SIPchain: SIP Defense Cluster with Blockchain

Aldo Febro*, Hannan Xiao†, and Joseph Spring‡
School of Computer Science
University of Hertfordshire, Hatfield, UK AL10 9AB
Email: *a.febro@herts.ac.uk, †h.xiao@herts.ac.uk, ‡j.spring@herts.ac.uk

***Abstract*—The Session Initiation Protocol (SIP) is an application-layer control protocol for creating, modifying, and terminating Voice/Video over IP sessions. While deployed globally to facilitate multimedia communications, SIP is subject to various attacks. The defense against SIP attacks, however, often lack expertise due to the limited resources within the organization. When there is a large footprint of SIP systems, scaling and keeping up SIP defense becomes crucial in safeguarding these systems. This paper proposes SIPchain, a distributed SIP defense cluster system that leverages Blockchain technology as a distributed, highly-available, and permanent ledger of Indicator of Compromise (IOC). Each node in this cluster is a sensor and shares attack intelligence with other nodes via Blockchain. Each node reads information from the Blockchain and implements the appropriate firewall rule based on this information. This approach scales the defense because each node can leverage the actionable intelligence provided by other nodes and does not have to perform detection on their own. Experiments have been performed using a cluster of three SIP nodes in three different countries (US, UK, and Singapore) and the Ethereum Blockchain network. The result shows that when a node detected an attack, it produced and stored the IOC information at the Ethererum. Fellow SIP nodes retrieved this information, implemented firewall rule based on this information, and were proactively prepared when the same attack was launched against them. This SIPchain approach scales the SIP defense effort by utilizing Blockchain technology to secure the ever-growing footprint of SIP systems within the organization.**

***Index Terms*—Blockchain, Ethereum, IOC, SIP, Asterisk, Fail2ban**

I. Introduction

The Session Initiation Protocol (SIP) [1] has gained wide acceptance and become the de facto signaling protocol for Voice over IP (VoIP) deployment worldwide. With its global footprint, it also has gained the attention of attackers who intend to exploit SIP. Communications Fraud Control Association (CFCA) reported $29.2 billion losses for 2017 with the leading causes of fraud being Private Branch Exchange (PBX) hacking and toll fraud [2]. Adequately securing PBX is challenging for most organizations. Traditional intrusion detection systems do not sufficiently protect PBX because these mechanisms are meant to detect attacks against popular services like web, email, ssh, etc. Even with the use edge gateway or Session Border Controller (SBC), the PBX is still susceptible to brute-force password attack on SIP accounts with weak passwords. To properly secure the PBX and SBC, it requires human expertise to fine-tune the rules and analyze the logs for anomalies [3]. These resources are limited within the organization; therefore, it is hard and costly to provide comprehensive coverage for all PBX within the organization.

A. Motivation and use case

The motivation for this paper is to investigate an alternative option to the traditional approach of securing the SIP-based system and to scaling up SIP security with limited resources. With Blockchain [4] technology and its unique properties, it opens up new possibilities for SIP security that were not available previously.

This motivation is relevant because the pace of SIP deployment is not slowing down, and internal resources are stretched too thin to provide adequate PBX protection. With the rapid pace of businesses that are going digital [5], more communication will take place over IP infrastructure that drives more of SIP deployment. However, the organization did not deploy the human resources required to maintain these new infrastructures at the same rate [6]. This condition is not sustainable in the long run, and a new approach is necessary to be able to find a scalable SIP defense solution for the increasing global footprint.

The use case is to scale SIP protection for a consortium of independent organizations [7] (e.g., a consortium of banks, universities, hospitals, etc.). Each organization autonomously own and maintain their PBX, but yet, they share similar needs and requirements when it comes to protecting their SIP infrastructure. While each organization has its limited resources, creating synergy between these organizations would help each organization to scale with the increased demand.

This consortium creates synergy by arranging these independently owned and operated PBXs into a cluster for attack detection and prevention purposes. Each PBX node can participate and contribute by notifying other members of the attacks that they observed. As such, the other members can leverage and take action based on this intelligence to build their protection. As a result, the members of this consortium are proactively prepared to defend against the same attacks.

B. Background

The proposed method is using several preexisting components as the building blocks that we are going to introduce in this section. These building blocks are arranged to work together to achieve the objective described in the previous section. These preexisting components consist of SIP Proxy server (Asterisk) [8], Firewall (iptables) [9], Intrusion Prevention System (fail2ban) [10], and elements from the

Blockchain ecosystem (Ethereum network [11], smart contract [12], Truffle framework [13], Web3.js library [14], and Infura online service [15]).

