

**Kennesaw State University**  
**DigitalCommons@Kennesaw State University**

---

Master of Science in Information Technology  
Theses

Department of Information Technology


---

Summer 7-9-2019

# An Architecture for Blockchain-based Collaborative Signature-based Intrusion Detection System

Daniel Laufenberg

Follow this and additional works at: [https://digitalcommons.kennesaw.edu/msit\\_etd](https://digitalcommons.kennesaw.edu/msit_etd)

 Part of the [Databases and Information Systems Commons](#), [Information Security Commons](#), and the [Systems Architecture Commons](#)

---

## Recommended Citation

Laufenberg, Daniel, "An Architecture for Blockchain-based Collaborative Signature-based Intrusion Detection System" (2019). *Master of Science in Information Technology Theses*. 5.  
[https://digitalcommons.kennesaw.edu/msit\\_etd/5](https://digitalcommons.kennesaw.edu/msit_etd/5)

This Thesis is brought to you for free and open access by the Department of Information Technology at DigitalCommons@Kennesaw State University. It has been accepted for inclusion in Master of Science in Information Technology Theses by an authorized administrator of DigitalCommons@Kennesaw State University. For more information, please contact [digitalcommons@kennesaw.edu](mailto:digitalcommons@kennesaw.edu).

An Architecture for Blockchain-based Collaborative Signature-based Intrusion Detection System

Daniel Laufenberg

A Thesis in the Field of Information Technology  
for the Degree of Master of Science in Information Technology

Kennesaw State University

July 2019



### Thesis/Dissertation Defense Outcome

Name Daniel Laufenberg 000878174  
 Email dlaufenb@students.kennesaw.edu Phone Number \_\_\_\_\_  
 Program MSIT

Title

An Architecture for Blockchain-based Collaborative Signature-based Intrusion  
 Detection System

Thesis/Dissertation Defense: Date

Passed  Failed  Passed With Revisions (attach)

#### Signatures

**Lei Li** Digitally signed by Lei Li Date: 2019.07.20 07:55:36 -04'00' 7/20/2019  
 Thesis Co- \_\_\_\_\_ Date

DocuSigned by: *Hossain Shukur* \_\_\_\_\_ Date

Thesis Co-Chair \_\_\_\_\_ Date

DocuSigned by: *Meng Fan* \_\_\_\_\_ Date

Committee \_\_\_\_\_ Date

\_\_\_\_\_ Date

Committee

Date

---

Committee

---

Date

**Lei Li**

Digitally signed by Lei Li  
Date: 2019.07.20 07:56:03 -04'00'

---

Program Director

---

Date

DocuSigned by:

*Rebecca Rutherford*

Department...

---

Date

---

Graduate

---

Date

## Acknowledgments

I would first like to thank my thesis advisors Dr. Lei Li, Dr. Hossain Shahriar and Dr. Meng Han. Whenever I had a question about the research involved their doors were always open to me and they consistently guided me and this paper in the right direction whenever I needed it. Their validation and passionate participation and input allowed me to glimpse the processes and nuances of complex research.

I would also like to express my profound gratitude to my wife Amber Lee for providing me with unfailing support and continuous encouragement throughout my years of study and the process of researching and writing this thesis. This would not have been possible without her. Thank you.

This work is dedicated to my daughter Allison Laufenberg-Lee who was born eleven days from the day of my thesis defense. I want to thank her for being patient and waiting until then to arrive.

## Abstract

Collaborative intrusion detection system (CIDS), where IDS hosts work with each other and share resources, have been proposed to cope with the increasingly sophisticated cyberattacks. Despite the promising benefits such as expanded signature databases and alert data from multiple sites, trust management and consensus building remain as challenges for a CIDS to work effectively. The blockchain technology with built-in immutability and consensus building capability provides a viable solution to the issues of CIDS. In this paper, we introduce an architecture for a blockchain-enabled signature-based collaborative IDS, discuss the implementation strategy of the proposed architecture and developed a prototype using Hyperledger and Snort. Our preliminary evaluation on a bench mark showed the proposed architecture offers a solution by addressing the issues of trust, data sharing and insider attacks in the network environment of CIDSs. The implications and limitations of this study are also discussed.

Table of Contents

Acknowledgments..... iv

**Chapter I. Introduction** .....8

**Chapter II. Background and Related Work** .....10

    2.1 Consensus Algorithms .....11

    2.2 Blockchain Application .....13

    2.3 Intrusion Detection System (IDS).....13

    2.4 Intersection of Blockchain and IDS .....15

**Chapter III. The Proposed Architecture** .....16

**Chapter IV. Implementation**.....19

    4.1 Prototype Environment .....22

        4.1.2 Hardware .....22

        4.1.3 Operating Systems .....23

        4.1.4 HyperLedger .....23

        4.1.5 Prerequisites .....24

    4.2 Prototype Guide .....26

        4.2.1 Setting up Blockchain .....27

    4.3 YouTube Tutorial Videos .....36

**Chapter V. Conclusions**.....37

Appendix I BBids Prototype Code, using FabCar contract in HyperLedger.....38

**Chaincode files:** .....38

        /lib/Fabcar.js.....38

**Startup Script:**.....54

<b>Contract files:</b> .....	57
enrollAdmin.js .....	57
Invoke.js .....	58
Query.js .....	60
registerUser.js .....	61
References .....	64



# Chapter I.

## Introduction

Interconnected computer networks have been the engine for economic growth and innovation for the past few decades. It has become increasingly important to protect the digital infrastructure of our society against attacks. The intrusion detection system (IDS), has been widely used by individuals, business and organizations for their computer networks protection.

Working with existing firewalls and anti-virus systems, an IDS is a device or software application which monitors network traffic, identifies attacks by building normal network profiles (anomaly-based IDS) or matching the patterns of malicious behavior or violations (signature-based IDS) that protects computer networks against attacks [34]. An IDS can offer real-time, cross-platform, and pre-host protection and is a viable solution to mitigate some malicious attacks [13, 29]. Anomaly-based IDSs are prone to having many false positives [32]. Signature-based IDSs are generally better with the precision rate but can often miss attacks if the signature database is outdated or incomplete [28, 33].

As cyber-attacks are becoming more sophisticated and being launched at a larger scale and across platforms [cite examples], an intrusion detection system would be more effective if it works with other IDSs. For example, IDS hosts can exchange resources such as network traffic, data alerts, signatures and share signature databases. [7, 10, 23, 30, 31]. Such a system is referred as a collaborative intrusion detection system (CIDS).

Despite promising benefits of CIDS, the underlying trust behind sharing of resources remains a major concern. In particular, an attacker host may join in a collaborative IDS system network and provide inaccurate or malicious signatures. Moreover, a host environment may be tampered with to alter the data files that actually store signatures (Snort IDS saves the rules in plain text files, which can be easily altered).

Recently, there has been a spike of interest in the blockchain technology where distributed data structure is shared and replicated among the participants in a peer-to-peer network [1, 2, 3, 4, 5, 8, 12]. The built-in immutability and consensus building make the blockchain technology a viable solution to develop collaborative IDS and overcome trust management and consensus building among IDS [26]. Alexopoulos et al. [23] proposed a general framework for block-based collaborative IDS which is focused on using blockchain for alert sharing and consensus building.

Inspired by the efforts of Alexopoulos et al.'s work, we introduce a blockchain-enabled architecture for a signature-based IDS. In addition to alert exchange, we also propose to use the blockchain technology for signature management such as signature sharing, creation and verification among hosts in a CIDS. We also present the implementation strategies of the architecture. Based on our knowledge, the proposed architecture is the first kind for CIDSs.

The remainder of the paper is organized as follows: Section II introduces related work on Blockchain and intrusion detection systems, Section III discusses an architecture for a collaborative Signature-based IDS based on the blockchain technology. Section IV presents the implementation consideration of the proposed architecture. Finally, Section V concludes the paper.

## **Chapter II.**

### **Background and Related Work**

Blockchain can loosely be defined as a data structure, database, or a growing list of records, called blocks, which are linked using cryptography [27]. There are three types of blockchains: public, private and consortium [27].

A public blockchain such as Bitcoin is an open system [26] where anyone can join and participate in the system. The two advantages of public blockchain are its characteristics of Permission-less and immutability. Having a public blockchain removes the necessity for an access control protocol. Applications can be added to the network without approval, and blockchain becomes the transport layer of these applications [26]. A public blockchain is stored typically on a peer-to-peer network. This allows for the data to be highly unchangeable due to many computers storing the data and agreeing on what is legitimate data and what could possibly be illegitimate.

A private blockchain is a closed system in which the use of the blockchain is controlled. There is limited application of private blockchains as the central control works against the decentralization aspect which is key to the blockchain concept.

In a consortium blockchain there is a mixture of both. Typically, a consortium is public but the number of nodes who can change the data in the blocks is limited. Consortium's are sometimes invite-only for this purpose.

Blockchain tends to fall short when it comes to scalability, depending on the consensus algorithm used. Speed is a big concern as well for any application of a blockchain system.

## 2.1 Consensus Algorithms

As a distributed structure, consensus building is very important for blockchain where nodes of the blockchain construct and support the decision that works best for the rest of them. It's a form of resolution on how to add blocks, data, or do anything to the blockchain. There are many consensus algorithms, however, this paper will only cover those pertinent to the paper: Proof of Work, Proof of Stake, Delegated Proof of Stake, Proof of Authority, Byzantine Fault Tolerance, Proof of Elapsed Time.

*Proof of work.* In a proof of work system, the new blocks in the chain are created by those that have the computational power to solve complex mathematical problems. PoW has some problems with power consumption and inefficiency. This system is used in Bitcoin and would not be ideal for the proposed IDS.

*Proof of stake.* In a proof of stake system, the new blocks are created in a distributed consensus. The next block is chosen by combinations of random selection and wealth range. Ethereum has a proof of stake currently in development called Casper.

*Delegated Proof of Stake.* In a delegated proof of stake, those that have "stake" in the blockchain can vote for others to have control of the chain. It isn't all about who owns

the most cryptocurrency or most stake in the blocks, it is about having democratic votes to mitigate the risks of the original proof of stake.

*Proof of Authority (PoA).* PoA consensus is built further off of Proof of Stake. Instead of voting or allowing someone who was an early adopter of a blockchain to have "stake" in it, the proof of authority puts the onus on those with the reputation to be in control of the chain. These are trusted individuals within the community or network that are well-respected.

*Byzantine Fault Tolerance (BFT).* Named after an old adage of Byzantine General problems, this algorithm has been around for some time. The idea is that two generals were attempting to communicate between enemy lines and can never be 100% sure that their messages are received. BFT at its simplest form is a way to avoid nodes or blocks in the chain doing something that they were not supposed to do. BFT can be found in many popular consensus algorithms in blockchain.

*Proof of Elapsed Time (PoET).* The PoET consensus algorithm that is designed to be a production-grade protocol capable of supporting large network populations. PoET algorithm relies on secure consensus without the power consumption drawbacks of the Proof of Work algorithm. Each person in the network waits a random amount of time, whoever finishes waiting first becomes the leader of the new block in the chain[25].

## 2.2 Blockchain Application

There are many different applications being conceived by researchers in the field of blockchain, such as consensus algorithm research Proof of Majority[2], supply chains, ProductChain, a scalable blockchain framework for supply chains[16], AutoBotCatcher a system proposed to protect the infrastructure of IoT devices[4], TickEth, a proposed system for using blockchain to buy sporting event tickets[22], a package delivery system [14], and a networking trading system [17]. There is also work showing how blockchain can be used with machine learning in [21] where the ledger self-adapts to transaction demands. As we can see there is a body of work done on how blockchain can be used for things other than cryptocurrencies.

## 2.3 Intrusion Detection System (IDS)

There is a wealth of studies on Intrusion Detection System (IDS) because of its impact on cybersecurity. IDSs can be categorized as host-based IDS and network-based IDS. There are pros and cons for each type of IDS. In a host based system the IDS runs on a single host, this allows for the IDS to directly monitor that host and which resources were attacked. Host-based can make it difficult to analyze the intrusion attempts on multiple computers and will be difficult to work in a large network environment. In a

network-based IDS a network sensor is installed on the network interface card and allows for an entire network to be monitored. All of these packets are analyzed, however this can take a lot of time and resources, and can miss packets going to a specific host.

