrusion detection system was proposed. In this detection system, nodes exchange attack features or signatures with a view of detecting an attack that has previously been detected by one of the other nodes in the system. Exchanging of attack features is necessary because a zero-day attacks (attacks without known signature) experienced in different locations are not the same. Although this solution enhanced the ability of a single IDS to respond to attacks that have been previously identified by cooperating nodes, malicious activities such as fake data injection, data manipulation or deletion and data consistency are problems threatening this approach. In this paper, we propose a solution that leverages blockchain's distributive technology, tamper-proof ability and data immutability to detect and prevent malicious activities and solve data consistency problems facing cooperative intrusion detection. Focusing on extraction, storage and distribution stages of cooperative intrusion detection, we develop a blockchain-based solution that securely extracts features or signatures, adds extra verification step, makes storage of these signatures and features distributive and data sharing secured. Performance evaluation of the system with respect to its response time and resistance to the features/signatures injection is presented. The result shows that the proposed solution prevents stored attack features or signature against malicious data injection, manipulation or deletion and has low latency.**

Keywords— Features, Signatures, Cyberattacks, Blockchain, IDS, Cooperative intrusion detection, Latency, Security, Data injection, Permissionless, Data consistency, Data integrity, Malicious activities.

I. INTRODUCTION

Computer networks are still experiencing cyberattacks despite their protections with different multilayer security infrastructure which includes intrusion detection system (IDS). Although IDS has been proven to be useful in identifying malicious activities, their abilities to detected coordinated or distributive are impaired because they have only single viewpoint. This has made it possible for some attacks to go undetected or not detected on time. Also, a zero-day attack (attack without known signature) experienced in an organization's IDS located, say in New York, USA might be different from that experienced in another organization's IDS located, say London, United Kingdom or another company located in the same region. Therefore, if this threat information is exchanged among the organization's IDS, more malicious activities can be stopped by coordinating efforts of participating IDS. To improve detecting power of single IDS, cooperative intrusion detection was proposed[1-3]. In cooperative intrusion detection system, IDS nodes exchange attack features or signatures among each other with the view of detecting attacks that have previously been detected by other IDS nodes. Cooperative intrusion was adopted because of its enhancement in detection rate of single IDS. However, malicious activities such as fake data injection, data manipulation or deletion and data consistency are some of the major problems facing this approach.

The main vulnerable stages in existing cooperative intrusion detection system are storage and distribution stages [4] (Fig 1a). Most of existing approaches to secure these target phases either utilize a centralized approach (which makes the network vulnerable to single point-of-failure and man-in-the-middle attacks [1,5,15]) or uses decentralized approach in which the integrity and consistency of the shared data cannot be guaranteed [2,6]. Several researches have been put forward to secure shared data in cooperative intrusion detection. Authors in [25] proposed message authentication code (MAC). Although this method detects accidental and intentional changes in the data, downloading and calculating MAC of large files is overwhelming and time consuming. Another method described in [25] to secure the integrity of cloud data is to compute the hash values of every data in the cloud by using hash tree. Although this solution is lighter as compared to first method, it is also not practical because computing the hash values of huge data requires more computation power and it is time consuming. The authors in [26] employs third party to coordinate activities of database. The problem with this approach is the need to trust third party which expose data to man-in-the-middle attack or the network to a single-point-of-failure attack. To solve these problems, we propose a solution which leverages distributive ledger technology, data immutability and tamper-proof abilities of blockchain technology to securely extract, store and distribute cyberattack features and signatures among nodes in real time (Fig. 1b). We define attack features as characteristics of attacks, retrieve from

978-1-7281-3885-5/19/$31.00 ©2019 IEEE

attacks traffic detected by anomaly-based IDS while attack signatures are predefined rules obtain from signature-based IDS.

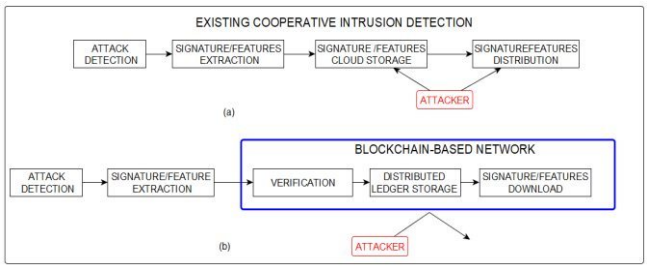

Fig. 1.(a)Cyber-attack targets of existing cooperative intrusion detection. (b) Blockchain-based solution for cooperative intrusion detection [4].