- Asterisk is a widely deployed SIP Proxy server. Asterisk serves as the targeted victim for a password brute-force attack.
- Iptables is a rule-based firewall that is typically preinstalled on Linux distribution. Iptables provides packet-filtering that makes an accept or drop decision based on the rules that are grouped under a chain.
- Fail2ban is a widely deployed intrusion prevention system (IPS) that scans Asterisk log file for signs of attack e.g., failed login attempts.
- Ethereum is a public Blockchain network. Ethereum has a production network called Mainnet and several test networks e.g., Ropsten, Kovan, Rinkeby. For our experiment, we used Kovan [16].
- Smart Contract is a scripting function provided by Ethereum, which allows the owner to add logic for processing transactions. A smart contract is written in a programming language called
- Solidity, an object-oriented and statically typed language. Solidity has a compiler that compiles the source code. The compiler produces an Application Binary Interface (ABI) and a bytecode. This bytecode is then deployed to the Ethereum network.
- Truffle Framework is a framework that provides tools and utilities for smart contract development.
- Web3.js Library is a javascript library that allows for connectivity with local or remote ethereum node over HTTP or IPC session.
- Infura is an online service that exposes API for Ethereum operations. This API greatly simplifies operation with Ethereum. Without this service, we would have to build a full-blown Ethereum node and to join this node to the Ethereum network.

How these components contribute to the overall solution is described in section II.

C. Related Works & Contribution

Researchers have been investigating into the SIP PBX hacking activities over the Internet. McInnes [17] experimented over ten days and analyzed about 19 million SIP messages. From this study, the author acknowledged that PBX hacking is indeed a fast-growing problem.

Ford et al. [18] uses fail2ban to produce intrusion data and store it in a centralized database. Romanets, Sachenko, Dubchak [19] also use fail2ban to create data about SIP attacks and then use fuzzy logic to build firewall rules. This paper differs that fail2ban is used by a cluster of synchronized SIP servers instead of on a stand-alone system.

Blockchain technology has been used in security-related literature, for example, securing end-to-end Voice over Long-Term Evolution (VoLTE) sessions and peer-to-peer communication [20] [21], as a control framework for overload within the trust domain [22], as well as in security management schemes [23]. Our paper differs from these approaches in our specific utilization of Blockchain in storing globally synchronized SIP threats.

The unique contribution of this paper is it proposes a novel scalable framework to securing distributed, multiple, and autonomous SIP systems by leveraging Blockchain technology. To the best of our knowledge, Blockchain and SIP have not been explored previously for this purpose.

The rest of the paper is organized as follows: the proposed SIPchain solution is outlined in Section II, followed by experiment in Section III. Section IV, we present our evaluation and results. Discussion is captured in section V, and finally, concluding remarks are drawn in Section VI.

II. The Proposed Solution

A. SIPchain Framework

In this paper, we propose SIPchain, a scalable and distributed SIP defense framework. This framework consists of a front-end and back-end layer that work together as a system. Figure 1 illustrates the SIPchain framework.

1) Front-end layer: The front-end layer consists of multiple PBX nodes (PBX1, PBX2, and PBX3) that collectively make up a cluster. The Administrator (a staff who is responsible to maintain the SIP security in the organization) controls cluster membership by adding or removing nodes as appropriate, which makes this cluster arrangement scalable and distributed. The front-end layer interacts with the back-end layer, the legitimate users, and the attackers.

When dealing with the attackers, the front-end layer performs these functions: attack detection, installing firewall rules, and writing the attacker's IP address to the back-end layer. Each front-end node uses a few components to achieve these functions. The components are SIP Proxy, IPS, Firewall, and Blockchain client scripts (`addIp.js`, `getAll.js`, and `delIp.js`) as listed in Table I. How each component contributes to the overall solution is described next.

TABLE I
SIPchain components

Front-end layer components	Back-end layer components
Firewall (Iptables)	Blockchain
SIP Proxy (Asterisk)	Smart contract
IPS (Fail2ban)	Blockchain API
Blockchain client scripts (addIp.js, delIp.js, getAll.js)	

When the SIP Proxy is under attack, it creates an entry in the log file about what happened. Depending on the attack and the SIP Proxy, the exact log entry would be different for each attack. The Administrator configures the IPS to scan the log file and recognize certain entries as evidence of attacks that took place. When such entries were found, the Administrator also configures the threshold for the IPS to automatically install new firewall rules to prevent the same kind of attack against the SIP Proxy. As the last step, the IPS launches the