IDSs can also be divided by the detection methods: signature-based intrusion detection systems and anomaly-based intrusion detection systems. A signature-based IDS relies on patterns of malicious behavior or violations to recognize the attacks. Signature-based IDS could ideally identify 100% of the attacks with no false alarms as long as signatures are specified ahead of time. However, each signature, even if it leads to the same attack, has the potential to be unique from any other signatures. This is the most commonly implemented IDS [34,35,36].

The other common type of IDS is an anomaly detection system. This type of IDS focuses on the system's normal behaviors instead of focusing on attack behaviors, as seen with signature-based intrusion detection systems. To implement this type of IDS, the approach is to use two phases. The first phase is the training phase where the systems behavior is observed in the absence of any type of attack. Normal behavior for the system is identified into a profile. After this, the second phase or detection phase, begins. In this phase, the stored profile is compared to the way the system is currently behaving and deviations from the profiles are considered potential attacks on the system. This can lead to several false positives [37, 38, 39].

There has been a growing trend of research towards CIDS due to the speed and efficiency of peer-to-peer networks[31][32]. As the Internet becomes faster the shareability of an application becomes more likely. We are no longer bound to slower speeds, or forced to store data locally, we can store and share data seamlessly among

many networks. CIDS is a part of this evolution and is a major reason why we chose this in our research. There are many other new ideas being conceived, such as the research done on securing Internet of Things devices with a blockchain-based collaborative IDS[5]. There is promise for CIDS to aid in securing intelligent electronic devices (IED) with intrusion detection[10]. Kademlia (a peer-to-peer overlay) also shows proof of concept with a CIDs-like system in [19].

## 2.4 Intersection of Blockchain and IDS

Given the built-in immutability and consensus building of blockchain technology, researchers [23, 26] have started to apply blockchain technology to tackle issues in Collaborative IDSs.

Meng et al. [26] conducted comprehensive survey on applicability of blockchain technology in intrusion detection and identified several open challenges in the field. Such as Latency, Complexity, Security, Privacy, and Limited Signature Coverage, among others, our work can address these challenges by establishing a peer-to-peer network of signatures, implementing security and trust policies via consensus building, and showing that the speed of networks has increased to allow for blockchain and peer-to-peer networks to succeed where in the past the limitations of network speeds was a limiting factor. Alexopoulous et al. [23] proposed a system that uses blockchain technology for trust building and alarm data exchange in CIDSs and discussed some design considerations.



## Chapter III.

### The Proposed Architecture

Building on Alexopoulos et al. [23]'s work, we introduce a more comprehensive architecture specifically for a signature-based CIDS. We argue that signature/rule exchange and protection are a critical part of a CIDS and blockchain technology can be used to facilitate rule exchange and secure the ruleset of each host IDS (hence we label each IDS as blockchain-IDS).

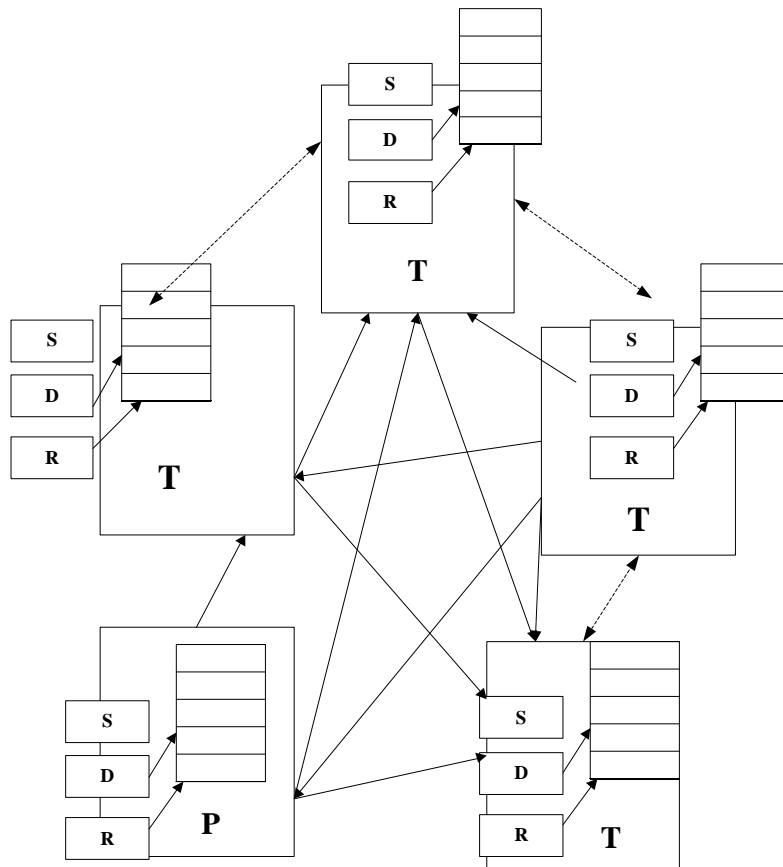


Figure. 1: Architecture of blockchain-based IDS

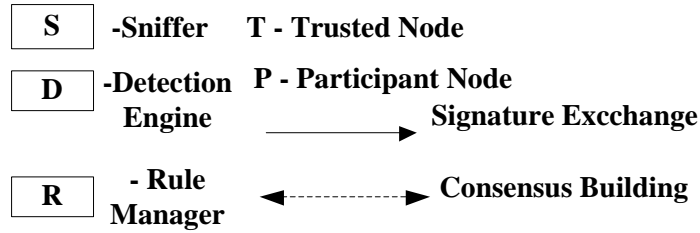


Figure 1. Legend

As shown in Fig. 1, the proposed architecture applies consortium blockchain infrastructure to build trust among participating IDS hosts and enable secure storage and exchange of rule sets. Similarly, to a traditional IDS, there are three components inside of a host in the proposed NIDS architecture: 1) a sniffer that reads and breaks down network traffic and sends them to the detection engine; 2) a detection engine that compares the packets received from the sniffer with rules/signature. 3) a rule manage that handles the maintenance of the rules in a host and rule exchange with other hosts.

Each host IDS creates block to store its rule set and alarm data while in a traditional IDS these rules are stored in ASCII format .txt files. Table 1 shows how an IDS store rules in the blocks.

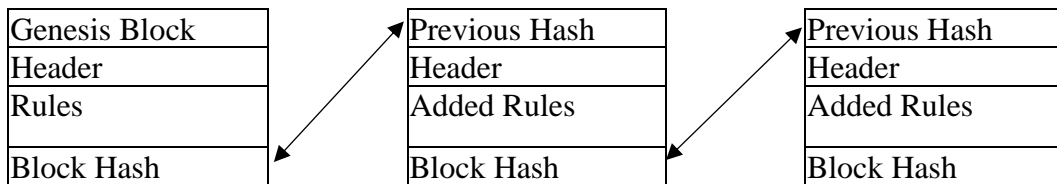


Figure. 2: Rule Storage in a Blockchain-based IDS

In the proposed architecture, there are two types of host IDS: trusted nodes (T) and participating nodes (P). All nodes can make a request to change the rule set such as adding new rules, modifying or deleting existing rules. However, only trusted nodes are involved in consensus building process which approve or reject the change request. The rule change approval process is illustrated in Figure 2.

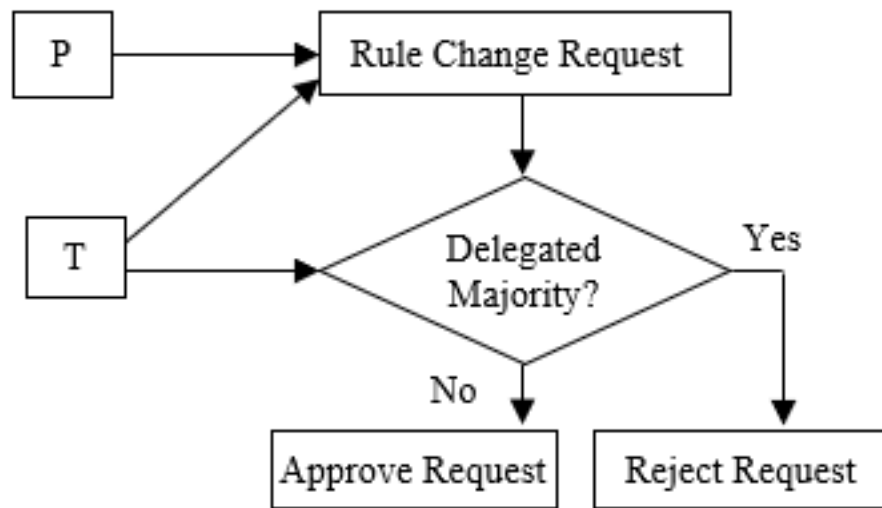


Figure 3: Consensus Building Process

Below is the general flow for adding a new rule to the system. The process for updating or removing a rule is similar to rule addition.

1. T or P node makes an add-rule request
2. Notifications are sent to all T nodes on the network that a new request needs to be voted on.
3. T nodes analyze the pending request and vote within a predesignated time frame.

4. Votes are tallied automatically by the system and the request either is approved or rejected.
5. Multiple requests can be voted on at once due to the blocks being immutable. The approved requests are implemented in batch sequence.

## **Chapter IV.**

### **Implementation**

We built a preliminary prototype for the proposed architecture. The rule set is adapted from Snort, an open-source, free and lightweight network intrusion detection system (NIDS). In term of blockchain implementation, there are three popular options: Ethereum Virtual Machine, Truffle Suite, HyperLedger.

Ethereum is best suited to cryptocurrency and would cost money to use the network.

Truffle Suite is only capable of development on the Ethereum network which would be counterproductive to a prototype for this research paper[24][20] HyperLedger is an umbrella project of open source blockchains and related tools, started in December 2015 by the Linux Foundation and supported by big industry players like IBM, Intel and SAP to support the collaborative development of blockchain-based distributed ledgers [9].

There are many tools and frameworks available via HyperLedger. We want to have to an open and consortium blockchain, hence the HyperLedger is a better choice to build our prototype.

The HyperLedger Sawtooth or Fabric framework could work well for the purposes of creating a blockchain-based IDS. The chaincode can initialize a ledger of

blockchain rules. We can then implement with javascript the following classes, this is not an exhaustive list and can be added/remove as needed.

- CreateRule- Creates a rule in the blockchain.
- RemoveRule – Remove a rule from the blockchain.
- QueryAllRules – This query will return all rules currently in the blockchain.
- QueryRuleProperties – Will return the properties of a ruleID # such as  
Port/Protocol/Owner/Etc
- UpdateRuleProperties – Allow update to the properties of a rule if something changes.
- UpdateRuleOwner -Update the owner of a rule.

A snippet of the CreateRule function written in JavaScript for the HyperLedger framework is listed as below.

```
async createRule(ctx, ruleNumber, RuleAction, protocol, sourceIP,
sourcePort, Direction, destIP, destPort, msg
sid, Revision, ClassType, Reference, RuleOwner) {
  console.info('===== START : Create Rule =====');

  const rule = {
    RuleAction,
    docType: 'rule',
    protocol,
    sourceIP,
    sourcePort,
    Direction,
    destIP,
    destPort,
    msg,
    sid,
    Revision,
    ClassType,
    Reference,
```

```
    RuleOwner,  
};  
  
    Await ctx.stub.putState(ruleNumber, Buffer.from(JSON.stringify(rule)));  
    console.info('==== END : Create Rule ====');  
}
```

These classes can be used to manipulate the blockchain from the backend. The frontend instantiates the consensus algorithm and allow for the consortium to take place on a larger scale.

Our prototype is built on a machine with following configurations: Intel Core i7-3630QM 2.4GHz with 6 MB L3 Cache, 8 GB DDR3 Memory, Dual NVIDIA GeForce GT 650M SLI. Running LUbuntu 19.04.

Our prototype can be accessed from the Github repository,

<https://github.com/delerak/bbids>

The configuration used in our benchmark is known as a “simple” config included with the Caliper framework. These config files define variables which are used during the benchmark process. Some examples of the variables are txNumber, txDuration and rateControl, these variables were left at default values for the tests that were run.

