Abstract—Blockchain, as a distributed ledger technology, becomes more and more popular in both industry and academia. Hyperledger Fabric is permissioned blockchain platform hosted by Linux foundation. Fabric has various components such as peer, ordering service, chaincode and state database. The structure of Fabric network is very complicated to provide reliable permissioned blockchain service. Generally, developers must deal with hundreds of parameters to configure a network. That will cause many reasonableness problems in configurations. In this paper, we focus on how to detect reasonableness problems in Fabric configurations. Firstly, we discuss and provide a reasonableness problem knowledge database based on the perspectives of functionality, security and performance. Secondly, we implemented a detect tool for reasonableness check to Fabric. Finally, we collect 108 sample networks as the testing dataset in the experiment. The result shows our tool can help developers to locate reasonableness problems and understand their network better.

Keywords—Permissioned blockchain, network configuration, Hyperledger Fabric

I. INTRODUCTION

In the past few years, blockchain becomes one of the most popular technologies in the world. As the success of Bitcoin provided by Nakamoto [1], many organizations and companies increase their interest in blockchain. Blockchain can be classified into two types: public and permissioned blockchain. Hyperledger Fabric [2] is one of the permissioned blockchain platforms. It’s open sourced and developed by Linux foundation. Fabric consists various components such as peer nodes, clients, ordering service, membership, and Chaincode [3]. Each component has different role for different purpose. The transaction flow contains four main phases, endorsement, ordering, validation and committing. All the components need to be customized and configured before network startup. Developers must deal with hundreds of parameters around all components to bootstrap their customized Fabric network. [4]

Even worse, many of these parameters are correlated with each other.

In this paper, we focus on reasonableness problems of network configuration in Fabric. The reasonableness problems are imperfections that cannot satisfy users’ requirements during network configuring and bootstrapping, which may cause low efficiency, insecurity or even functionality missing. Developers need a lot of experience to avoid these problems before starting up the network. Therefore, we provide a solution and a tool to help the developers to detect reasonableness problems in their network configurations.

II. SOLUTION FOR REASONABLE CHECK AND EXPERIMENTS

For now we haven’t found that there is any related work or tool aiming at reasonableness problems, especially for Fabric yet. So according to our knowledge, experience and discovery about reasonableness issues in Fabric, we design a solution and a tool to check whether there is any reasonableness problem in Fabric network configuration.

A. Reasonableness check patterns

A reasonableness knowledge database is generated for these problems. These reasonableness problems can be categorized into 3 types. Overall, Table I shows all the unreasonable problems. We implement 12 patterns to check these reasonableness problems according to these 3 types: functionality, performance and security. Every pattern contains several rules to check whether there is problem in specific configuration files.

Noticed that these patterns are summarized based on our experience and trials. We choose the most important reasonableness problems after careful consideration. Obviously there will be more other reasonableness problems which we haven’t recognized yet. So this pattern list will be extended in the future.
In our solution, we firstly collect and parse the configuration files of one Fabric network into pre-defined configuration items. The configuration files are `crypto-config.yaml`, `configtx.yaml`, `docker-compose.yaml`, `start.sh` and `scripts.sh`. According the usages of these 5 configuration files, the tool dumps and parses them as the network configurations together.

The dumped configurations will be stored in our additional database and waiting for analysis. We implement rules related with the 12 patterns. The rules are all based on the conditional statement, string matching and regular expression. Once a rule is matched, a reasonableness tag will be generated in the check reports. Finally the check reports contain all the match result of every rules in every pattern. Also we provide a brief introduction and suggestion for this reasonableness problem.

### C. Experiments and result analysis

Some keywords related with Hyperledger Fabric are used to query projects manually from the Github. 108 different sample network configurations has been collected. Then, we execute all the rules in the patterns against the sample network configurations. Table II shows the overall results of every pattern.

Overall, there are totally 504 reasonableness problems detected in these 108 Fabric networks. For detail, the most frequently problem is that TLS [5][6] is turned off in the sample networks. We believe that is because most of networks are not from real project, they are just sample networks for study and development.

Some of problems will cause fatal error that prevent the network from starting. `Yaml` [7] syntax, Docker compose file syntax and component missing are these kinds of reasonableness problems. The results indicate that some developers publish their projects without audit and trial.

There are 26 sample networks that are using Solo as the ordering service. The Solo implementation has been deprecated and may be removed in a future release of Fabric. So developers must pay attention to Solo.

53 sample networks choose LevelDB [8] as the state database. Fabric provides great flexibility in the types of state database, but developers must consider which database should be chosen to satisfy their own requirements. Because state database cannot be modified once the network is running.

## III. DISCUSSION AND CONCLUSION

In this paper, we focus on the reasonableness problems of Hyperledger Fabric framework.

We first discuss and summarize the potential reasonableness problems of Hyperledger Fabric according to different categories, functionality, security and performance. Then we implemented a check tool for reasonableness problems with 12 corresponding patterns. Every pattern contains a set of rules to match related problems. Finally we collect 108 sample network configurations from Internet, and run our tool on these sample networks as the experiment. The result of experiment shows that our tool is useful for checking and locating the reasonableness problems.

However, there is still many work to do about reasonableness check. 1) For now we only focus the static configuration before network running. In future, we will consider to focus on the dynamic network reasonableness check. 2) The experiment data are all just sample or experimental network configurations. We should collect more network configurations from real use cases. 3) All the 12 patterns is dedicated to Hyperledger Fabric, we think there are more common patterns that are appropriate for other permissioned blockchain system.