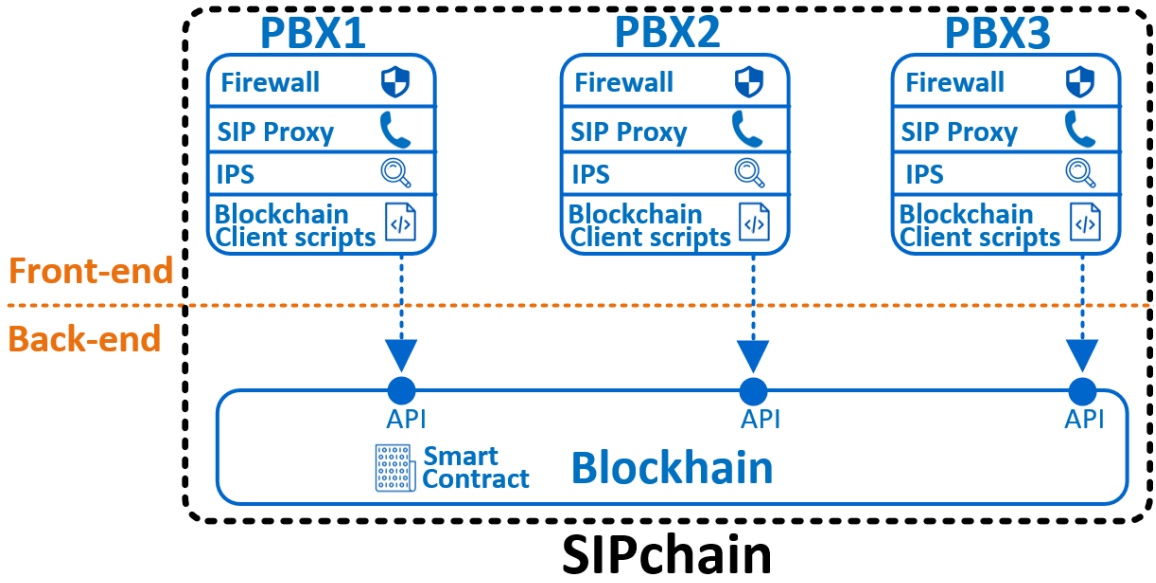

Fig. 1. SIPchain: Distributed SIP Defense Framework with Blockchain

Blockchain client script (`addIp.js`) to add the attacker's IP address to the Blockchain so that other nodes are informed.

When the SIP Proxy is not under attack, `getAll.js` script on each node connects to the Blockchain every minute to see if there is a piece of new information about an attacker. If yes, it will create a new firewall rule so that it can be ready to block attack coming from that particular IP address.

Each rule has a time out period to avoid self-inflicted denial-of-service (DoS) and keep the list of IP addresses manageable. The originating node keeps track of the timing, and when the firewall rule has expired, the IPS removed the rule from the firewall and launched `delIp.js` to remove that particular IP address from the Blockchain.

2) Back-end layer: The back-end layer consists of Blockchain, smart contract, and Blockchain API. It serves as a centralized database for all the member nodes in the cluster. This cluster is a private group that is maintained by the Administrator. The back-end layer interacts exclusively with Blockchain client scripts that were installed on each member node.

Instead of using legacy database technology as the back-end, we used Blockchain for this purpose due to its inherent features and capabilities which are appropriate for the tasks. Blockchain, in this case, serves as a globally synchronized, distributed, and permanent ledger of Indicator of Compromise (IOC) information. The IOC, for this purpose, is a dynamic list of the attacker IP addresses. This list is dynamically added by member nodes (as they are under attack) or removed (as the attack has stopped).

The smart contract provides programmable logic to access the data in the Blockchain and to create Blockchain transaction. It defines the data structure and functions required for managing this dynamic IOC. For example, it includes functions such as add, delete, and retrieve IP addresses. The smart contract is accessible by the Blockchain client scripts using its unique address (a 40-characters hex string).

Blockchain API provides ease of access for the Blockchain client scripts to access the smart contract. Without this API, we would have to own and operate a Blockchain node that is a member of the production Blockchain network. This requirement would make it impractical for end-user environments. In contrast, with the use of API, the Blockchain client script can access the smart contract by calling a REST API endpoint over the Internet. In this case, the requirement is greatly reduced.

B. SIPchain Security Analysis

SIPchain maintains data integrity via a combination of manual administrative processes and authentication method. The Administrator has full control over the membership of this closed-group and private cluster. Membership has to be manually added or removed to ensure that the data is sourced from trustworthy sources. Another manual administrative step required is to install the Blockchain client scripts on each of the participating nodes.

The API provider and smart contract provide the authentication methods. The API provider provides several mechanisms for maintaining security. For example, the URL endpoint includes a dynamically generated portion (i.e., 32-characters hexadecimal string), API secret, list of white list IP address, etc. The smart contract itself could also be programmed to restrict connection only from certain users that have certain characteristics. The users also need to know the smart contract address, which is a 40-characters hexadecimal string. The combination of manual and authentication methods reduce the chance of unauthorized users or attackers to infiltrate and spread false information.

The underlying assumption here is, it is a closed-group that makes use of public Blockchain network. Manual adminis-