Once the benchmark test is run Caliper begins sending transactions to the blockchain network. These transactions are simply communication packets being sent between the network nodes and ensuring that the blockchain can function under network stress. None of the blockchain data is altered during these tests.

The summary in Table. 2 shows several outputs. The name of the tests is open/query these are simply labels that are used to differentiate testing variables that can

be defined in the configuration file. We defined open as opening an account within the system, and query for querying the blockchain for transactions. The transaction send rate is how many transactions are being sent per second, the latency are based on the time it takes for a transaction or query from the submission by the client until it is processed and written on the ledger. Throughput is the number of transactions or queries per second (TPS) that was processed by the blockchain network itself.

Table 1.  
*Benchmark tests of preliminary prototype.*

Test	Name	Send Rate	Latency			Throughput
			Max	Min	Avg.	
1	Open	50.3	78.16	1.26	42.43	10.3
2	Open	100.5	71.13	1.22	36.81	12.3
3	Open	149.5	74.63	1.08	38.10	12.3
4	Query	100.2	0.10	0.01	0.01	1002
5	Query	199.8	0.02	0.01	0.01	199.4

Note: 1) send rate and throughput are measured in transaction per seconds; 2) Latency is measured in seconds.

#### 4.1 Prototype Environment

The process by which I used to come to my results can be found here.

##### 4.1.2 Hardware

The hardware used during the testing process was a Lenovo IdeaPad Y500 with the following specs:

- 3rd Generation Intel Core i7-3630QM Processor( 2.40GHz 6MB)
- NVIDIA GeForce GT650M 2GB
- 8.0GB PC3-12800 DDR3 SDRAM 1600 MHz
- 1TB 5400 rpm

This is clearly an older model and has no special hardware or functionality. Running the HyperLedger and Caliper on a newer hardware would likely increase speed and response times.

#### 4.1.3 Operating Systems

The OS used during this is Lubuntu 19.04, a lightweight version of Ubuntu which requires less memory, space, and processor usage. The Linux OS was chosen due its speed and efficiency of the system and the ease of usage within the scope of software development and testing. Lubuntu was specifically chosen for its fast, lightweight, clean and easy-to-use interface. A full Ubuntu install would work just as well with a faster computer.

#### 4.1.4 HyperLedger

HyperLedger (HL) has a significant amount of documentation, the complexity of which cannot be understated. I will detail as best I can how I was able to get HyperLedger running on my machine and include the necessary links to the HyperLedger docs themselves.



Firstly, HyperLedger is simply an umbrella term, what you will be installing and building towards is one of the HL frameworks of which there are many. The two frameworks used in my project are Fabric (for the blockchain) and Caliper (for the benchmark). Sawtooth is another option, however, Sawtooth was a much more complex and confusing sort of blockchain. It should be noted that Sawtooth could conceptually work as the blockchain but due to time constraints I decided that Fabric would be the best option since it has a plug-n-play sort of setup, whereas Sawtooth is a customizable, feature-rich framework which requires deep understanding of the setup.

#### 4.1.5 Prerequisites

There are many prerequisites which can be timely to sort through. There is no easy way to get these done, while some Linux distributions might already include these, it is very likely that you will need to spend significant time simply setting up the environment so that HyperLedger can work. I will state that there is a very concise and solid guide to do this from the folks over at HyperLedger, I will include my own walkthrough, but I highly recommend that the official site also be used. You can find the link below.

<https://hyperledger-fabric.readthedocs.io/en/latest/prereqs.html>

#### cURL

The curl tool is necessary for some installs and typically comes preconfigured on most Linux distros. If you don't have it download it here: <https://curl.haxx.se/download.html>

#### Docker and Docker Compose

Docker and Docker Compose are required for HyperLedger to run. This is one of the most difficult steps in the process because Docker in and of itself is a complex concept to grasp. I highly recommend that at least some familiarization with containerization and Docker fundamentals is understood before moving any further. This link is very helpful to begin: <https://www.docker.com/resources/what-container>

Docker documentation: <https://docs.docker.com/>

Once you have Docker installed and have some familiarity with it, I recommend going thru the Docker tutorials completely and running containers and learning how the network nodes will work, this will save you a lot of headaches once you get to installing HyperLedger. Once you feel you have a firm grasp of Docker begin with the next steps of the prerequisites.

#### Go Programming Language

The Go programming language is required for HyperLedger to run. Thankfully this step is straightforward and simple, you just need to download the package and install it.

#### Node.js runtime and Node Package Manger (NPM)

Node.js and NPM are required to use HyperLedger. Node.js comes prepackaged with NPM. One of the major problems you may run into with Node/NPM is the version usage. You really should consider using NVM (node version manager) and then downgrading to what is required to run HyperLedger. HyperLedger runs on 8.0x Node/NPM therefore you cannot simply `sudo apt-get npm` `sudo apt-get node` and have it work. You need to install NVM and then use NVM to install a specific version of npm/node and go from there.

## Python

Python is needed to use node.js. Simply download and install Python on your linux distro.

## Fabric

Ah finally we get to install the blockchain software. Hopefully we don't get any errors but that is highly unlikely. The HyperLedger documentation should suffice from this point on found at:

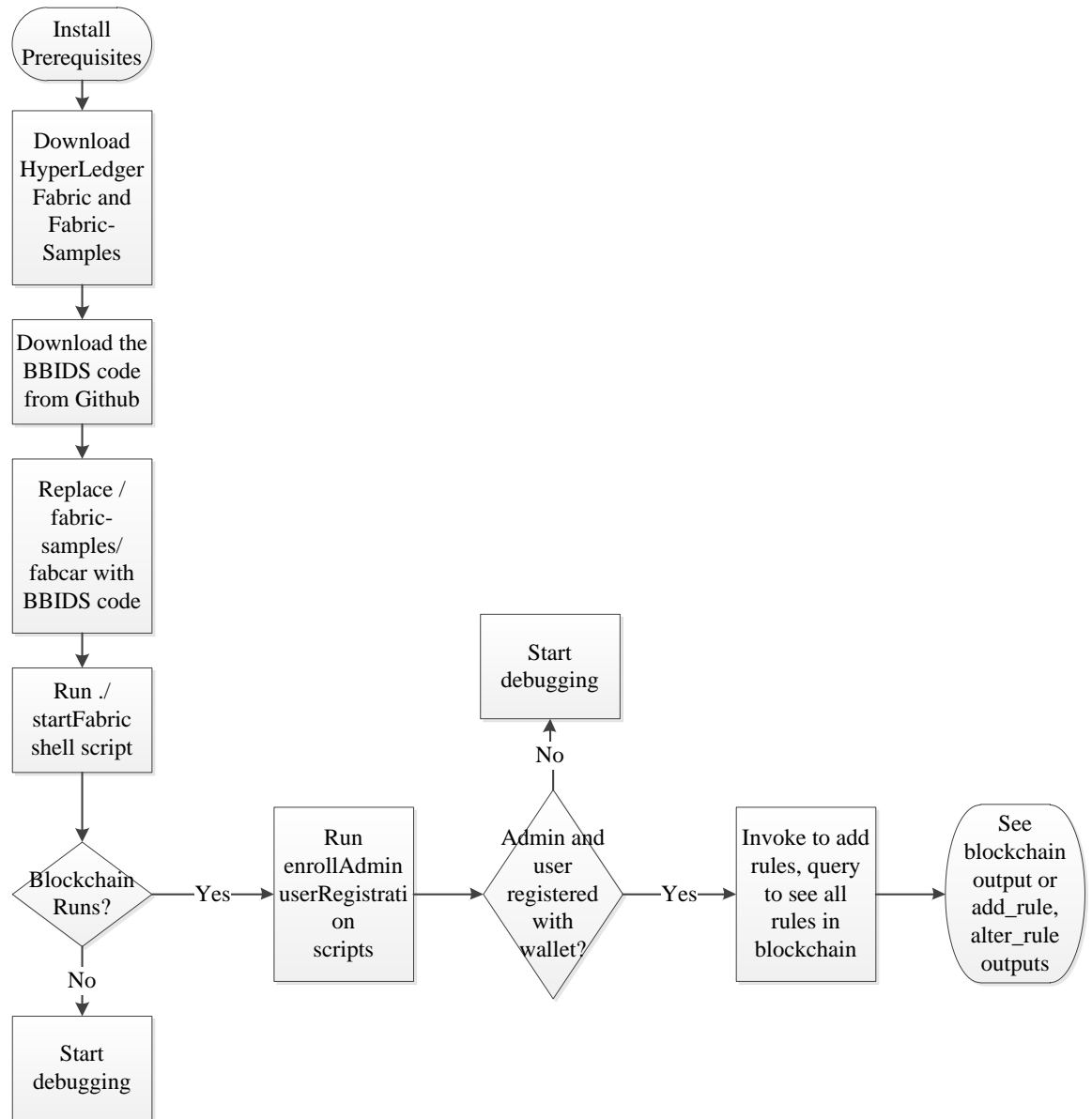
<https://hyperledger-fabric.readthedocs.io/en/latest/install.html>

There were many points where I had to debug and the best resources if an error is encountered is to use Google and find StackOverflow results.

## 4.2 Prototype Guide

This section will be used to display the blockchains features such as rule\_add, rule\_alter, change\_owner functions. There are several steps that are outlined to describe and show how the prototype works from start to finish. The prototype is a proof of concept and would require more work to finish it completely. There will be a TODO section at the end of the prototype guide to show what work would be useful to have done to simplify the process and make it more robust.

Below you will find a flowchart detailing the steps necessary to get the prototype up and running.



#### 4.2.1 Setting up Blockchain

Once the prerequisites are installed we begin by moving the BBIDS github code from [github.com/delerak/bbids](https://github.com/delerak/bbids), we can clone the repository from the command line.

Step 1: Clone github repo. Make sure you are in the directory you need to be.

```
laufentech@laufentech-linux-server2019:~/bbids$ git clone https://github.com/delerak/bbids.git
Cloning into 'bbids'...
remote: Enumerating objects: 232, done.
remote: Counting objects: 100% (232/232), done.
remote: Compressing objects: 100% (168/168), done.
remote: Total 232 (delta 42), reused 226 (delta 42), pack-reused 0
Receiving objects: 100% (232/232), 49.85 MiB | 425.00 KiB/s, done.
Resolving deltas: 100% (42/42), done.
laufentech@laufentech-linux-server2019:~/bbids$
```

Figure. 4: Cloning the git repository

HyperLedger uses Go and your \$GOPATH will probably be /home/go/src/github.com/\*

I recommend cloning into that path and then working from there or else you will have path errors for the rest of the setup.