The contributions of our work can be summarized as follows:

- We propose a private-public blockchain-based architecture that automatically retrieves attack features from attack traffic detected by any anomaly-based IDS or retrieves attack signature from signature-based IDS.
- The architecture automatically verifies integrity and consistency of retrieved features or signatures and present in standard format compatible with other IDS nodes.
- The verified attack features and signature are securely stored in a blockchain network.
- The architecture grants permissionless access to any public node to securely join the blockchain network and obtain stored attack features or signatures in real time.

Blockchain technology was first implemented to solve double spending problem in cryptocurrency called bitcoin in 2009 [7]. Blockchain network is an append-only, public ledger that keeps records of transaction that has occurred in the network. Every participant in a blockchain network are called nodes. The data in blockchain network is known as transaction and it is divided into blocks. Each block is dependent on previous one (parent block). Every block stores some metadata and hash value of previous block. So, every block has a pointer to its parent block. Each transaction in the public ledger is verified by consensus (i.e. an agreement) of most of participants in the system. Once transaction is verified, it is impossible to mutate / erase the records [7]. Blockchain technology is broadly divided into two: public and private blockchain[8]. Public blockchain is a permissionless blockchain in which verification and validation of transactions are done by all nodes. 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. Blockchain has been applied to diverse areas since its inception in 2009 e.g. health system [9,10], intrusion detection system [11-13], and data integrity security [14, 24].

The remainder of this paper is organized as follows: related works on cooperative intrusion detection are discussed in Section II. Section III describes proposed architecture. Section IV presents results. Section V presents conclusions of this paper and possible future works.

II. RELATED WORKS

A. Cooperative Intrusion detection

Authors in [1] 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 in preventing viruses spreading in chain way. However, with introduction of intrusion detection manager (IDM), who maintains and update data including cooperative protocols, rules and logs, there is need to trust IDM which may expose the system to attacks such as man-in-the-middle or single-point-of-failure. In another research put forward in [2], 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 compute 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 shows a promising result, they failed to consider situation when each cooperative agent uses different IDSs. Also, their system is susceptible to malicious intruder activities such as data hijacking via medium of transmission.

A cooperative intrusion detection based on granular computing was proposed in [5]. 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 hosts 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, response unit and database are susceptible to hacking, data can be injected, manipulated and deleted. In [15] the authors 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. They considered how to track a user moving across network with a new user-id on each computer. The result showed that their prototype demonstrate viability in solving network-user identification problem. However, with DIDS director responsible for all evaluation, the system is vulnerable to single-point-of-failure or man-in-the-middle attacks.

In [25], the authors described two models of proving the integrity of data. In the first model, the file is downloaded, and the hash is checked. A message authentication code algorithm (MAC) is used. The data owner downloads outsourced data and then calculates the MAC. By using this method, accidental and intentional changes can be detected. However, downloading and calculate the MAC of huge data is overwhelming, requires more bandwidth and time consuming. The second method computes the hash value in the cloud using a hash tree. This is also not practical as

computing the hash values of large data requires more computation. The authors in [26] proposed the use of third-party auditor (TPA) to check the data integrity of stored data. The problem with this approach is the need to trust third party which expose data to man-in-the-middle attack or the network to a single-point-of-failure attack. Recently, more attention has been drawn to remote data auditing by which data integrity and correctness of remotely stored data is investigated [27], [28] and[29].

Despite the efforts in the existing solutions, cyber attackers can explore the vulnerabilities of the systems and compromise stored data in some of the solutions, while in others, most of the solutions are not practical especially for real-time and huge data. Thus, data security and consistency problems are not completely eradicated. Hence, the motivation for this work.