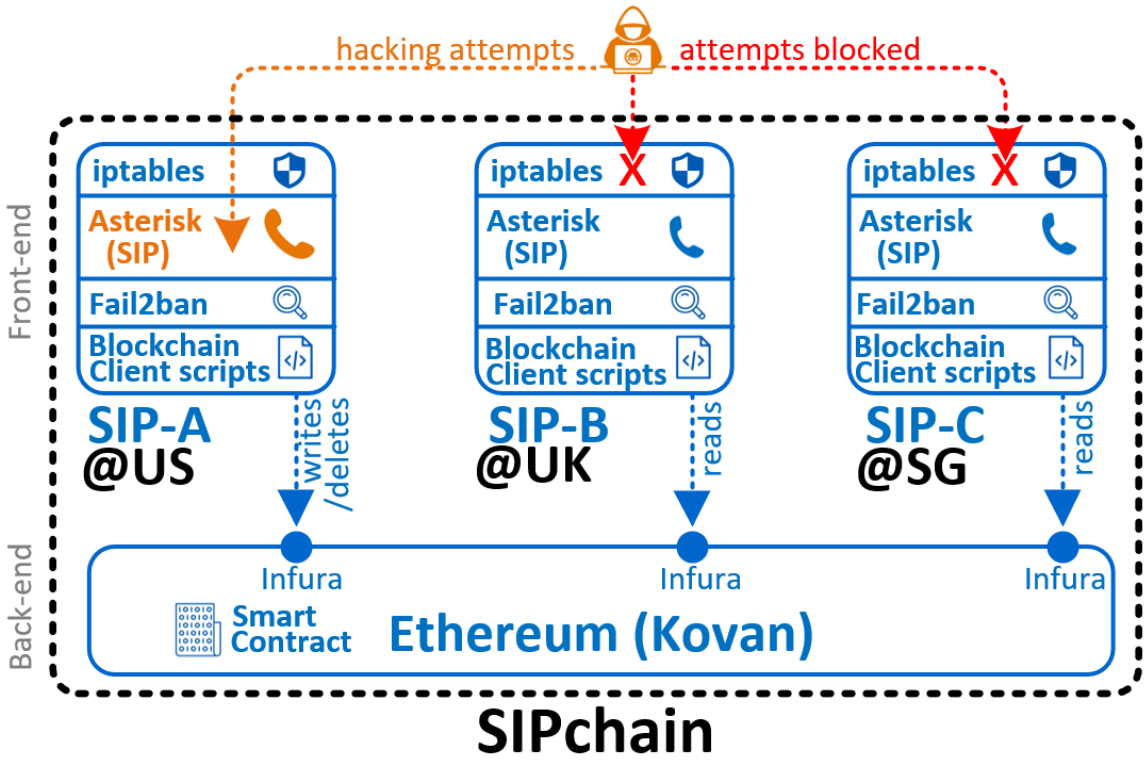

Fig. 2. SIPchain Experiment

trative tasks are required to access the smart contract that is hosted on the public Blockchain network. These tasks include controlling the membership, installation of client scripts and knowing the correct information and credentials to use.

C. Attack Detection

The kind of attack that can be detected is largely determined by the SIP Proxy and IPS being used. Conceptually, if the IPS could detect an attack, it could launch preventative steps. The steps are installing new firewall rules and launching a script to disseminating this information to other member nodes via Blockchain. For example, Asterisk SIP Proxy is able to create a log entry for failed authentication which contains keywords `"Wrong Password"`. The IPS tool, fail2ban is able to recognize this keyword and automatically launch the preventative steps. Some attacks that are recognized by Asterisk by default are brute-force attack, registration attack, invalid account, failed challenge-response, etc. For our experiment, only failed registration attempt was tracked

III. Experiments

In this section, we describe the process involved in our experiments based on the framework from the previous section. It starts with the description of front-end and back-end components, followed by brute-force attack simulation, and concluded with the description of the end-to-end workflow. The testbed is depicted in Figure. 2. The test case and result are described in the next section.

A. Front-end components

We build a testbed on Amazon Web Service (AWS) [24] cloud infrastructure to validate our proposed solution. The testbed consists of three virtual machines in three different continents (US, Europe, and Asia) to represent three PBX in a cluster (SIP-A@US, SIP-B@UK, and SIP-C@SG) as depicted in Figure 2. Each virtual machine is built with 1GB of RAM, 1 virtual CPU, and 8GB of hard drive. Within each virtual machine, we used Asterisk as the SIP proxy server, Fail2ban as the intrusion detection tool, Iptables as the firewall, and custom scripts as the Ethereum client.

1) Firewall (Iptables): Iptables is configured to allow inbound connection at port TCP and UDP 5060, which is the default port for SIP. For this experiment, only SIP ports are made accessible by the Internet.

To keep it manageable and avoid a self-inflicted DoS attack, each firewall rule has a 10 minutes expiry timeout. These additions and removal of firewall rules will be handled automatically by fail2ban and Ethereum client scripts.

2) SIP Proxy & Registrar (Asterisk): We configured Asterisk with ten three-digits extensions that are in sequence (101 - 110) with various password strengths. When the attacker found an extension and launched a brute-force attack against these extensions, Asterisk produces a log entry to indicate such an event. This is the default behavior of Asterisk that does not require additional configuration or add-on module.

3) IPS(Fail2ban): Fail2ban is configured to look for log entries that are related to wrong password attempts. Fail2ban

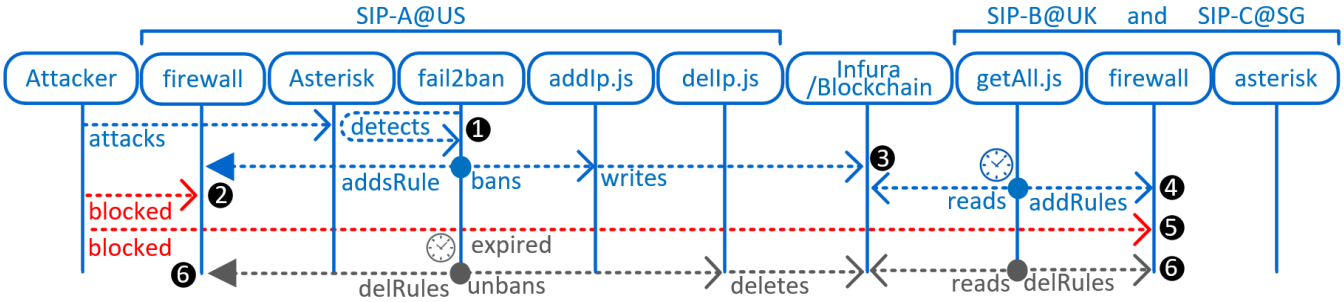

Fig. 3. SIPchain workflow diagram (number indicates test case).

uses three configuration files for its operation, e.g., jail, filter, and action.