Step 2:

Start the network with the startFabric script as seen below.

```

elp javascript javascript-low-level startFabric.sh typescript
aufentech@aufentech-linux-server2019:~/bbids/fabcar$ ./startFabric.sh javascript

# don't rewrite paths for Windows Git Bash users
export MSYS_NO_PATHCONV=1

docker-compose -f docker-compose.yml down
Removing cli ... done
Removing peer0.org1.example.com ... done
Removing couchdb ... done
Removing orderer.example.com ... done
Removing ca.example.com ... done
Removing network net_basic

docker-compose -f docker-compose.yml up -d ca.example.com orderer.example.com peer0.org1
WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All co
To deploy your application across the swarm, use `docker stack deploy`.

Creating network "net_basic" with the default driver
Creating ca.example.com ... done
Creating couchdb ... done
Creating orderer.example.com ... done
Creating peer0.org1.example.com ... done

```

Figure. 5: Starting the fabric blockchain network with javascript as the selected codebase

You will see output like the following:

```

WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the current node
To deploy your application across the swarm, use `docker stack deploy`.

Creating cli ... done
CONTAINER ID        IMAGE                      STATUS                PORTS                NAMES                COMMAND
37b9eabd93e5h"    hyperledger/fabric-tools  Up Less than a second  cli                  peer0.org1.example.com
9979388374e9e start" 28 seconds ago      Up 20 seconds        0.0.0.0:7051->7051/tcp, 0.0.0.0:7053->7053/tcp peer0.org1.example.com
7e27cdd4e1cf    hyperledger/fabric-orderer  Up 28 seconds        0.0.0.0:7050->7050/tcp orderer.example.com
426aee476871    hyperledger/fabric-ca      Up 28 seconds        0.0.0.0:7054->7054/tcp ca.example.com
5096a21d9b06    hyperledger/fabric-couchdb  Up 28 seconds        4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp couchdb
/docker-ent... 37 seconds ago      Up 28 seconds        dev-peer0.org1.example.com-fabcar-1.0-5c906e402ed29f20260ae42283216aa75549c571e2e380f3615826365d8269ba "/bin/sh
532c7d9578fa    dev-peer0.org1.example.com-fabcar-1.0-5c906e402ed29f20260ae42283216aa75549c571e2e380f3615826365d8269ba "/bin/sh
-c 'cd /usr... 30 hours ago      Exited (137) About an hour ago dev-peer0.org1.example.com-fabcar-1.0

2019-06-29 00:52:55.628 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2019-06-29 00:52:55.628 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
2019-06-29 00:52:56.048 UTC [chaincodeCmd] install -> INFO 003 Installed remotely response:<status:200 payload:"OK" >
2019-06-29 00:52:56.493 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2019-06-29 00:52:56.493 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
2019-06-29 00:53:15.121 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200

Total setup execution time : 61 secs

```

Figure. 6: Blockchain startup output

It could take some time to initialize the ledger of rules. This current iteration only has 50 IDS rules and took 115 seconds:

```
Total setup execution time : 115 secs ...
Next, use the BBIDS applications to interact with the deployed BBIDS contract.

Start by changing into the "javascript" directory:
  cd javascript

Next, install all required packages:
  npm install

Then run the following applications to enroll the admin user, and register a new user
called user1 which will be used by the other applications to interact with the deployed
BBIDS contract:
  node enrollAdmin - this is considered the Trusted Node in the BBIDS Architecture
  node registerUser - this is considered a Participating Node in the BBIDS Architecture

You can run the invoke application as follows. By default, the invoke application will
create a new rule, but you can update the application to submit other transactions:
  node invoke

You can run the query application as follows. By default, the query application will
return all rules, but you can update the application to evaluate other transactions:
  node query

NOTE: Currently everything is hardcoded so you must go into the query/invoke JavaScript
files in order to adjust their queries or invocations.
```

Figure 7: Blockchain startup script complete.

### Step 3:

Now we must install the Node packages that are required by the HyperLedger. These packages will install all required Node modules so that the code will run properly.

We do this with the command `npm install` from the `/javascript` folder.

If you run into any errors double check that you are using NPM 8.0 and not the latest version of 12.4, HyperLedger only works with 8.0x.

NVM:

Ensure you are using 8.0 or you will receive errors and will be unable to run the HyperLedger framework.

```
laufentech@laufentech-linux-server2019:~/bbids/fabcar/javascript$ nvm version
v12.4.0
laufentech@laufentech-linux-server2019:~/bbids/fabcar/javascript$ nvm use 8.0
Now using node v8.0.0 (npm v5.0.0)
laufentech@laufentech-linux-server2019:~/bbids/fabcar/javascript$
```

Figure 8: Installing the node modules with npm.

Here is the output of the npm install:

```
Now using node v8.0.0 (npm v5.0.0)
laufentech@laufentech-linux-server2019:~/bbids/fabcar/javascript$ npm install
> grpc@1.17.0 install /home/laufentech/bbids/fabcar/javascript/node_modules/grp
> node-pre-gyp install --fallback-to-build --library=static_library

node-pre-gyp WARN Using needle for node-pre-gyp https download
[grpc] Success: "/home/laufentech/bbids/fabcar/javascript/node_modules/grpc/src
de.node" is installed via remote
npm WARN fabcar@1.0.0 No repository field.

added 74 packages and removed 127 packages in 26.316s
laufentech@laufentech-linux-server2019:~/bbids/fabcar/javascript$
```

Figure 9: NPM install output screenshot

Step 4: Enroll the users (trusted node and participating nodes).

The enrollAdmin and registerUser files are used to create the accounts used in the querying and contract process. If these Node commands do not work, more than likely you have a pathing problem with your \$GOPATH and need to double-check that the code is placed within /gopath/src/github.com/fabric-samples/fabcar

Enrolling the Trusted Node with enrollAdmin:



```
laufentech@laufentech-linux-server2019:~/go/src/github.com/fabric-samples/fabcar/javascript$ node enrollAdmin.js
Wallet path: /home/laufentech/go/src/github.com/fabric-samples/fabcar/javascript/wallet
Successfully enrolled admin user "admin" and imported it into the wallet
laufentech@laufentech-linux-server2019:~/go/src/github.com/fabric-samples/fabcar/javascript$
```

Figure 10: Creating the “Trusted Node”

Enrolling the Participating Node with registerUser:

```
laufentech@laufentech-linux-server2019:~/go/src/github.com/fabric-samples/fabcar/javascript$ node registerUser.js
Wallet path: /home/laufentech/go/src/github.com/fabric-samples/fabcar/javascript/wallet
Successfully registered and enrolled admin user "user1" and imported it into the wallet
laufentech@laufentech-linux-server2019:~/go/src/github.com/fabric-samples/fabcar/javascript$
```

Figure 11: Creating the “Participating Node”

Step 5: Now that we have the network up and running and both the trusted and participant nodes installed we can query the ledger and see our rules with the ‘Node Query’ command:

The Query command returns all the rules stored in the blockchain. It is currently unformatted.

```

rule\","msg\":"MALWARE-BACKDOOR QAZ Worm Client Login access\\\"; flow:
\":"tcp\","sid\":"108\","sourceIP\":"$EXTERNAL_NET\","sourcePort
sc-activity\","Direction\":"->\","Reference\":"ruleset_community\","
afee,98775\","destIP\":"$HOME_NET\","destPort\":"7597\","docType\":"
ss\\\"; flow:to_server,established; content:\\\"qazwsx.hsq\\\"\\\"\\\";
'sourcePort\":"any\\\"}},{\\"Key\":"RULE48\","Record\":{\\"ClassType\":"
community\","Revision\":"11\","RuleAction\":"alert\","RuleOwner\":"
\":"docType\":"rule\","msg\":"MALWARE-BACKDOOR QAZ Worm Client Login a
\","protocol\":"tcp\","sid\":"108\","sourceIP\":"$EXTERNAL_NET\
asType\":"misc-activity\","Direction\":"->\","Reference\":"ruleset
leOwner\":"mcafee,98775\","destIP\":"$HOME_NET\","destPort\":"7597
ent Login access\\\"; flow:to_server,established; content:\\\"qazwsx.hs
TERNAL_NET\","sourcePort\":"any\\\"}},{\\"Key\":"RULE5\","Record\":{\\"
\":"ruleset_community\","Revision\":"11\","RuleAction\":"alert\","
t\":"7597\","docType\":"rule\","msg\":"MALWARE-BACKDOOR QAZ Worm C
'qazwsx.hsq\\\"\\\"\\\"; flow:to_server,established; content:\\\"qazwsx.hs
ord\":{\\"ClassType\":"misc-activity\","Direction\":"->\","Referenc
alert\","RuleOwner\":"mcafee,98775\","destIP\":"$HOME_NET\","destP
QAZ Worm Client Login access\\\"; flow:to_server,established; content:\\
teIP\":"$EXTERNAL_NET\","sourcePort\":"any\\\"}},{\\"Key\":"RULE7\","
\":"Reference\":"ruleset_community\","Revision\":"11\","RuleAction\":"
\","destPort\":"7597\","docType\":"rule\","msg\":"MALWARE-BACKDOO
content:\\\"qazwsx.hsq\\\"\\\"\\\"; flow:to_server,established; content:\\\"qazwsx.hsq\\\"\\\"\\\";
RULE8\","Record\":{\\"ClassType\":"misc-activity\","Direction\":"->\
eAction\":"alert\","RuleOwner\":"mcafee,98775\","destIP\":"$HOME_N
RE-BACKDOOR QAZ Worm Client Login access\\\"; flow:to_server,establishe
108\","sourceIP\":"$EXTERNAL_NET\","sourcePort\":"any\\\"}},{\\"Key\":"
on\":"->\","Reference\":"ruleset_community\","Revision\":"11\","R
: \"$HOME_NET\","destPort\":"7597\","docType\":"rule\","msg\":"MAL
established; content:\\\"qazwsx.hsq\\\"\\\"\\\"; flow:to_server,establishe
1"

```

Figure 12: IDS rules in Blockchain format.

### Step 6: Adding a Rule, Altering a Rule, Querying a Specific Rule

At this point there are 3 other functions available. Adding a rule, altering a rule, and querying a specific rule. First we show adding, then altering, then querying a specific rule. Note: Since everything is hardcoded you must change the source file in order to make these queries as shown below.

Adding a rule:

Edit invoke.js and go to the addRule function line.

```
// createRule transaction - requires 13 argument, ex: ('createRule', 'RULE12', 'alert', 'tcp', 'test', 'test', 'test', 'test', 'test', 'test', 'test', 'test', 'test', 'test', 'test')
// changeRuleOwner transaction - requires 2 args , ex: ('Rule', 'RULE10', 'Daniel')
await contract.submitTransaction('createRule', 'RULE54', 'NewRule', 'KSU',
  'test', 'test', 'test', 'test', 'test', 'test', 'test', 'test', 'test', 'test', 'test');
console.log('Transaction has been submitted');
```

Figure 13: Creating a rule within invoke.js

You can see I'm adding a rule with a couple of test fields and KSU in one. Next let's find the rule with the following command: node query | grep test

And it should highlight the new rules that were added:

```
:\ruleset_community\", \"Revision\": \"11\", \"RuleAction\": \"alert\", \"RuleOwner\": \"mcafee,98775\", \"destIP\": \"$HOME_NET\", \"destPort\": \"7597\", \"docType\": \"rule\", \"msg\": \"MALWARE-BACKDOOR QAZ Worm Client Login access\\\\\\\\\"; flow:to_server,established; content:\"qazwsx.hsqq\\\\\\\\\", \"protocol\": \"tcp\", \"sid\": \"108\", \"sourceIP\": \"$EXTERNAL_NET\", \"sourcePort\": \"any\"}}, {\"Key\": \"RULE54\", \"Record\": {\"ClassType\": \"test\", \"Direction\": \"test\", \"Reference\": \"test\", \"Revision\": \"test\", \"RuleAction\": \"Dr. Sha\", \"RuleOwner\": \"test\", \"destIP\": \"test\", \"destPort\": \"test\", \"docType\": \"rule\", \"msg\": \"test\", \"protocol\": \"test\", \"sourceIP\": \"test\", \"sourcePort\": \"test\"}}, {\"Key\": \"RULE54\", \"Record\": {\"ClassType\": \"test\", \"Direction\": \"test\", \"Reference\": \"test\", \"Revision\": \"test\", \"RuleAction\": \"NewRule\", \"RuleOwner\": \"test\", \"destIP\": \"test\", \"destPort\": \"test\", \"docType\": \"rule\", \"msg\": \"test\", \"protocol\": \"KSU\", \"sid\": \"test\", \"sourceIP\": \"test\", \"sourcePort\": \"test\"}}, {\"Key\": \"RULE6\", \"Record\": {\"ClassType\": \"misc-activity\", \"Direction\": \"->\", \"Reference\": \"ruleset_community\", \"Revision\": \"11\", \"RuleAction\": \"alert\", \"RuleOwner\": \"mcafee,98775\", \"destIP\": \"$HOME_NET\", \"destPort\": \"7597\", \"docType\": \"rule\", \"msg\": \"MALWARE-BACKDOOR QAZ Worm Client Login access\\\\\\\\\"; flow:to_server,established; content:\\\\\"qazwsx.hsqq\\\\\\\\\", \"protocol\": \"tcp\", \"sid\": \"108\", \"sourceIP\": \"$EXTERNAL_NET\", \"sourcePort\": \"any\"}} {\"Key\": \"RULE7\", \"Record\": {\"ClassType\": \"misc-activity\", \"Direction\": \"->\", \"Reference\": \"ruleset_community\", \"Revision\": \"11\", \"RuleAction\": \"alert\", \"RuleOwner\": \"mcafee,98775\", \"destIP\": \"$HOME_NET\", \"destPort\": \"7597\", \"docType\": \"rule\", \"msg\": \"MALWARE-BACKDOOR QAZ Worm Client Login access\\\\\\\\\"; flow:to_server,established; content:\\\\\"qazwsx.hsqq\\\\\\\\\", \"protocol\": \"tcp\", \"sid\": \"108\", \"sourceIP\": \"$EXTERNAL_NET\", \"sourcePort\": \"any\"}}
```

Figure 14: Output of the created 'test' rule.