### B. Implementation and experiments

In our solution, we firstly collect and parse the configuration files of one Fabric network into pre-defined configuration items. The configuration files are `crypto-config.yaml`, `configtx.yaml`, `docker-compose.yaml`, `start.sh` and `scripts.sh`. According the usages of these 5 configuration files, the tool dumps and parses them as the network configurations together.

The dumped configurations will be stored in our additional database and waiting for analysis. We implement rules related with the 12 patterns. The rules are all based on the conditional statement, string matching and regular expression. Once a rule is matched, a reasonableness tag will be generated in the check reports. Finally the check reports contain all the match result of every rules in every pattern. Also we provide a brief introduction and suggestion for this reasonableness problem.

### C. Experiments and result analysis

Some keywords related with Hyperledger Fabric are used to query projects manually from the Github. 108 different sample network configurations has been collected. Then, we execute all the rules in the patterns against the sample network configurations. Table II shows the overall results of every pattern.

Overall, there are totally 504 reasonableness problems detected in these 108 Fabric networks. For detail, the most frequently problem is that TLS [5][6] is turned off in the sample networks. We believe that is because most of networks are not from real project, they are just sample networks for study and development.

Some of problems will cause fatal error that prevent the network from starting. `Yaml` [7] syntax, Docker compose file syntax and component missing are these kinds of reasonableness problems. The results indicate that some developers publish their projects without audit and trial.

There are 26 sample networks that are using Solo as the ordering service. The Solo implementation has been deprecated and may be removed in a future release of Fabric. So developers must pay attention to Solo.

53 sample networks choose LevelDB [8] as the state database. Fabric provides great flexibility in the types of state database, but developers must consider which database should be chosen to satisfy their own requirements. Because state database cannot be modified once the network is running.

### III. DISCUSSION AND CONCLUSION

In this paper, we focus on the reasonableness problems of Hyperledger Fabric framework.

We first discuss and summarize the potential reasonableness problems of Hyperledger Fabric according to different categories, functionality, security and performance. Then we implemented a check tool for reasonableness problems with 12 corresponding patterns. Every pattern contains a set of rules to match related problems. Finally we collect 108 sample network configurations from Internet, and run our tool on these sample networks as the experiment. The result of experiment shows that our tool is useful for checking and locating the reasonableness problems.

However, there is still many work to do about reasonableness check. 1) For now we only focus the static configuration before network running. In future, we will consider to focus on the dynamic network reasonableness check. 2) The experiment data are all just sample or experimental network configurations. We should collect more network configurations from real use cases. 3) All the 12 patterns is dedicated to Hyperledger Fabric, we think there are more common patterns that are appropriate for other permissioned blockchain system.

### TABLE I. REASONABLENESS CHECK PATTERNS

| Category   | Reasonableness problems | Comment                              |
|------------|-------------------------|--------------------------------------|
| Functionality | CouchDB vs LevelDB      | CouchDB vs LevelDB                   |
|            | Inconsistent parameters | Inconsistent configurations between different sources |
|            | Parameter hardcoded     | Hardcoded parameters increase cost of debugging and maintenance |
|            | Component missing       | Configuration integrity              |
|            | Yaml syntax             | -                                   |
|            | Docker compose file syntax | -                                |
| Performance | BlockTime / BlockSize   | Configuration of ordering service    |
|            | Complex chaincode endorsement policy | Too complex, leads to low efficiency |
|            | Simple chaincode endorsement | Too simple, leads to low security    |
|            | TLS on/off              | TLS off leads to low security in data transportation |
| Security    | State database security | State database authentication information missing |
|            | Consensus mechanism     | Solo - None CFT, Kafka - CFT, hard to governance, Raft - CFT |

### TABLE II. PATTERN RESULTS

| Patterns                          | Result count |
|-----------------------------------|--------------|
| State database choice             | 53           |
| Inconsistent parameters           | 0            |
| Parameter hardcoded               | 7            |
| Component missing                 | 3            |
| Yaml syntax                       | 24           |
| Docker compose file syntax        | 12           |
| BlockTime / BlockSize             | 21           |
| Complex chaincode endorsement policy | 2            |
| Simple chaincode endorsement policy | 5            |
| TLS on/off                        | 343          |
| State database security           | 8            |
| Consensus mechanism               | 26           |
REFERENCES

[1] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system[J]. 2008.

[2] Androulaki E, Barger A, Bortnikov V, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains[C]//Proceedings of the Thirteenth EuroSys Conference. ACM, 2018: 30.

[3] Chaincode tutorials. https://hyperledger-fabric.readthedocs.io/en/release-1.4/chaincode.html

[4] Build your first network. https://hyperledger-fabric.readthedocs.io/en/latest/build_network.html

[5] T. Dierks and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2,” IETF RFC 5246, 2008, http://www.ietf.org/rfc/rfc5246.txt.

[6] Securing communication with TLS. https://hyperledger-fabric.readthedocs.io/en/release-1.4/enable_tls.html

[7] Ben-Kiki O, Evans C, Ingerson B. Yaml ain't markup language (yaml™) version 1.1[J]. yaml.org, Tech. Rep, 2005: 23.

[8] Thakkar P, Nathan S, Viswanathan B. Performance benchmarking and optimizing hyperledger fabric blockchain platform[C]//2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). IEEE, 2018: 264-276.