The jail file controls the service that is being monitored. For our example, the jail.local file enables monitoring for the asterisk (SIP proxy server). It also specifies the following settings: the port that Asterisk is using (TCP 5060 and TCP 5061), the path to the log file, the number of attempts required to trigger an action, and the action to take.

The filter file controls the pattern that could trigger the action. It consists of a complex regular expression that matches a specific condition. For example, for the wrong password, this pattern would match with a log entry in /var/log/asterisk/messages file:

```
^Registration from '[^']*' failed for
'<HOST>(:\d+)?' - (?:Wrong password
```

The action file specifies what command to launch when a specific action is required, e.g., actionstart, actionstop, actioncheck, actionban, actionunban. For our case, actionban and actionunban is configured to trigger a node.js client script with the attacker IP address passed as the parameter, e.g.,:

```
actionban = node addIp.js "<ip>"
actionunban = node delIp.js "<ip>"
```

4) Blockchain client scripts: There are a few Blockchain client scripts involved: addIp.js, delIp, and getAll.js, which are Nodejs scripts. They use libraries such as Truffle framework and Web3.js library to facilitate communication with the Ethereum network. This script is launched by fail2ban and used the attacker's IP address as an input parameter. It will then take this parameter and write it to the Blockchain via Infura. Ethereum client scripts are considered to work correctly when they can call an Infura API endpoint over the Internet and the newly created transaction can be viewed in the Ethereum network.

B. Back-end components

1) Blockchain (Kovan: Ethereum development network) [16]: Ethereum has several networks that differ on its purpose (test vs. production) and their consensus algorithm (Proof-of-Authority [25]). For our experiment, we used Kovan, a test network that uses a proof-of-authority consensus algorithm that is faster and more efficient in its operation.

2) Smart contract: In this work, smart contracts are used to define a string in a dynamic array to store the IP address of an attacker. We also define various functions required to add, delete, and retrieve the IP address of an attacker.

Access to this smart contract is limited to group members only. Only members that have the right contract address (40-character hexadecimal strings) and credentials from Infura (32-character hexadecimal strings) can interact with the smart contract. In absence of the above, non-authorized users (or hackers) are not be able to create, read, update or delete information that is stored on the smart contract.

3) Blockchain API (Infura): Infura provides an endpoint (URL) that is specific for accessing the Ethereum network (Kovan). This endpoint URL, together with the access credentials are used in Node.js ethereum scripts. This online service provided by Infura alleviates the need for us to build and run an Ethereum node on our own.

C. Brute-force Attack Simulation

We performed experiments with a brute-force attacks against a valid SIP extension on Asterisk PBX by using a SIP client. This use case was selected because it is a common scenario where the hacker gains entry into the system by repeatedly guessing the password of a valid extension to access the account and commit a call fraud [2]. Users tend to use weak passwords for their SIP account because it is hard to enter strong passwords via a telephone keypad.

To simulate the brute-force attack, the SIP client sent multiple `REGISTER` requests to Asterisk PBX with a valid extension but supplied the wrong password which generated a failed login event.

D. Attack Detection

Attack detection was done by using a set of rules defined in fail2ban. Asterisk PBX, by default, created a log entry for every failed login attempt from a SIP client.

When fail2ban scanned the log file, it found these entries and considered these as evidence of attacks that took place. Fail2ban has predefined rules that indicate if there are three or more consecutive entries of a failed login attempts, recognize these as brute-force attacks, and perform the mitigation steps.

The mitigation steps, in this case, is to create a new firewall rule (to drop further packets from this particular attackers IP address) and create a new entry in the Blockchain.

E. End-to-end Workflow

The end-to-end workflow is depicted in Figure 3, where it was initiated by a brute force attack against the Asterisk SIP proxy at SIP-A@US. This attack triggered Asterisk to create a log entry for the failed login attempt due to incorrect passwords.

Fail2ban running on SIP-A@US scanned the Asterisk log file regularly, and through its filter (i.e., complex regular expressions) detected failed login attempts. This finding triggered two actions, i.e., created a new firewall rule (to drop packets coming from the attackers IP address) and launched addIp.js script. The addIp.js script wrote the attacker's IP address to the Blockchain via Infura's URL endpoint.

On the SIP-B@UK and SIP-C@SG node, there was a script that launched every minute to check the smart contract and to retrieve the current list of attacker IP addresses. Based on this information, the script created the appropriate firewall rules for SIP-B and SIP-C to be proactively prepared, should the same attack be directed at them.

As the final step, the same attacker launched the same attack against SIP-B@UK and SIP-C@SG. Since they have added the new firewall rule from the previous step, these packets were dropped.