Alter Rule:

We go into the source code for Invoke once more and call the function `changeRuleOwner` from the `fabcar.js` ChainCode. See below:

```
//await contract.submitTransaction('createRule', 'RULE55', 'NewRule', 'KSU',
//'test', 'test', 'test', 'test', 'test', 'test', 'test', 'test', 'test', 'test', 'test');
await contract.submitTransaction('changeRuleOwner', 'RULE54', 'Dr. Lei Li');

console.log('Transaction has been submitted');
```

Figure 15: Altering a rules 'owner' property.

Next, we run `node invoke` again.

```
laufentech@laufentech-linux-server2019:~/go/src/github.com/fabric-samples/fabcar/javascript$ node invoke
Wallet path: /home/laufentech/go/src/github.com/fabric-samples/fabcar/javascript/wallet
Transaction has been submitted
laufentech@laufentech-linux-server2019:~/go/src/github.com/fabric-samples/fabcar/javascript$
```

Figure 16: Altering rule output.

Now we will try to find the new rule owner changed with `node query | grep Dr`.

```
"sourceIP\":"$EXTERNAL_NET", "sourcePort\":"any"}}, {"Key\":"RULE53\","Reference\":"test", "Revision\":"test", "RuleAction\":"Dr. Shahriar\ntest", "docType\":"rule", "msg\":"test", "protocol\":"test", "sid\":"Key\":"RULE54", "Record\":{"ClassType\":"test", "Direction\":"test\":"NewRule", "RuleOwner\":"Dr. Lei Li", "destIP\":"test", "destPort\":"KSU", "sid\":"test", "sourceIP\":"test", "sourcePort\":"test"}}y", "Direction\":"->", "Reference\":"ruleset_community", "Revision\":"", "destIP\":"$HOME_NET", "destPort\":"7597", "docType\":"rule", "mow:to_server, established; content:\\\\qazwsx.hsq\\\\", "protocol\":"tcp\":"any"}}, {"Key\":"RULE7", "Record\":{"ClassType\":"misc-activity\":"Revision\":"11", "RuleAction\":"alert", "RuleOwner\":"mcafee,98775\":"rule", "msg\":"MALWARE-BACKDOOR QAZ Worm Client Login access\\\\"; floocol\":"tcp", "sid\":"108", "sourceIP\":"$EXTERNAL_NET", "sourcePortmisc-activity", "Direction\":"->", "Reference\":"ruleset_community", \mcafee,98775", "destIP\":"$HOME_NET", "destPort\":"7597", "docType\":"
```

Figure 17: Finding the new rule within the blockchain.

As we can see from the screenshot the RuleOwner was switched to Dr. Lei Li.

Querying a specific rule:

To query a specific rule # we have to edit the query.js file directly as shown below:

```
// queryRule transaction - requires 1 argument, ex: ('queryRule', 'RULE4')
// queryAllRules transaction - requires no arguments, ex: ('queryAllRules')
//const result = await contract.evaluateTransaction('queryAllRules');
const result = await contract.evaluateTransaction('queryRule', 'RULE54')

console.log(`Transaction has been evaluated, result is: ${result.toString()}`)
```

Figure 18: Querying for a specific rule number.

Comment out the QueryAllRules function and write in the queryRule function:

As we can see the query now only returns RULE54 in the blockchain.

```
laufentech@laufentech-linux-server2019:~/go/src/github.com/fabric-samples/fabcar/javascript$ node query
Wallet path: /home/laufentech/go/src/github.com/fabric-samples/fabcar/javascript/wallet
Transaction has been evaluated, result is: {"ClassType":"test","Direction":"test","Reference":"",
"RuleAction":"NewRule","RuleOwner":"Dr. Lei Li","destIP":"test","destPort":"test","docT
t","protocol":"KSU","sid":"test","sourceIP":"test","sourcePort":"test"}
laufentech@laufentech-linux-server2019:~/go/src/github.com/fabric-samples/fabcar/javascript$
```

Figure 19: Output of the query for RULE54.

This concludes the walkthrough for the Blockchain Based IDS prototype.

#### 4.3 YouTube Tutorial Videos

You can find my YouTube tutorial videos at the following playlist link.

<https://www.youtube.com/playlist?list=PLfobw40cSck2faRCNRK1inOt9mpF93R38>

## **Chapter V.**

### **Conclusions**

Collaborative IDS (CIDS) has been proposed to detect increasingly complex cyber-attacks. Overcoming the issue of trust and sharing of rulesets has remained to a challenge. In this paper, we proposed a CIDS architecture leveraging blockchain technology's record immutability and tamper proof properties of stored data in the distributed ledger. We provided details on various workflows for our CIDS including how to add or update rulesets. We implemented our prototype using HyperLedger frameworks and evaluated using an available benchmark. The initial results look promising, including running over 1000 transactions on the blockchain, latency numbers were low and acceptable range for a small peer-to-peer network.

Our future work plan includes applying our prototype within a real network, testing with simulated attack network traffics, and evaluating the performance with many IDS nodes in place.

## Appendix I

BBids Prototype Code, using FabCar contract in HyperLedger

### Chaincode files:

/lib/Fabcar.js

```
/*
 * SPDX-License-Identifier: Apache-2.0
 */

'use strict';

const { Contract } = require('fabric-contract-api');

class FabCar extends Contract {

  async initLedger(ctx) {
    console.info('===== START : Initialize Ledger
=====');
    const rules = [
      {
        ruleAction: 'alert',
        protocol: 'tcp',
        sourceIP: '$HOME_NET',
        sourcePort: '2589',
        Direction: '->',
        destIP: '$EXTERNAL_NET',
        destPort: 'any',
        msg: 'MALWARE-BACKDOOR - Dagger_1.4.0';
        flow:to_client,established; content:"2|00 00 00 06 00 00 00|Drives|24
00|",depth 16',
        sid: '105',
        Revision: '14',
```

```

        ClassType: 'misc-activity',
        Reference: 'ruleset_community',
        RuleOwner: 'laufenberg',
    },
    {
        RuleAction: 'alert',
        protocol: 'tcp',
        sourceIP: '$EXTERNAL_NET',
        sourcePort: 'any',
        Direction: '->',
        destIP: '$HOME_NET',
        destPort: '7597',
        msg: 'MALWARE-BACKDOOR QAZ Worm Client Login access";
flow:to_server,established; content:"qazwsx.hsq"',
        sid: '108',
        Revision: '11',
        ClassType: 'misc-activity',
        Reference: 'ruleset_community',
        RuleOwner: 'mcafee,98775',
    },
    {
        RuleAction: 'alert',
        protocol: 'tcp',
        sourceIP: '$EXTERNAL_NET',
        sourcePort: 'any',
        Direction: '->',
        destIP: '$HOME_NET',
        destPort: 20034,
        msg: 'MALWARE-BACKDOOR NetBus Pro 2.0 connection
established"; flow:to_client,established;
flowbits:isset,backdoor.netbus_2.connect; content:"BN|10 00 02 00 "',
        sid: 115,
        Revision: '15',
        ClassType: 'misc-activity',
        Reference: 'ruleset_community',
        RuleOwner: 'none',
    },
    {
        RuleAction: 'alert',
        protocol: 'tcp',
        sourceIP: '$EXTERNAL_NET',
        sourcePort: 'any',
        Direction: '->',

```



```

    destIP: '$HOME_NET',
    destPort: '7597',
    msg: 'MALWARE-BACKDOOR Infector.1.x";
flow:established,to_client; content:"WHATISIT",depth 9;
metadata:impact_flag red,ruleset community; reference:nessus,11157',
    sid: '117',
    Revision: '17',
    ClassType: 'misc-activity',
    Reference: 'ruleset_community',
    RuleOwner: 'none',
  },
  {
    RuleAction: 'alert',
    protocol: 'tcp',
    sourceIP: '$HOME_NET',
    sourcePort: 'any',
    Direction: '->',
    destIP: '$EXTERNAL_NET',
    destPort: '666',
    msg: "MALWARE-BACKDOOR SatansBackdoor.2.0.Beta";
flow:to_client,established; content:"Remote|3A| ",depth 11,nocase;
content:"You are connected to me.|0D 0A|Remote|3A| Ready for commands',
    sid: '118',
    Revision: '12',
    ClassType: 'misc-activity',
    Reference: 'ruleset_community',
    RuleOwner: 'none',
  },
  {
    RuleAction: 'alert',
    protocol: 'tcp',
    sourceIP: '$EXTERNAL_NET',
    sourcePort: '12345',

    Direction: '->',
    destIP: '$HOME_NET',
    destPort: 12345,
    msg: 'MALWARE-BACKDOOR netbus getinfo";
flow:to_server,established; content:"GetInfo|0D| "',
    sid: '110',
    Revision: '10',
    ClassType: 'trojan-activity',
    Reference: 'ruleset_community',

```

```

    RuleOwner: 'none',
  },
  {
    RuleAction: 'alert',
    protocol: 'tcp',
    sourceIP: '$HOME_NET',
    sourcePort: 'any',
    Direction: '->',
    destIP: '$EXTERNAL_NET',
    destPort: '6789',
    msg: 'MALWARE-BACKDOOR Doly 2.0 access';
flow:established,to_client; content:"Wtzup Use '",
    sid: '119',
    Revision: '11',
    ClassType: 'misc-activity',
    Reference: 'ruleset_community',
    RuleOwner: 'none',
  },
  {
    RuleAction: 'alert',
    protocol: 'tcp',
    sourceIP: '$HOME_NET',
    sourcePort: 'any',
    Direction: '->',
    destIP: '$EXTERNAL_NET',
    destPort: '7597',
    msg: 'MALWARE-BACKDOOR Infector 1.6 Client to Server
Connection Request'; flow:to_server,established; content:"FC'",
    sid: '121',
    Revision: '14',
    ClassType: 'misc-activity',
    Reference: 'ruleset_community',
    RuleOwner: 'nessus,1157',
  },
  {
    RuleAction: 'alert',
    protocol: 'tcp',
    sourceIP: '$EXTERNAL_NET',
    sourcePort: '31785',
    Direction: '->',
    destIP: '$HOME_NET',
    destPort: 'any',
  }

```

```

        msg: 'MALWARE-BACKDOOR HackAttack 1.20 Connect';
flow:established,to_client; content:"host",
    sid: '141',
    Revision: '10',
    ClassType: 'misc-activity',
    Reference: 'ruleset_community',
    RuleOwner: 'none',
},
{
    RuleAction: 'alert',
    protocol: 'tcp',
    sourceIP: '$EXTERNAL_NET',
    sourcePort: 'any',
    Direction: '->',
    destIP: '$HOME_NET',
    destPort: '21',
    msg: 'PROTOCOL-FTP ADMw0rm ftp login attempt';
flow:to_server,established; content:"USER",nocase; content:"w0rm",distance
1,nocase; pcre:"/^USER\s+w0rm/smi",
    sid: '144',
    Revision: '16',
    ClassType: 'misc-activity',
    Reference: 'ruleset_community',
    RuleOwner: 'mcafee,98775',
},
{
    RuleAction: 'alert',
    protocol: 'tcp',
    sourceIP: '$EXTERNAL_NET',
    sourcePort: 'any',
    Direction: '->',
    destIP: '$HOME_NET',
    destPort: '666',
    msg: "'MALWARE-BACKDOOR BackConstruction 2.1 Client FTP
Open Request"; flow:to_server,established; content:"FTPON '",
    sid: '157',
    Revision: '16',
    ClassType: 'misc-activity',
    Reference: 'ruleset_community',
    RuleOwner: 'mcafee,98775',
},
{
    RuleAction: 'alert',