III. THE PROPOSED ARCHITECTURE

The proposed architecture is built on Ethereum blockchain platform. It combines characteristics of both private and public blockchain to store and distribute cyberattack features and signatures. 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. Ethereum blockchain handles a great number of concurrent transactions which makes it scalable [16]. It is an open source blockchain based distributed computing featuring smart contracts. Smart contract is an agreement among members of consortium which is stored on the chain and run by all participants[17]. Although the main Ethereum platform is a public blockchain, in this paper it is configured as public-private blockchain networks. Fig. 2 shows a pictorial representation of the proposed architecture.

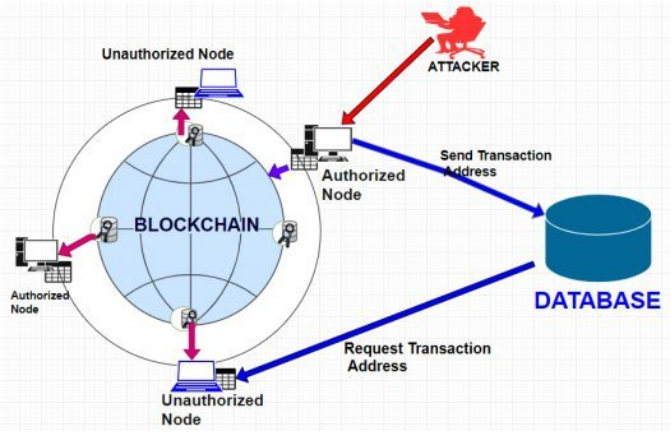

Fig. 2. The Proposed Architecture

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 retrieve stored signatures or features. They are not privileged to prepare, verify, validate or run consensus algorithm. They do not update the database but can only request transaction address of mined transactions/blocks.

- **Database**
  Database; which is accessible to all nodes, stores address of transaction, smart contract and their Application Binary Interface (ABIs). Every public node has read-only access to it. All information is updated by authorized nodes. Any data manipulation in database results in inability to access 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 3 stages as shown below.

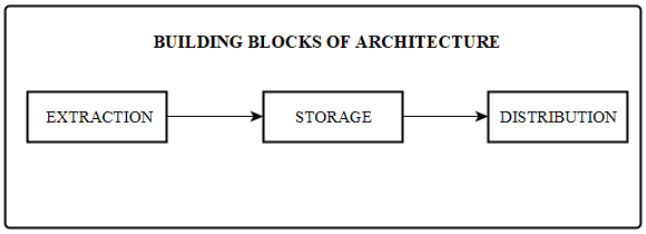

Fig. 3. Building blocks of the proposed architecture

A. Extraction

This stage is divided into two: signature extraction and features extraction.

1. *Signature Extraction:* Attack signature or rule is extracted from a signature-based IDS e.g. Snort[18], Bro[19], Suricata[20] etc. Whenever an attack is detected, the signature that was used to detect such an attack is retrieved from the IDS. An example of a retrieved signature is presented as shown in Fig. 4.

S/N	Type	Signature

Fig. 4. Retrieved signature

   - *S/N*: Serial number of retrieved signatures.
   - *Type*: The name of IDS that detected the attack. We experimented with Snort, Bro and Suricata.
   - *Signature*: The actual signature that is retrieved is placed in this column

2. *Feature Extraction*: Attack features are characteristics of an attack traffic that differentiate it from a normal traffic. Anomaly-based IDS is trained with features extracted from normal traffic, then used to detect any deviation from known traffic pattern. Anomaly-based IDS has been shown to have capabilities of detecting zero-day attacks (i.e. attacks with unknown signatures) with high accuracy [21,22]. In this work, network-based attack features are extracted based on feature

names proposed in [23] using network traffic analyzing tools. Attack features are extracted under two categories: (i) Connection features and (ii) packet features

(i) Connection features: These are attack features that are obtained from attack network connections. We develop a script that sniffs and analyzes network traffic using *tcpdump v. 4.9.2., libpcap v. 1.9.0, tcptrace 6.6.0* and *wireshark v. 3.0.1*. Tcpdump is installed to analyze tcp packets while wireshark uses libpcap to capture attack traffic in real time. Tcptrace is used to analyze the capture attack traffic. When an attack is detected, attack traffic is sniffed, captured and network connections are analyzed. Some of features extracted from attack network connection are shown in Table I

Table I: Attack Network Connection Features

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

(ii) Packet features: These are attack features obtained by sniffing and analyzing attack packets. We develop a script that use *scapy v 2.4.0* to analyze attack packets**.** Scapy decodes traffic packets and matches request with replies. When an attack is detected, attack packets are captured and decoded using script. Table II shows some of the packet features extracted.