IV. Evaluation

A. Test Cases

The test cases are laid out to facilitate validation for each phase of the workflow as described above. These six test cases are numbered in Figure 3.

1) Test case 1. Initial attack: The attacker launched a brute-force attack against one of the valid extensions on this SIP proxy (extension 100). This test validates whether Asterisk correctly generates a log entry that can be searched by fail2ban. It is a success when Asterisk logs these attacks, and fail2ban can find those entries in the log file.

2) Test case 2. Add FW rule at SIP-A@US: This test validates the fail2ban configuration and functionality. Fail2ban should be able to detect the attack and create a new firewall rule to drop packets from the attackers IP address. It is a success when the new firewall rule is created at SIP-A@US and starts to block the identified attacks.

3) Test case 3. Creates a new Blockchain transaction: This test validates the addIp.js script which is supposed to create a new transaction so that other nodes can retrieve and be informed about the attackers IP address. It is a success when it can create new transaction on the Blockchain.

4) Test case 4. Reads from Blockchain and adds new FW rule: This test validates whether the getAll.js script on SIP-B or SIP-C can successfully retrieve the information from the Blockchain. It also confirms whether getAll.js can create new firewall rules so that the firewall can be proactively prepared against the same attack. It is a success when a new firewall rule is created on SIP-B@UK and SIP-C@SG.

```
21:27:34,623 fail2ban.filter        [843]: INFO    [asterisk] Found 52.67.198.7 - 2019-07-01 21:27:34
21:27:36,636 fail2ban.filter        [843]: INFO    [asterisk] Found 52.67.198.7 - 2019-07-01 21:27:36
21:27:41,443 fail2ban.filter        [843]: INFO    [asterisk] Found 52.67.198.7 - 2019-07-01 21:27:41
21:27:41,747 fail2ban.actions       [843]: NOTICE  [asterisk] Ban 52.67.198.7
```

Fig. 4. Fail2ban log entries captured the attacks and banned the attacker's IP address.

```
=== SIP-A-US@21:28:01 ===
Chain INPUT (policy ACCEPT 108 packets, 41349 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 f2b-asterisk-tcp  tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
   12  7653 f2b-asterisk-udp  udp  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain f2b-asterisk-tcp (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 REJECT     all  --  *      *       52.67.198.7          0.0.0.0/0
    0     0 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain f2b-asterisk-udp (1 references)
 pkts bytes target     prot opt in     out     source               destination
    6  3576 REJECT     all  --  *      *       52.67.198.7          0.0.0.0/0
    6  4077 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

Fig. 5. New firewall rules were created at SIP-A@US in response to the attack (screenshot is edited for brevity and clarity).

5) Test case 5. Launching attack against SIP-B@UK & SIP-C@SG: This test validates whether SIP-B and SIP-C can successfully defend the same attack that was launched against SIP-A without any manual intervention from the SIP server Administrator. It is a success when the attack can be blocked by the new firewall rule that was created by test case 4.

6) Test case 6. Deletes expired FW rule: This test validates whether fail2ban and getAll.js script could successfully remove the expired firewall rules automatically. It is a success when the old firewall rules are removed.

B. Results

1) Test case 1 result: Figure 4 shows the entries in fail2ban.log file about the attack and the action taken i.e., ban the source IP address. This log entry implies that Asterisk observed three failed brute-force attempts that originated from this particular IP address.

2) Test case 2 result: Figure 5 displays the new firewall rule at SIP-A@US, which blocked packets coming from 52.67.198.7 at UDP/5060/5061. The pkt column shows the number of packets that have been dropped by this rule. This entry confirmed that the rule was effective for blocking packets from the attacker.

3) Test case 3 result: Figure 6 shows the transaction record at https://kovan.eterhscan.io website. This implies that the client script has successfully created a transaction at Kovan and returned a transaction hash `0x4ff44a6be8caecee9408042e0e97c9cd151c5cfdd90e4c87595d4188c1166a51`.

4) Test case 4 result: Figure 7 shows the new firewall rule on SIP-B@UK that was created based on the information retrieved from the Blockchain. The client script at SIP-B@UK (and SIP-C@SG) was able to retrieve the attacker IP address from the Blockchain and added a new firewall rule.

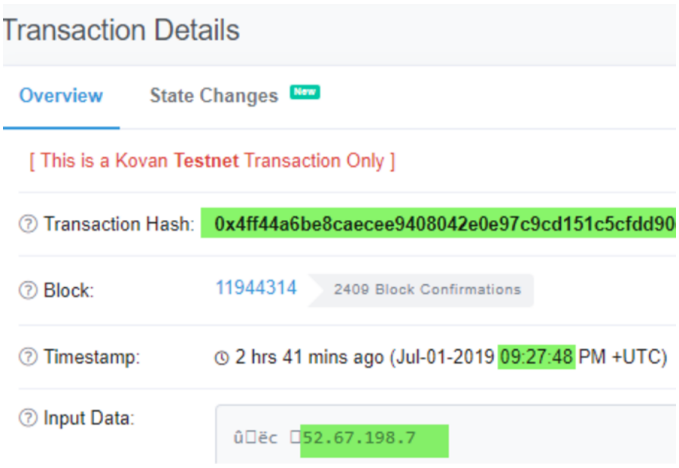

Fig. 6. Transaction record at https://kovan.etherscan.io (screenshot is edited for brevity and clarity).