```

```

    protocol: 'tcp',
    sourceIP: '$EXTERNAL_NET',
    sourcePort: 'any',
    Direction: '->',
    destIP: '$TELNET_SERVICES',
    destPort: '23',
    msg: 'MALWARE-BACKDOOR w00w00 attempt';
flow:to_server,established; content:"w00w00 ",
    sid: '209',
    Revision: '19',
    ClassType: 'attempted_admin',
    Reference: 'ruleset_community',
    RuleOwner: 'mcafee,98775',
},
{
    RuleAction: 'alert',
    protocol: 'tcp',
    sourceIP: '$EXTERNAL_NET',
    sourcePort: 'any',
    Direction: '->',
    destIP: '$TELNET_SERVICES',
    destPort: '23',
    msg: 'MALWARE-BACKDOOR w00w00 attempt';
flow:to_server,established; content:"w00w00 ",
    sid: '210',
    Revision: '19',
    ClassType: 'attempted_admin',
    Reference: 'ruleset_community',
    RuleOwner: 'mcafee,98775',
},
{
    RuleAction: 'alert',
    protocol: 'tcp',
    sourceIP: '$EXTERNAL_NET',
    sourcePort: 'any',
    Direction: '->',
    destIP: '$TELNET_SERVICES',
    destPort: '23',
    msg: 'MALWARE-BACKDOOR w00w00 attempt';
flow:to_server,established; content:"w00w00 ",
    sid: '211',
    Revision: '19',
    ClassType: 'attempted_admin',

```

```

Reference: 'ruleset_community',
RuleOwner: 'mcafee,98775',
},
{
RuleAction: 'alert',
protocol: 'tcp',
sourceIP: '$EXTERNAL_NET',
sourcePort: 'any',
Direction: '->',
destIP: '$TELNET_SERVICES',
destPort: '23',
msg: 'MALWARE-BACKDOOR w00w00 attempt";
flow:to_server,established; content:"w00w00 "',
sid: '212',
Revision: '19',
ClassType: 'attempted_admin',
Reference: 'ruleset_community',
RuleOwner: 'mcafee,98775',
},
{
RuleAction: 'alert',
protocol: 'tcp',
sourceIP: '$EXTERNAL_NET',
sourcePort: 'any',
Direction: '->',
destIP: '$TELNET_SERVICES',
destPort: '23',
msg: 'MALWARE-BACKDOOR w00w00 attempt";
flow:to_server,established; content:"w00w00 "',
sid: '213',
Revision: '19',
ClassType: 'attempted_admin',
Reference: 'ruleset_community',
RuleOwner: 'mcafee,98775',
},
{
RuleAction: 'alert',
protocol: 'tcp',
sourceIP: '$EXTERNAL_NET',
sourcePort: 'any',
Direction: '->',
destIP: '$TELNET_SERVICES',
destPort: '23',

```

```

        msg: 'MALWARE-BACKDOOR w00w00 attempt';
flow:to_server,established; content:"w00w00 ",
    sid: '214',
    Revision: '19',
    ClassType: 'attempted_admin',
    Reference: 'ruleset_community',
    RuleOwner: 'mcafee,98775',
},
{
    RuleAction: 'alert',
    protocol: 'tcp',
    sourceIP: '$EXTERNAL_NET',
    sourcePort: 'any',
    Direction: '->',
    destIP: '$TELNET_SERVICES',
    destPort: '23',
    msg: 'MALWARE-BACKDOOR w00w00 attempt';
flow:to_server,established; content:"w00w00 ",
    sid: '215',
    Revision: '19',
    ClassType: 'attempted_admin',
    Reference: 'ruleset_community',
    RuleOwner: 'mcafee,98775',
},
{
    RuleAction: 'alert',
    protocol: 'tcp',
    sourceIP: '$EXTERNAL_NET',
    sourcePort: 'any',
    Direction: '->',
    destIP: '$TELNET_SERVICES',
    destPort: '23',
    msg: 'MALWARE-BACKDOOR r00t attempt';
flow:to_server,established; content:" r00t",
    sid: '216',
    Revision: '19',
    ClassType: 'attempted_admin',
    Reference: 'ruleset_community',
    RuleOwner: 'mcafee,98775',
},
{
    RuleAction: 'alert',
    protocol: 'tcp',

```

```

        sourceIP: '$EXTERNAL_NET',
        sourcePort: 'any',
        Direction: '->',
        destIP: '$TELNET_SERVICES',
        destPort: '23',
        msg: 'MALWARE-BACKDOOR rewt attempt";
flow:to_server,established; content:" rewt"',
        sid: '217',
        Revision: '19',
        ClassType: 'attempted_admin',
        Reference: 'ruleset_community',
        RuleOwner: 'mcafee,98775',
    },
    {
        RuleAction: 'alert',
        protocol: 'tcp',
        sourceIP: '$EXTERNAL_NET',
        sourcePort: 'any',
        Direction: '->',
        destIP: '$TELNET_SERVICES',
        destPort: '23',
        msg: 'MALWARE-BACKDOOR attempt";
flow:to_server,established; content:" wh00t"',
        sid: '218',
        Revision: '19',
        ClassType: 'attempted_admin',
        Reference: 'ruleset_community',
        RuleOwner: 'mcafee,98775',
    },
    {
        RuleAction: 'alert',
        protocol: 'tcp',
        sourceIP: '$EXTERNAL_NET',
        sourcePort: 'any',
        Direction: '->',
        destIP: '$TELNET_SERVICES',
        destPort: '23',
        msg: 'MALWARE-BACKDOOR d13hh attempt";
flow:to_server,established; content:" d13hh"',
        sid: '219',
        Revision: '19',
        ClassType: 'attempted_admin',
        Reference: 'ruleset_community',
    }

```

```

    RuleOwner: 'mcafee,98775',
  },
  {
    RuleAction: 'alert',
    protocol: 'tcp',
    sourceIP: '$EXTERNAL_NET',
    sourcePort: 'any',
    Direction: '->',
    destIP: '$TELNET_SERVICES',
    destPort: '23',
    msg: 'MALWARE-BACKDOOR lrkr0x attempt';
flow:to_server,established; content:" lrkr0x",
    sid: '220',
    Revision: '19',
    ClassType: 'attempted_admin',
    Reference: 'ruleset_community',
    RuleOwner: 'mcafee,98775',
  },
  {
    RuleAction: 'alert',
    protocol: 'udp',
    sourceIP: '$EXTERNAL_NET',
    sourcePort: 'any',
    Direction: '->',
    destIP: '$HOME_NET',
    destPort: '53',
    msg: 'PROTOCOL-DNS dns zone transfer via TCP detected";
flow:to_server,established; content:"|00 01 00 00 00 00 00|",depth 8,offset 6;
byte_test:1,!&,0xF8,4"',
    sid: '255',
    Revision: '23',
    ClassType: 'attempted-recon',
    Reference: 'ruleset_community',
    RuleOwner: 'none',
  },
  {
    RuleAction: 'alert',
    protocol: 'udp',
    sourceIP: '$EXTERNAL_NET',
    sourcePort: 'any',
    Direction: '->',
    destIP: '$HOME_NET',
    destPort: '53',
  }

```



```

        msg: 'PROTOCOL-DNS dns zone transfer via TCP detected";
flow:to_server,established; content:"|00 01 00 00 00 00 00|",depth 8,offset 6;
byte_test:1,!&,0xF8,4",
    sid: '256',
    Revision: '12',
    ClassType: 'attempted-admin',
    Reference: 'ruleset_community',
    RuleOwner: 'none',
},
{
    RuleAction: 'alert',
    protocol: 'udp',
    sourceIP: '$EXTERNAL_NET',
    sourcePort: 'any',
    Direction: '->',
    destIP: '$HOME_NET',
    destPort: '53',
    msg: 'PROTOCOL-DNS dns zone transfer via TCP detected";
flow:to_server,established; content:"|00 01 00 00 00 00 00|",depth 8,offset 6;
byte_test:1,!&,0xF8,4",
    sid: '257',
    Revision: '12',
    ClassType: 'misc-activity',
    Reference: 'ruleset_community',
    RuleOwner: 'mcafee,98775',
},
{
    RuleAction: 'alert',
    protocol: 'udp',
    sourceIP: '$EXTERNAL_NET',
    sourcePort: 'any',
    Direction: '->',
    destIP: '$HOME_NET',
    destPort: '53',
    msg: 'PROTOCOL-DNS dns zone transfer via TCP detected";
flow:to_server,established; content:"|00 01 00 00 00 00 00|",depth 8,offset 6;
byte_test:1,!&,0xF8,4",
    sid: '258',
    Revision: '22',
    ClassType: 'misc-activity',
    Reference: 'ruleset_community',
    RuleOwner: 'mcafee,98775',
},

```

```

{
  RuleAction: 'alert',
  protocol: 'udp',
  sourceIP: '$EXTERNAL_NET',
  sourcePort: 'any',
  Direction: '->',
  destIP: '$HOME_NET',
  destPort: '53',
  msg: 'PROTOCOL-DNS dns zone transfer via TCP detected";
flow:to_server,established; content:"|00 01 00 00 00 00 00|",depth 8,offset 6;
byte_test:1,!&,0xF8,4"',
  sid: '259',
  Revision: '5',
  ClassType: 'misc-activity',
  Reference: 'ruleset_community',
  RuleOwner: 'mcafee,98775',
},
{
  RuleAction: 'alert',
  protocol: 'udp',
  sourceIP: '$EXTERNAL_NET',
  sourcePort: 'any',
  Direction: '->',
  destIP: '$HOME_NET',
  destPort: '53',
  msg: 'PROTOCOL-DNS dns zone transfer via TCP detected";
flow:to_server,established; content:"|00 01 00 00 00 00 00|",depth 8,offset 6;
byte_test:1,!&,0xF8,4"',
  sid: '260',
  Revision: '23',
  ClassType: 'misc-activity',
  Reference: 'ruleset_community',
  RuleOwner: 'mcafee,98775',
},
{
  RuleAction: 'alert',
  protocol: 'udp',
  sourceIP: '$EXTERNAL_NET',
  sourcePort: 'any',
  Direction: '->',
  destIP: '$HOME_NET',
  destPort: '53',

```

```

        msg: 'PROTOCOL-DNS dns zone transfer via TCP detected";
flow:to_server,established; content:"|00 01 00 00 00 00 00|",depth 8,offset 6;
byte_test:1,!&,0xF8,4"',
    sid: '261',
    Revision: '23',
    ClassType: 'misc-activity',
    Reference: 'ruleset_community',
    RuleOwner: 'mcafee,98775',
},
{
    RuleAction: 'alert',
    protocol: 'udp',
    sourceIP: '$EXTERNAL_NET',
    sourcePort: 'any',
    Direction: '->',
    destIP: '$HOME_NET',
    destPort: '53',
    msg: 'PROTOCOL-DNS dns zone transfer via TCP detected";
flow:to_server,established; content:"|00 01 00 00 00 00 00|",depth 8,offset 6;
byte_test:1,!&,0xF8,4"',
    sid: '262',
    Revision: '23',
    ClassType: 'misc-activity',
    Reference: 'ruleset_community',
    RuleOwner: 'mcafee,98775',
},
{
    RuleAction: 'alert',
    protocol: 'udp',
    sourceIP: '$EXTERNAL_NET',
    sourcePort: 'any',
    Direction: '->',
    destIP: '$HOME_NET',
    destPort: '53',
    msg: 'PROTOCOL-DNS dns zone transfer via TCP detected";
flow:to_server,established; content:"|00 01 00 00 00 00 00|",depth 8,offset 6;
byte_test:1,!&,0xF8,4"',
    sid: '263',
    Revision: '23',
    ClassType: 'misc-activity',
    Reference: 'ruleset_community',
    RuleOwner: 'mcafee,98775',
},

```

```

{
  RuleAction: 'alert',
  protocol: 'udp',
  sourceIP: '$EXTERNAL_NET',
  sourcePort: 'any',
  Direction: '->',
  destIP: '$HOME_NET',
  destPort: '53',
  msg: 'PROTOCOL-DNS dns zone transfer via TCP detected";
flow:to_server,established; content:"|00 01 00 00 00 00 00|",depth 8,offset 6;
byte_test:1,!&0xF8,4"',
  sid: '264',
  Revision: '23',
  ClassType: 'misc-activity',
  Reference: 'ruleset_community',
  RuleOwner: 'mcafee,98775',
},
{
  RuleAction: 'alert',
  protocol: 'udp',
  sourceIP: '$EXTERNAL_NET',
  sourcePort: 'any',
  Direction: '->',
  destIP: '$HOME_NET',
  destPort: '53',
  msg: 'PROTOCOL-DNS dns zone transfer via TCP detected";
flow:to_server,established; content:"|00 01 00 00 00 00 00|",depth 8,offset 6;
byte_test:1,!&0xF8,4"',
  sid: '265',
  Revision: '23',
  ClassType: 'misc-activity',
  Reference: 'ruleset_community',
  RuleOwner: 'mcafee,98775',
},
{
  RuleAction: 'alert',
  protocol: 'udp',
  sourceIP: '$EXTERNAL_NET',
  sourcePort: 'any',
  Direction: '->',
  destIP: '$HOME_NET',
  destPort: '53',