Table II: Attack Network Packet Features

S/N	Feature Name	Definition
1	Land	'1' if 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 data portion of attack packet
4	Ip flags	How packet should be routed or processed by higher layer
5	TCP Flags	Defines type of packet sent by attack node
6	Urgent (urg)	Indicates priority of handling packets by router
7	Time to Live	Time left for packet to be discarded
8	Checksum	Error checking in packet header
9	Wrong Fragment	'1' if checksum is 'incorrect'; otherwise '0'

Transaction is prepared based on attack signatures in Fig. 4 or attack features in Tables II and III. The transaction is signed with owner's private key and submits for verification. For additional verification process introduced by our architecture, owner submits other verification information. Examples of verification information are IP address, MAC address, and transaction account. Fig. 5 shows submitted transaction.

Class	**(FEATURES)**priv_key	**(SIGNATURE)p**riv_key	**Verification Information**

Fig. 5. Submitted Transaction

- *Class*: This states the type of IDS. It is either signature based, or anomaly based.
- ***(FEATURES)****priv_key*: This is signed features. This field is updated if class states "anomaly based" otherwise it is skipped.
- ***(SIGNATURE)p****riv_key*: This is signed signature. This field is updated if class states "signature based" otherwise it is skipped.
- *Verification information*: Additional security information

B. Storage

Storage stage is divided into three main steps:

- Verification of transaction and owner.
- Standard format creation
- Transaction validation (i.e. joining transaction block to blockchain).

Agreed upon transaction format (Fig. 5), verification information of all authorized nodes, conversion and format creation scripts (Algorithm 2 and 3) are written as smart contract and mine into the blockchain network. Network feature extraction scripts are run by all authorized nodes. Smart contract executes the following functions:

1. *Transaction and owner's verification:* These are handled by smart contract to ensure that no malicious intruder submits transaction. Algorithm 1 describes how smart contract handles verification of both transaction and owner. Smart contract verifies the privilege of owner to submit transaction and

consistency of submitted transaction with agreed format. For this to be successful and push the transaction to format creation step, the transaction must agree with the format (Fig. 5), private key must be verified using sender's public key and verification information must be in their respective sets. If any of these conditions fail, smart contract returns fail, and transaction is dropped.

Algorithm 1: Verification

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

```
If (Transaction agrees with Format) 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
```

2. *Standard format creation*: This is one of the novelties in our approach. Algorithm 2 describes how smart contract converts a transaction to a standard format. Smart contract checks the "class" field in Fig. 5. If "anomaly based", smart contract checks for missing columns in feature subfields. If there are no missing subfield columns, transaction is arranged in agreed standard format and pushed to validation step, otherwise, it returns fail and transaction is dropped. On the other hand, if "class" field indicates "signature based" signature is converted to standard format (Algorithm 3) and pushed for validation. Fig. 6 shows the standard format of transaction submitted for validation.

Algorithm 2: Standard Format Creation

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

```
if class is "signature based":
    Push transaction to conversion script
else:
    if no missing subfields in features field:
        Rearrange and push transaction for validation
    else:
        Return error
        Drop transaction
    endif
endif
end procedure
```

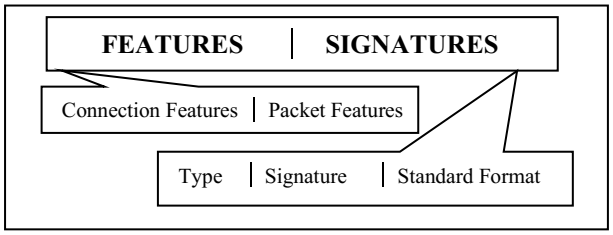

Fig. 6. Standard Format of mined Transaction

Algorithm 3: Signature Conversion Script

Procedure: Converts IDS signature to Standard Format based on common fields
Inputs: Variables of retrieved Signature (SIG_{var}) and values of retrieved Signature (SIG_{value})
Outputs: Variable of standard format (SF_{var})
Mandatory variable: (Action, protocol, source IP, source port, destination IP, destination port, message, sid, rev)

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

Algorithm 3 describes how an attack signature is converted to standard format. Retrieved signature is 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 the different ways of writing rules, we ignore any other standard format variables without equivalent values from retrieving signature.