```
=== SIP-B-UK@21:29:01 ===
Chain INPUT (policy ACCEPT 2 packets, 188 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 f2b-asterisk-tcp  tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 f2b-asterisk-udp  udp  --  *      *       0.0.0.0/0            0.0.0.0/0
Chain f2b-asterisk-tcp (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 REJECT     all  --  *      *       52.67.198.7          0.0.0.0/0
    0     0 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain f2b-asterisk-udp (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 REJECT     all  --  *      *       52.67.198.7          0.0.0.0/0
    0     0 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

Fig. 7. New firewall rule is created at SIP-B@UK (screenshot is edited for brevity and clarity).

5) Test case 5 result: Figure 8 shows the number of packets that were dropped by SIP-B@UK. This attack was launched from the same attacker that attacked SIP-A@US.

6) Test case 6 result: When the firewall rule reached its expiry time (10 minutes), fail2ban on SIP-A@US performed two tasks: removed the firewall rule and launched the delIp.js script (also on SIP-A@US) which deleted the attacker IP address from Blockchain. When the getAll.js script on SIP-B@UK and SIP-C@SG read this information, the script removed the expired firewall rule as well.

```
=== SIP-B-UK@21:30:01 ===
Chain INPUT (policy ACCEPT 2 packets, 188 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 f2b-asterisk-tcp  tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
   43 25757 f2b-asterisk-udp  udp  --  *      *       0.0.0.0/0            0.0.0.0/0
Chain f2b-asterisk-tcp (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 REJECT     all  --  *      *       52.67.198.7          0.0.0.0/0
    0     0 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain f2b-asterisk-udp (1 references)
 pkts bytes target     prot opt in     out     source               destination
   43 25757 REJECT     all  --  *      *       52.67.198.7          0.0.0.0/0
    0     0 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

Fig. 8. New firewall rule at SIP-B@UK started dropping packets from the same attacker that launched attack against SIP-A@US earlier (screenshot is edited for brevity and clarity).

V. Discussion

In this framework, we see a synergy where each node leverages other nodes detection capability. The detection that has been carried out by one node, need not to be carried out again by other nodes.

Whilst Blockchains are showing real potential, the components that make up a solution (e.g., the programming language, compiler, tools, libraries) are also under active development. These components are generally built by teams that work independently from each other. We need to pay close attention to the dependencies of each component that we use in our Blockchain applications because just one of these components may introduce a change that may compromise a previously developed solution to the problem. For example, a function name change was removed from the library that caused unresolved dependency and therefore a compilation error.

Solidity, as a smart contract programming language, is still maturing and has its unique characteristics when compared to other modern languages [26]. It requires a manual workaround to perform typical array operation such as update or delete. It does not have high-level string operation such as concatenation, extraction, etc. This limitation requires the developer to do it manually using the lower-level constructs of the language. In addition, the web3 library and truffle framework also have some restrictions of their own. For example, while solidity supports returning strings as part of the dynamic array, web3 does not. This intricacy between the language and the library impose additional challenges.

It is necessary to balance the requirements against Blockchain characteristics in order to gain the optimum use of Blockchain technology. For example, different operations have a different associated cost, forcing the developer to be mindful with the data structure, storage, algorithm, etc. that contributes to the overall cost. The Ethereum Yellow paper Appendix G publishes the fee schedule [27].

Speed is another essential factor for consideration. Due to its nature, each Blockchain transaction has to be mined before it is appended to a block. This mining operation requires more time than the traditional database insert operation. As such, the Ethereum clients (or the distributed application) need to accommodate this type of asynchronous operation to ensure that the interaction (between client and Ethereum network) is timed appropriately. In our experiment, it took about 7 seconds to create a new insert transaction in the Blockchain (from 21:27:41 to 21:27:48). It took about 3 seconds to retrieve the data. The performance difference between writing and reading is expected because the level of operation involved. With the writing operation, the Blockchain network has to perform computational operations to create and append a new block.

For the use case of a SIP defense cluster, the use of Blockchain technology delivers back-end capabilities that were not previously available. Blockchain offer desirable features such as data integrity, high-availability, no single point of failure, and global consensus. Before Blockchain, it would

have posed operational, financial, and trust challenges to maintain a publicly verifiable ledger that offers these features.