```

```

        msg: 'PROTOCOL-DNS dns zone transfer via TCP detected";
flow:to_server,established; content:"|00 01 00 00 00 00 00|",depth 8,offset 6;
byte_test:1,!&,0xF8,4",
    sid: '266',
    Revision: '23',
    ClassType: 'misc-activity',
    Reference: 'ruleset_community',
    RuleOwner: 'mcafee,98775',
  },
];

```

```

for (let i = 0; i < rules.length; i++) {
  rules[i].docType = 'rule';
  await ctx.stub.putState('RULE' + i,
Buffer.from(JSON.stringify(rules[i])));
  console.info('Added <--> ', rules[i]);
}
console.info('===== END : Initialize Ledger
=====');
}

```

```

async queryRule(ctx, ruleNumber) {
  const ruleAsBytes = await ctx.stub.getState(ruleNumber); // get the rule
from chaincode state
  if (!ruleAsBytes || ruleAsBytes.length === 0) {
    throw new Error(`${ruleNumber} does not exist`);
  }
  console.log(ruleAsBytes.toString());
  return ruleAsBytes.toString();
}

```

```

async createRule(ctx, ruleNumber, RuleAction, protocol, sourceIP,
sourcePort, Direction, destIP, destPort, msg, sid, Revision, ClassType,
Reference, RuleOwner) {
  console.info('===== START : Create Rule =====');

```

```

const rule = {
  RuleAction,
  docType: 'rule',
  sourceIP,
  protocol,
  sourcePort,
  Direction,

```

```

    destIP,
    destPort,
    msg,
    sid,
    Revision,
    ClassType,
    Reference,
    RuleOwner
};

await ctx.stub.putState(ruleNumber, Buffer.from(JSON.stringify(rule)));
console.info('===== END : Create Rule =====');
}

async queryAllRules(ctx) {
  const startKey = 'RULE0';
  const endKey = 'RULE999';

  const iterator = await ctx.stub.getStateByRange(startKey, endKey);

  const allResults = [];
  while (true) {
    const res = await iterator.next();

    if (res.value && res.value.value.toString()) {
      console.log(res.value.value.toString('utf8'));

      const Key = res.value.{'print $3'}key;
      let Record;
      try {
        Record = JSON.parse(res.value.value.toString('utf8'));
      } catch (err) {
        console.log(err);
        Record = res.value.value.toString('utf8');
      }
      allResults.push({ Key, Record });
    }
  }
  if (res.done) {
    console.log('end of data');
    await iterator.close();
    console.info(allResults);
    return JSON.stringify(allResults);
  }
}

```

```

    }
  }

  async changeRuleOwner(ctx, ruleNumber, newOwner) {
    console.info('===== START : changeOwner
=====');

    const ruleAsBytes = await ctx.stub.getState(ruleNumber); // get the rule
    from chaincode state
    if (!ruleAsBytes || ruleAsBytes.length === 0) {
      throw new Error(`${ruleNumber} does not exist`);
    }
    const rule = JSON.parse(ruleAsBytes.toString());
    rule.RuleOwner = newOwner;

    await ctx.stub.putState(ruleNumber, Buffer.from(JSON.stringify(rule)));
    console.info('===== END : changeOwner =====');
  }
}

module.exports = FabCar;

```

## Startup Script:

```

#!/bin/
n/ba
sh

#
# Copyright IBM Corp All Rights Reserved
#
# SPDX-License-Identifier: Apache-2.0
#
# Exit on first error
set -e

# don't rewrite paths for Windows Git Bash users
export MSYS_NO_PATHCONV=1

```

```

starttime=$(date +%s)
CC_SRC_LANGUAGE=${1:-"go"}
CC_SRC_LANGUAGE=`echo "$CC_SRC_LANGUAGE" | tr [:upper:]
[:lower:]`
if [ "$CC_SRC_LANGUAGE" = "go" -o "$CC_SRC_LANGUAGE" =
"golang" ]; then
    CC_RUNTIME_LANGUAGE=golang
    CC_SRC_PATH=github.com/fabcar/go
elif [ "$CC_SRC_LANGUAGE" = "javascript" ]; then
    CC_RUNTIME_LANGUAGE=node # chaincode runtime language is
node.js
    CC_SRC_PATH=/opt/gopath/src/github.com/fabcar/javascript
elif [ "$CC_SRC_LANGUAGE" = "typescript" ]; then
    CC_RUNTIME_LANGUAGE=node # chaincode runtime language is
node.js
    CC_SRC_PATH=/opt/gopath/src/github.com/fabcar/typescript
    echo Compiling TypeScript code into JavaScript ...
    pushd ../chaincode/fabcar/typescript
    npm install
    npm run build
    popd
    echo Finished compiling TypeScript code into JavaScript
else
    echo The chaincode language ${CC_SRC_LANGUAGE} is not
supported by this script
    echo Supported chaincode languages are: go, javascript, and typescript
    exit 1
fi

# clean the keystore
rm -rf ./hfc-key-store

# launch network; create channel and join peer to channel
cd ../basic-network
./start.sh

# Now launch the CLI container in order to install, instantiate chaincode
# and prime the ledger with our 50 rules
docker-compose -f ./docker-compose.yml up -d cli
docker ps -a

```



```

docker exec -e "CORE_PEER_LOCALMSPID=Org1MSP" -e
"CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger
/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.
example.com/msp" cli peer chaincode install -n fabcar -v 1.0 -p
"$CC_SRC_PATH" -l "$CC_RUNTIME_LANGUAGE"
docker exec -e "CORE_PEER_LOCALMSPID=Org1MSP" -e
"CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger
/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.
example.com/msp" cli peer chaincode instantiate -o
orderer.example.com:7050 -C mychannel -n fabcar -l
"$CC_RUNTIME_LANGUAGE" -v 1.0 -c '{"Args":[]}' -P "OR
(Org1MSP.member,'Org2MSP.member')"
sleep 10
docker exec -e "CORE_PEER_LOCALMSPID=Org1MSP" -e
"CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger
/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.
example.com/msp" cli peer chaincode invoke -o orderer.example.com:7050 -
C mychannel -n fabcar -c '{"function":"initLedger","Args":[]}'

```

cat <<EOF

Total setup execution time : \$(((\$(date +%s) - starttime)) secs ...

Next, use the BBIDS applications to interact with the deployed BBIDS contract.

Start by changing into the "javascript" directory:

```
cd javascript
```

Next, install all required packages:

```
npm install
```

Then run the following applications to enroll the admin user, and register a new user

called user1 which will be used by the other applications to interact with the deployed

BBIDS contract:

node enrollAdmin - this is considered the Trusted Node in the BBIDS Architecture

node registerUser - this is considered a Participating Node in the BBIDS Architecture

You can run the invoke application as follows. By default, the invoke application will

create a new rule, but you can update the application to submit other transactions:

```
node invoke
```

You can run the query application as follows. By default, the query application will

return all rules, but you can update the application to evaluate other transactions:

node query

NOTE: Currently everything is hardcoded so you must go into the query/invoke JavaScript

files in order to adjust their queries or invocations.

EOF

## Contract files:

### enrollAdmin.js

```
/*
 * SPDX-License-Identifier: Apache-2.0
 */

'use strict';

const FabricCAServices = require('fabric-ca-client');
const { FileSystemWallet, X509WalletMixin } = require('fabric-network');
const fs = require('fs');
const path = require('path');

const ccpPath = path.resolve(__dirname, '..', '..', 'basic-network',
'connection.json');
const ccpJSON = fs.readFileSync(ccpPath, 'utf8');
const ccp = JSON.parse(ccpJSON);

async function main() {
  try {

    // Create a new CA client for interacting with the CA.
    const caURL = ccp.certificateAuthorities['ca.example.com'].url;
    const ca = new FabricCAServices(caURL);

    // Create a new file system based wallet for managing identities.
    const walletPath = path.join(process.cwd(), 'wallet');
    const wallet = new FileSystemWallet(walletPath);
    console.log(`Wallet path: ${walletPath}`);
```

```

// Check to see if we've already enrolled the admin user.
const adminExists = await wallet.exists('admin');
if (adminExists) {
  console.log('An identity for the admin user "admin" already exists in
the wallet');
  return;
}

// Enroll the admin user, and import the new identity into the wallet.
const enrollment = await ca.enroll({ enrollmentID: 'admin',
enrollmentSecret: 'adminpw' });
const identity = X509WalletMixin.createIdentity('Org1MSP',
enrollment.certificate, enrollment.key.toBytes());
wallet.import('admin', identity);
console.log('Successfully enrolled admin user "admin" and imported it
into the wallet');

} catch (error) {
  console.error(`Failed to enroll admin user "admin": ${error}`);
  process.exit(1);
}
}

main();

```

### Invoke.js

```

/*
 * SPDX-License-Identifier: Apache-2.0
 */

'use strict';

const { FileSystemWallet, Gateway } = require('fabric-network');
const fs = require('fs');
const path = require('path');

const ccpPath = path.resolve(__dirname, '..', '..', 'basic-network',
'connection.json');
const ccpJSON = fs.readFileSync(ccpPath, 'utf8');
const ccp = JSON.parse(ccpJSON);

```

```

async function main() {
  try {

    // Create a new file system based wallet for managing identities.
    const walletPath = path.join(process.cwd(), 'wallet');
    const wallet = new FileSystemWallet(walletPath);
    console.log(`Wallet path: ${walletPath}`);

    // Check to see if we've already enrolled the user.
    const userExists = await wallet.exists('user1');
    if (!userExists) {
      console.log('An identity for the user "user1" does not exist in the
wallet');
      console.log('Run the registerUser.js application before retrying');
      return;
    }

    // Create a new gateway for connecting to our peer node.
    const gateway = new Gateway();
    await gateway.connect(ccp, { wallet, identity: 'user1', discovery: {
enabled: false } });

    // Get the network (channel) our contract is deployed to.
    const network = await gateway.getNetwork('mychannel');

    // Get the contract from the network.
    const contract = network.getContract('fabcar');

    // Submit the specified transaction.
    // createRule transaction - requires 5 argument, ex: ('createRule',
'RULE12', 'alert', 'tcp', 'source-port', 'source-ip')
    // changeRuleOwner transaction - requires 2 args , ex: ('Rule', 'RULE10',
'Daniel')
    await contract.submitTransaction('createRule', 'RULE53', 'Dr. Shahriar',
'test', 'test', 'test', 'test', 'test', 'test', 'test', 'test', 'test', 'test');
    console.log('Transaction has been submitted');

    // Disconnect from the gateway.
    await gateway.disconnect();

  } catch (error) {
    console.error(`Failed to submit transaction: ${error}`);
    process.exit(1);
  }
}

```

```
    }  
  }  
  
  main();
```

## Query.js

```
/*  
 * SPDX-License-Identifier: Apache-2.0  
 */  
  
'use strict';  
  
const { FileSystemWallet, Gateway } = require('fabric-network');  
const fs = require('fs');  
const path = require('path');  
  
const ccpPath = path.resolve(__dirname, '..', '..', 'basic-network',  
'connection.json');  
const ccpJSON = fs.readFileSync(ccpPath, 'utf8');  
const ccp = JSON.parse(ccpJSON);  
  
async function main() {  
  try {  
  
    // Create a new file system based wallet for managing identities.  
    const walletPath = path.join(process.cwd(), 'wallet');  
    const wallet = new FileSystemWallet(walletPath);  
    console.log(`Wallet path: ${walletPath}`);  
  
    // Check to see if we've already enrolled the user.  
    const userExists = await wallet.exists('user1');  
    if (!userExists) {  
      console.log('An identity for the user "user1" does not exist in the  
wallet');  
      console.log('Run the registerUser.js application before retrying');  
      return;  
    }  
  
    // Create a new gateway for connecting to our peer node.
```

```

    const gateway = new Gateway();
    await gateway.connect(ccp, { wallet, identity: 'user1', discovery: {
enabled: false } });

    // Get the network (channel) our contract is deployed to.
    const network = await gateway.getNetwork('mychannel');

    // Get the contract from the network.
    const contract = network.getContract('fabcar');

    // Evaluate the specified transaction.
    // queryRule transaction - requires 1 argument, ex: ('queryRule', 'RULE4')
    // queryAllRules transaction - requires no arguments, ex:
('queryAllRules')
    const result = await contract.evaluateTransaction('queryAllRules');
    console.log(`Transaction has been evaluated, result is:
${result.toString()}`);

    } catch (error) {
        console.error(`Failed to evaluate transaction: ${error}`);
        process.exit(1);
    }
}

main();