3. *Transaction Validation*: This step is handled by blockchain consensus protocol. The pending transaction is converted to standard format, then built to a block by authorized node. 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 block input through a mathematical function, which produces an output of a fixed length. The function used to generate the target hash 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 blockchain platform. The characteristics of proof-of-work is that it is computationally difficult to compute and easy to verify. The process of guessing the hash starts in block header. The hash contains block version number, a timestamp, the hash used in previous block, the hash of Merkle Root, the nonce, and the target hash. Successfully mining a block requires an authorized node to keep guessing the nonce that produced the target hash. Nonce is a random string of numbers which keeps changing until target hash is produced. The right nonce is broadcasted 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 set forth in target, then the block is added to the blockchain. It is impossible to mutate /erase the block (i.e. the stored features or signature can neither be manipulated nor deleted).

C. Distribution

After new block has been chained to the blockchain, transaction address is issued to owner (sender). Steps involved in secure distribution of mined transaction are summarized in the following steps:

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

2. *Signatures/Features 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. This information is used to obtain stored transaction (attack signature or features). Nodes extract signature or features from retrieved transaction and use in their intrusion detection system.

IV. RESULTS

The proposed architecture is implemented on Ethereum blockchain platform. We use Solidity *v 0.5.4* implementation of Ethereum for smart contract and *geth v 1.4.18* for Ethereum. For the proof-of-concept, blockchain network is set up in a laboratory with five blockchain nodes, one MYSQL database and one attack nodes as shown in Fig. 2. We implement four authorized nodes (to ensure consensus of miners during validation stage) and one unauthorized (public) node. To make a node authorized, its verification information is included in smart contract. Table III shows the configuration of all nodes.

Table III: Node configurations

Node	OS	RAM	Processor
Authorized node 1	Desktop, Ubuntu 16.04	4GB	2.2GHz
Authorized node 2	Laptop, Ubuntu 18.04	16GB	2.81GHz
Authorized node 3	Desktop, Ubuntu 18.04	8GB	2.44GHz
Authorized node 4	Laptop, Ubuntu 18.04	4GB	2.44GHz
Unauthorized node	Laptop, Ubuntu 18.04	4GB	2.40GHz
Attack node	Laptop, Ubuntu 16.04	4GB	2.20GHz
Database node	Desktop, windows 10	4GB	i5 @2.44GHz

We install 3 signature-based IDS randomly on all blockchain nodes. *Snort v2.9.7* is installed on authorized nodes 1, 4 and unauthorized node, *Bro v 2.6.1* is installed on authorized nodes 2, 3 and unauthorized node and *Suricata v 4.1.3* is installed on authorized node 4. Denial of Service (DoS) attack rule (shown below) is written at local rule file of authorized node 2 snort IDS and snort is started in monitoring mode.

Attack rule: `alert tcp ! $ any any -> $HOME_NET 80 (flags: S; msg:"Possible DoS"; count 70, seconds 10; sid:10001;rev:1;).`

DoS attack is launched at authorized node 2. This node detects this attack, retrieves the above signature and submits it as a transaction to an already set up blockchain network as explained section III. Three other authorized nodes validate this transaction and chain it to the blockchain network. Table IV shows the standard format of mined signature.

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. We run connection and packet analyzing scripts on authorized node 2 in addition to an anomaly-based IDS called Dendritic Cell Algorithm (DCA) [21]. DoS attack was launched at authorized node 2. This is submitted to the blockchain network as discussed in previous section. Three other authorized nodes validate this transaction and attach it to the blockchain network. Furthermore, other forms of attack such as port scanning and Land attacks were launched at authorized node 2. Each attack was repeated 20 different times. Table V shows sample values of features extracted for each attack in one attack launch. The following assumptions are made:

1. *None of the authorized nodes is compromised i.e all features or signatures submitted are good.*
2. *We implement for moderate network traffic and all authorized node is assumed to have similar network traffic.*