Future work will consider methods for improving the information dissemination time. We are currently exploring a publish-subscribe method where each node can subscribe to events that occurred with the smart contract. While not eliminating the time gap, this method would shorten the time it takes to act on the intelligence received from the Blockchain.

VI. Conclusion

This paper presented SIPchain, a scalable method of securing multiple autonomous SIP systems by leveraging the Blockchain technology. It scales the defense by automatically capturing an attacker's IP address and disseminating it to other nodes via the Blockchain. The experiment shows that other nodes in the cluster were able to retrieve the attacker IP address from the Blockchain and then updated their own firewall rule. This action proactively prepares them should they get attacked by the same attacker. Similarly, when the rule has expired, it was automatically deleted from the Blockchain, which then propagated to other nodes. This dynamic and automated approach helps organizations to scale their limited human resources to secure their ever-growing SIP footprint.

Whilst Blockchain is showing real potential in this use case, some limitations require careful considerations so that we can leverage its advantages while offsetting its limitations.

References

[1] J. Rosenberg, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "RFC 3261: Session Initiation Protocol (SIP)," *Internet Engineeging Task Force*, vol. 1, no. 11, pp. 1829–1841, 2002.

[2] J. Osenbaugh, "Telecom fraud on the rise: What enterprises need to know." https://www.nojitter.com/security/telecom-fraud-rise-what-enterprises-need-know, February 2019. Web. Accessed 26 Jun. 2019.

[3] S. Mishra, R. K. Raj, C. J. Romanowski, J. Schneider, and A. Critelli, "On building cybersecurity expertise in critical infrastructure protection," in *2015 IEEE International Symposium on Technologies for Homeland Security (HST)*, pp. 1–6, April 2015.

[4] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[5] A. Aagaard, M. Presser, M. Beliatis, H. Mansour, and S. Nagy, "A tool for internet of things digital business model innovation," in *2018 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, Dec 2018.

[6] W. Crumpler and J. A. Lewis, "The cybersecurity workforce gap," *Center for Strategic and International Studies, Washington, DC.[Online]. Available: https://www. csis. org/analysis/cybersecurityworkforce-gap*, 2019.

[7] A. Irrera, "Wef leads creation of fintech cyber security consortium." https://www.reuters.com/article/us-cyber-fintech/world-economic-forum-leads-creation-of-fintech-cyber-security-consortium-idUSKCN1GI17G, March 2018. Web. Accessed 26 Jun. 2019.

[8] "Open source communications software — asterisk official site." https://asterisk.org/, September 2019. Web. Accessed 9 Sep. 2019.

[9] "Man page of iptables." http://ipset.netfilter.org/iptables.man.html. Web. Accessed 28 Aug. 2019.

[10] M. Javed and V. Paxson, "Detecting stealthy, distributed ssh brute-forcing," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 85–96, ACM, 2013.

[11] "Ethereum." https://www.ethereum.org. Web. Accessed 28 Aug. 2019.

[12] "Introduction to smart contracts." https://github.com/ethereum/solidity/blob/v0.5.11/docs/introduction-to-smart-contracts.rst, June 2019. Web. Accessed 28 Aug. 2019.

[13] "Truffle — overview — documentation — truffle suite." https://www.trufflesuite.com/docs/truffle/overview. Web. Accessed 28 Aug. 2019.

[14] "Github - ethereum/web3.js: Ethereum javascript api." https://https://github.com/ethereum/. Web. Accessed 28 Aug. 2019.

[15] "Ethereum api — ipfs api amp; gateway — eth nodes as a service — infura." https:/infura.io. Web. Accessed 28 Aug. 2019.

[16] "Kovan testnet." https://kovan-testnet.github.io/website/. Web. Accessed 28 Aug. 2019.

[17] N. McInnes, G. Wills, and E. Zaluska, "Analysis of threats on a voip based pbx honeypot," 03 2019.

[18] M. Ford, C. Mallery, F. Palmasani, M. Rabb, R. Turner, L. Soles, and D. Snider, "A process to transfer Fail2ban data to an adaptive enterprise intrusion detection and prevention system," in *Conference Proceedings - IEEE SOUTHEASTCON*, vol. 2016-July, Institute of Electrical and Electronics Engineers Inc., jul 2016.

[19] I. Romanets, A. Sachenko, and L. Dubchak, "Method of protection against traffic termination in voip," in *2018 10th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pp. 1–5, June 2018.

[20] E. Kfoury and D. Khoury, "Securing natted iot devices using ethereum blockchain and distributed turn servers," in *2018 10th International Conference on Advanced Infocomm Technology (ICAIT)*, pp. 115–121, Aug 2018.

[21] E. F. Kfoury and D. J. Khoury, "Secure end-to-end volte based on ethereum blockchain," in *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*, pp. 1–5, July 2018.

[22] K. Inamdar, G. Salgueiro, *et al.*, "Using blockchain to simplify session initiation protocol overload control," 2018.

[23] Y. Jung, M. Peradilla, and R. Agulto, "Packet key-based end-to-end security management on a blockchain control plane," *Sensors*, vol. 19, no. 10, p. 2310, 2019.

[24] "Amazon web services (aws) - cloud computing services." https://https://aws.amazon.com/. Web. Accessed 16 Sep. 2019.

[25] "Proposal kovan testnet." https://kovan-testnet.github.io/ website/proposal/. Web. Accessed 28 Aug. 2019.

[26] M. Wohrer and U. Zdun, "Smart contracts: security patterns in the ethereum ecosystem and solidity," in *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, pp. 2–8, March 2018.

[27] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger byzantium version d664f - 2019-06-13." https://ethereum.github.io/yellowpaper/paper.pdf, june 2019. Web. Accessed 30 Jun. 2019.