```

### registerUser.js

```

/*
 * SPDX-License-Identifier: Apache-2.0
 */

'use strict';

const { FileSystemWallet, Gateway, X509WalletMixin } = require('fabric-
network');
const fs = require('fs');
const path = require('path');

```

```

const ccpPath = path.resolve(__dirname, '..', '..', 'basic-network',
'connection.json');
const ccpJSON = fs.readFileSync(ccpPath, 'utf8');
const ccp = JSON.parse(ccpJSON);

async function main() {
  try {

    // Create a new file system based wallet for managing identities.
    const walletPath = path.join(process.cwd(), 'wallet');
    const wallet = new FileSystemWallet(walletPath);
    console.log(`Wallet path: ${walletPath}`);

    // Check to see if we've already enrolled the user.
    const userExists = await wallet.exists('user1');
    if (userExists) {
      console.log('An identity for the user "user1" already exists in the
wallet');
      return;
    }

    // Check to see if we've already enrolled the admin user.
    const adminExists = await wallet.exists('admin');
    if (!adminExists) {
      console.log('An identity for the admin user "admin" does not exist in
the wallet');
      console.log('Run the enrollAdmin.js application before retrying');
      return;
    }

    // Create a new gateway for connecting to our peer node.
    const gateway = new Gateway();
    await gateway.connect(ccp, { wallet, identity: 'admin', discovery: {
enabled: false } });

    // Get the CA client object from the gateway for interacting with the CA.
    const ca = gateway.getClient().getCertificateAuthority();
    const adminIdentity = gateway.getCurrentIdentity();

    // Register the user, enroll the user, and import the new identity into the
wallet.
    const secret = await ca.register({ affiliation: 'org1.department1',
enrollmentID: 'user1', role: 'client' }, adminIdentity);

```

```
    const enrollment = await ca.enroll({ enrollmentID: 'user1',
enrollmentSecret: secret });
    const userIdentity = X509WalletMixin.createIdentity('Org1MSP',
enrollment.certificate, enrollment.key.toBytes());
    wallet.import('user1', userIdentity);
    console.log('Successfully registered and enrolled admin user "user1" and
imported it into the wallet');

    } catch (error) {
        console.error(`Failed to register user "user1": ${error}`);
        process.exit(1);
    }
}

main();
```



## References

- [1] Ranganathan, V.P. et al. 2018. "A Decentralized Marketplace Application on the Ethereum Blockchain.," *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)* (Philadelphia, PA, Oct. 2018), 90–97.
- [2] Kim, J.-T. et al. 2018. "A study on an energy-effective and secure consensus algorithm for private blockchain systems (PoM: Proof of Majority).," *2018 International Conference on Information and Communication Technology Convergence (ICTC)* (Jeju, Oct. 2018), 932–935.
- [3] Xu, J.J. 2016. "Are blockchains immune to all malicious attacks?," *Financial Innovation*. 2, 1 (Dec. 2016). DOI:<https://doi.org/10.1186/s40854-016-0046-5>.
- [4] Sagirlar, G. et al. 2018. "AutoBotCatcher: Blockchain-Based P2P Botnet Detection for the Internet of Things.," *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)* (Philadelphia, PA, Oct. 2018), 1–8.
- [5] Singla, A. and Bertino, E. 2018. "Blockchain-Based PKI Solutions for IoT.," *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)* (Philadelphia, PA, Oct. 2018), 9–15.
- [6] Dannen, C. 2017. "Bridging the Blockchain Knowledge Gap.," *Introducing Ethereum and Solidity*. Apress. 1–20.
- [7] Golomb, T. et al. 2018. "CIoTA: Collaborative Anomaly Detection via Blockchain.," *Proceedings 2018 Workshop on Decentralized IoT Security and Standards* (San Diego, CA, 2018).
- [8] Pop, C. et al. 2018. "Decentralizing the Stock Exchange using Blockchain An Ethereum-based implementation of the Bucharest Stock Exchange.," *2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP)* (Cluj-Napoca, Sep. 2018), 459–466.
- [9] "Hyperledger - Open Source Blockchain Technologies," Hyperledger. [Online]. Available: <https://www.hyperledger.org/>. [Accessed: 20-Feb-2019].
- [10] J. Hong and C.-C. Liu, "Intelligent Electronic Devices With Collaborative Intrusion Detection Systems," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 271–281, Jan. 2019.

- [11] K. A. Al-Utaibi and E.-S. M. El-Alfy, "Intrusion detection taxonomy and data preprocessing mechanisms," *Journal of Intelligent and Fuzzy Systems*, vol. 34, no. 3, pp. 1369–1383, Mar. 2018.
- [12] Xin, W. et al. 2017. "On Scaling and Accelerating Decentralized Private Blockchains.," *2017 IEEE 3rd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing, (HPSC) and IEEE International Conference on Intelligent Data and Security (IDS)* (Beijing, China, May 2017), 267–271.
- [13] Czirkos, Z. and Hosszú, G. 2019. "P2P based intrusion detection," [Encyclopedia of Information Communication Technology](#)
- [14] Ngamsuriyaroj, S. et al. 2018. "Package Delivery System Based on Blockchain Infrastructure.," *2018 Seventh ICT International Student Project Conference (ICT-ISPC)* (Nakhonpathom, Jul. 2018), 1–6.
- [15] L. Junjoewong, S. Sangnapachai, T. Sunetnanta, "ProCircle: A promotion platform using crowdsourcing and web data scraping technique," *2018 Seventh ICT International Student Project Conference (ICT-ISPC)*, pp. 1–5, 2018.
- [16] Malik, S. et al. 2018. "ProductChain: Scalable Blockchain Framework to Support Provenance in Supply Chains.," *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)* (Cambridge, MA, Nov. 2018), 1–10.
- [17]Wanjun, Y. and Yuan, W. 2018. "Research on Network Trading System Using Blockchain Technology.," *2018 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)* (Bangkok, Oct. 2018), 93–97.
- [18] P.-F. Marteau, "Sequence Covering for Efficient Host-Based Intrusion Detection," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 994–1006, Apr. 2019.
- [19]Czirkos, Z. and Hosszú, G. 2013. "Solution for the broadcasting in the Kademlia peer-to-peer overlay.," *Computer Networks*. 57, 8 (Jun. 2013), 1853–1862. DOI:<https://doi.org/10.1016/j.comnet.2013.02.021>.
- [20] "State of the DApps A list of 2,551 blockchain apps for Ethereum, Steem, EOS, and more." [Online]. Available: <https://www.stateofthedapps.com/>. [Accessed: 20-Feb-2019].
- [21]Anceaume, E. et al. 2018. "Sycomore: A Permissionless Distributed Ledger that Self-Adapts to Transactions Demand.," *National Institute of Information and Automation, France*(2018), 1–8.
- [22] Corsi, P. et al. 2019. "TickEth, a Ticketing System built on Ethereum.," *Association for Computing Machinery*. (Apr. 2019).

- [23] Alexopoulos, N. et al. 2018. “Towards Blockchain-Based Collaborative Intrusion Detection Systems.,” *Critical Information Infrastructures Security* (2018), 107–118.
- [24] H. Carmen, “UNDERSTANDING BLOCKCHAIN OPPORTUNITIES AND CHALLENGES.,” *eLearning & Software for Education* . 2018, Vol. 4, p275-283. 9p.,2018.
- [25] Rilee, K. 2018. “Understanding Hyperledger Sawtooth — Proof of Elapsed Time.,” *Medium*.
- [26] Meng, W. et al. 2018. “When Intrusion Detection Meets Blockchain Technology: A Review.,” *IEEE Access*. 6, (2018), 10179–10188.  
DOI:<https://doi.org/10.1109/ACCESS.2018.2799854>.
- [27] Yli-Huumo, J. et al. 2016. Where Is Current Research on Blockchain Technology?—A Systematic Review. *PLOS ONE*. 11, 10 (Oct. 2016), e0163477.  
DOI:<https://doi.org/10.1371/journal.pone.0163477>.
- [28] A. Warzynski and G. Kolaczek, ”Intrusion detection systems vulnerability on adversarial examples,” in 2018 *Innovations in Intelligent Systems and Applications (INISTA)*, Thessaloniki, 2018, pp. 1-4.
- [29] ”Intrusion Detection Systems - Techotopia.” [Online].  
Available: [https://www.techotopia.com/index.php/Intrusion Detection Systems](https://www.techotopia.com/index.php/Intrusion%20Detection%20Systems)  
[Accessed: 04-Mar-2019].
- [30] E. Vasilomanolakis, M. Stahn, C. G. Cordero, and M. Muhlhauser, ”On probe-response attacks in Collaborative Intrusion Detection Systems,” in *2016 IEEE Conference on Communications and Network Security (CNS)*, Philadelphia, PA, 2016, pp. 279–286.
- [31] R. Jin, X. He, and H. Dai, ”Collaborative IDS Configuration: A Two-Layer Game-Theoretic Approach,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 4, pp. 803–815, Dec. 2018.
- [32] E. Ficke, K. M. Schweitzer, R. M. Bateman, and S. Xu, ”Characterizing the Effectiveness of Network-Based Intrusion Detection Systems,” in *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, Los Angeles, CA, 2018, pp. 76–81.
- [33] F. Massicotte and Y. Labiche, ”On the Verification and Validation of Signature-Based, Network Intrusion Detection Systems,” in *2012 IEEE 23rd International Symposium on Software Reliability Engineering*, Dallas, TX, USA, 2012, pp. 61–70.

- [34] G. Vigna, W. Robertson, and D. Balzarotti, "Testing network-based intrusion detection signatures using mutant exploits," in *Proceedings of the 11th ACM conference on Computer and communications security - CCS '04*, Washington DC, USA, 2004, p. 21.
- [35] Accorsi, R., Stocker, T. and MAijller, G. 2013. "On the exploitation of process mining for security audits: the process discovery case.," *ACM Symposium of Applied Computing (SAC)*, Coimbra, Protugal, pp. 1462-1468.
- [36] J. King and L. Williams, "Log your CRUD: design principles for software logging mechanisms," in *Proceedings of the 2014 Symposium and Bootcamp on the Science of Security - HotSoS '14*, Raleigh, North Carolina, 2014, pp. 1-10.
- [37] R. Sekar et al., "Specification-based anomaly detection: a new approach for detecting network intrusions," in *Proceedings of the 9th ACM conference on Computer and communications security - CCS '02*, Washington, DC, USA, 2002, p. 265.
- [38] Mashima D. and Ahamad, M. 2009. "Using identity credential usage logs to detect anomalous service accesses," *Proceedings of the 5th ACM workshop on Digital identity management (DIM)*, Chicago, Illinois, USA, pp. 73–79.
- [39] Liu, Y., Zhang, L. and Guan, Y. 2009. "A distributed data streaming algorithm for network-wide traffic anomaly detection.," *ACM SIGMETRICS Performance Evaluation Review*, 37, 2, pp. 81–82.
- [40] A. de Vries, "Bitcoin's Growing Energy Problem," *Joule, Cell Press*, vol. 2, no. 5, pp. 801–805, May 2018.
- [41] Hyperledger Caliper. (2019). *Architecture*. [online] Available at: [https://hyperledger.github.io/caliper/docs/2\\_Architecture.html](https://hyperledger.github.io/caliper/docs/2_Architecture.html) [Accessed 16 Jun. 2019].