Table IV: Standard format of retrieved signature

Standard Format Variable	Signature Values
Action	Alert
Protocol	tcp
Source IP	Any
Source port	Any
Destination IP	Home_net
Destination port	80
Flags	S
Message	Possible Dos
flow	------
Packets/sec	70
Time (seconds)	10
sid	10001
rev	1

Table V: Extracted Features for DoS, Port scanning and Land Attacks

S/N	Features	DoS	Port Scanning	Land
1	No of connections	6594	8	11
2	Source bytes(kbytes)	1147008	708	846
3	Source frames	6592	10	9
4	Source throughput(kbps)	1698.4	216.1	917.96
5	Source frame/second	9995.4	3125.0	9999.99
6	Destination bytes(kbytes)	355968	364	0
7	Destination frames	6592	6	0
8	Destination throughput(kbps)	527.1	111.08	0
9	Destination frame/second	9995.4	1875.0	0
10	Duration(seconds)	0.65	0.32	0.09
11	Source diff. host rate	0	0.16	0.11
12	Source same host rate	1	0.84	0.89
13	Source diff. service rate	0.99	1	0.11
14	Source same service rate	0.01	0	0.89
15	Source diff. host count	1	1	1
16	Source count	6583	6	1
17	Destination diff. host count	1	1	1
18	Destination service count	1	3	1
19	Source IP	192.168.0.144	192.168.0.144	192.168.0.161
20	Source port	8131	48314	80
21	Destination IP	192.168.0.161	192.168.0.161	192.168.0.161
22	Destination port	21	22	80
23	Protocol	TCP	TCP	TCP
24	Type of service	0	0	0
25	Time to live	64	64	64
26	TCP flags	SYN	SYN	SYN
27	IP flags	RES	DF	RES
28	Urgent	0	0	0
29	Fragment	0	0	0
30	Land	0	0	1
31	Checksum	Correct	Correct	Correct
32	Wrong fragment	0	0	0

To evaluate performance of our system, we examine its security against unauthorized transaction submission, then evaluates its response time. The following data are collected for each transaction.

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

A. *Unauthorized transaction:* The architecture is tested against malicious transaction injection. An unauthorized node prepares a transaction and submit to the blockchain. Authorized nodes work to validate this transaction. It is observed that transaction address is not issued to the sender. This implies that transaction is not validated (i.e.transaction is not chained to the blockchain). We manually generate transaction address and ABI, then use them to query the blockchain. We observe that no transaction is returned from the blockchain. This is because no transaction with such address is mined to the blockchain network.

B. *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 includes verification time, mining time and time taken for nodes to request transaction address and retrieve mined features or signatures. Fig.7 shows response time of each node for every transaction. It is observed that latency of 3rd transaction is smallest for all nodes while 10^{th} and 18^{th} transaction has highest latencies for all nodes. This is because latency is mostly affected by validation time which is an iterative process in this case. Fig. 8 shows the average response time of each node. For each node, average response time is addition of response times for all transaction divided by number of transactions. The difference in average response time depends mainly on the computing power of the nodes. It can be seen that average response time for all nodes is less than 1.6 seconds which is considerably small.

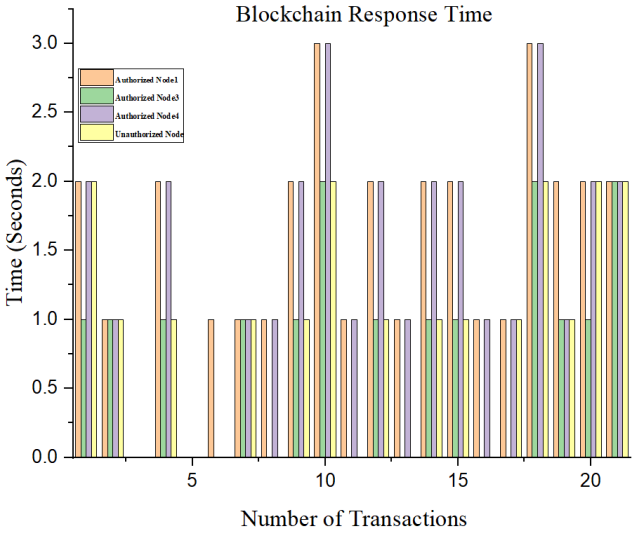

Fig.7. Blockchain response time

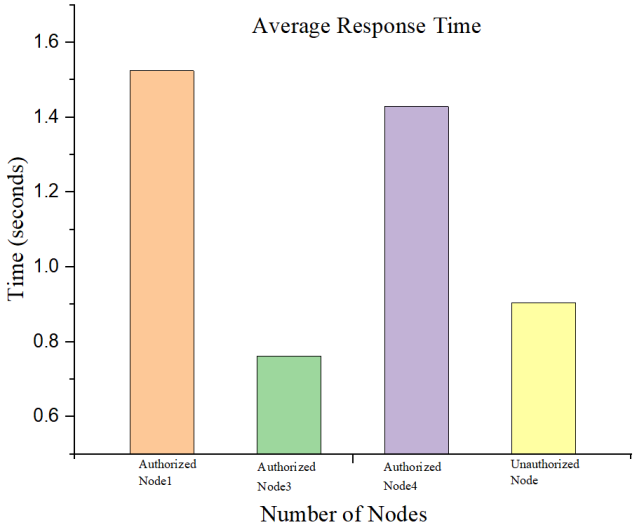

Fig. 8. Average response time of each node

V. Conclusion

In this paper, we propose an architecture that detects and prevents malicious features/signatures injection into shared features or signatures in cooperative intrusion detection. The proposed solution leverages blockchain's distributive technology, tamper-proof ability and data immutability to solve the data security and consistency problems facing cooperative intrusion detection. Focusing on extraction, storage and distribution stages of cooperative intrusion detection, the solution adds extra validation step that makes it impossible for unauthorized nodes to attach blocks of transaction to the blockchain network, hence detects and prevents fake data injection. The architecture also presents a standard format for mined cyber-attack features and signatures which can be easily read and understood by nodes running different IDSs. Apart from this, we implement how the architecture grant permissionless access to public nodes to securely join and retrieve attack features or signatures in real time. Performance evaluation of the system with respect to its response time and resistance to the features/signatures injection is tested. The result showed that the architecture detects and prevents fake data injection, manipulation or deletion and at the same time distribute attack features or signatures with low response time. In future we wish to expand our work to accommodate the following :

1. *Detect compromised authorized nodes:* The only way fake data can be injected into the network is when authorized node (Miner) is compromised (i.e. authorized node sends illegitimate transactions). We intend to implement how to detect when an authorized node is compromised. We will implement different ways an authorized node can be compromised and how our architecture detects such compromised transactions.
2. *Optimization of performance*: If the response time can be reduced more, the architecture can be applied in other areas where real-time data processing is utilized e.g. autonomous vehicle communication. We plan to implement the architecture using different consensus protocols with the view of reducing the response time.
3. *Diverse locations with different network traffic*: One of the features of the proposed system is that it allows nodes in different locations and experiencing different network traffic to exchange data. We plan to implement when blockchain nodes are located in diverse regions and are experiencing different network traffic.
4. *Scalability of the blockchain network:* one of the major characteristics of the architecture is that it should be robust to network growth. We plan to implement and observe the behavior (response time) of the system as more nodes join the blockchain network.

References

[1] 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.

[2] 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.

[3] 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.

[4] O. Ajayi, M. Cherian and T. Saadawi," Secured Cyber-Attack Signatures Distribution using Blockchain Technology." 17th IEEE International Conference on Embedded and Ubiquitous Computing (IEEE EUC 2019), in press.

[5] 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

[6] 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

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

[8] 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).

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

[10] 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

[11] 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

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

[13] 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.

[14] 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)

[15] S.R. Snapp, J. Brentano, G.V. 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.

[16] 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.

[17] 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

[18] 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

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

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

[21] 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.

[22] 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.

[23] 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

[24] 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)

[25] 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

[26] 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

[27]] C. Erway, A. Kupc¸ ¨ u, C. Papamanthou, and R. Tamassia, "Dynamic provable ¨ data possession," in Proceedings of the 16th ACM conference on Computer and communications security. Acm, 2009, pp. 213–222.

[28] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in Proceedings of the 14th ACM conference on Computer and communications security. Acm, 2007, pp. 598–609.

[29] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in Proceedings of the 4th international conference on Security and privacy in communication netowrks. ACM, 2008, p. 9.