Cybersecurity & Stress Testing Working Group

1. Motivation

In order to keep moving forward, it's essential to know the limits of the infrastructure from a cybersecurity and a performance perspective.

2. Main Goals

- Smart Contracts Vulnerability Analysis
- Network Stress Testing
- Node Scoring
- Perform cybersecurity analysis and evaluations on the infrastructure
- Perform coordinated stress test together with Pegasys

3. Participants

- Tech Lead: Sergio Cerón Figueroa
- Coordinator: Antonio Leal
- Supervisor: Marcos Allende

4. Detailed description

4.1 Smart Contracts Vulnerability Analysis

<u>Background</u>

Smart contracts are the base of blockchain networks as Ethereum and Hyperledger Besu, therefore, their importance in the operation and features of the network make it important to optimize the code to be executed.

The blockchain smart contracts are executed in a decentralized virtual machine and deployed through a transaction on the network, which makes it very difficult to update. In fact, until recently, it was almost impossible to update them. Meaning that if the code was flawed when it was deployed, there was a chance that hackers would take advantage of it. For example, during DAO hacking, 3.6 million ETH was stolen because the initial code had a vulnerability.

In permissioned networks deployed over frameworks such as Hyperledger Besu, contracts play an even bigger role as they are part of the network permission system, a security breach in their code and deployment can compromise all the nodes of the blockchain network.

Once a smart contract is deployed, it is not possible to obtain the source code from which it was created (usually Solidity), making it even more difficult to detect vulnerabilities, there are three types of analysis to detect vulnerabilities: static, dynamic and symbolic.

The following describes what each of these methods consists of:

- **Static analysis:** This method may guarantee full coverage without executing the smart contract and may run fast enough on a reasonable size of code. Usually involves three stages:
    - i. Building an intermediate representation, such as abstract syntax tree or three-address code.
    - ii. Enriching the IR with additional information, using algorithms such as control and dataflow.
    - iii. Vulnerability detection with a database of patterns, which define vulnerability criteria.
- **Dynamic analysis:** This method runs the smart contract and considers only a subset of all execution paths on some input data; leveraging symbolic execution, taint tracking, and fuzzing to discover vulnerabilities.
- **Symbolic analysis:** At a high-level, this method executes the target contract symbolically and produces a) an execution tree representing all possible program states and b) execution paths and constraints. During the symbolic execution, the executed program states and constraints are passed to the security analysis modules when necessary.

Proposal

An initial proposal is to perform a vulnerability analysis of the LACChain permissioning and third-party deployed smart contracts, using open source and commercial tools, such as: Harvey, Maru and Mythril++ tools, to execute a static, dynamic and symbolic analysis. To perform this analysis, it will involve creating and deploying a smart contract monitor to analyze smart contracts in real time.

Another proposal is to conduct a preliminary analysis of smart contract code to be executed on the network by exposing an endpoint to audit them at the same time as deployed with a security seal. This mechanism would involve developing a DAPP as a proxy to analyze, seal and deploy each smart contract in LACChain.

Architecture

Figure 1 shows the proposed solution for vulnerability analysis in smart contracts already deployed in the LACChain network, as well as a possible alternative to scan the contracts before their deployment, issuing a stamp to mark those which pass the security tests.

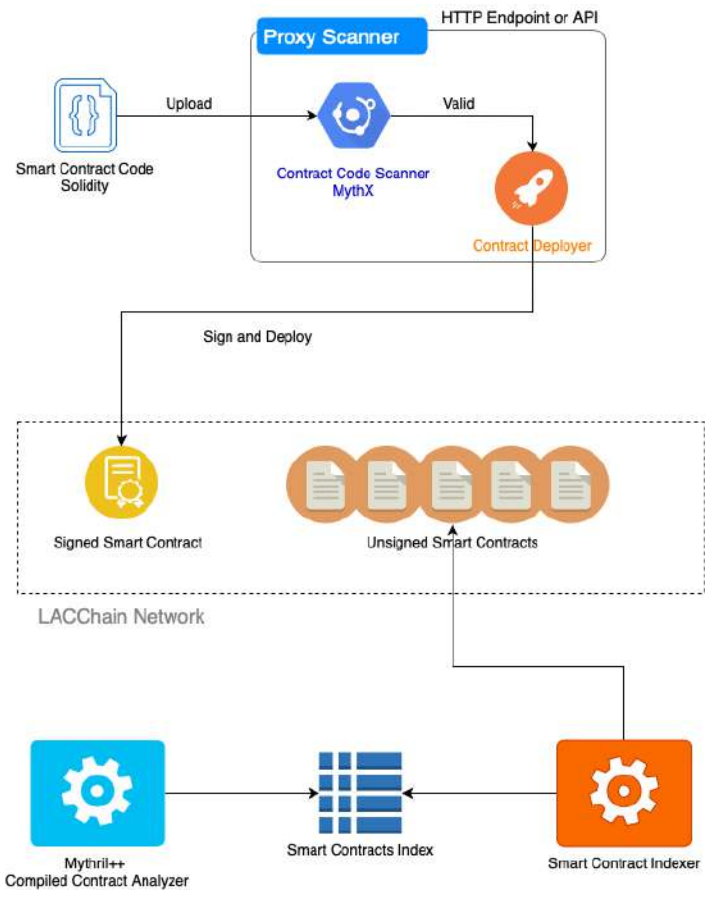

Figure 1. Smart Contracts Vulnerability Analysis

Technical milestones:

- Index currently deployed smart contracts
  - Scan and index deployed smart contracts in LACChain, and store them in a database (MySQL), including as much information as possible.
  - **Deliverable**: Database of currently deployed smart contracts in LACChain
- Analyze currently deployed smart contracts
  - Create a script to scan with Mythril every indexed contract and analyze vulnerabilities.
  - Generate a report of possible bugs in every deployed contract.
  - **Deliverable**: Vulnerabilities report.
- Proxy Scanner
  - Create and endpoint to upload smart contracts and perform a coverage test before deploying in the network, including an approval signature. This analysis will be done using MythX tool from Consensys.

- Create a summary of vulnerabilities and bugs using static, dynamic and symbol analysis
- Create a seal for any contract that passes the tests and stamp before deploying
- **Deliverable**: Web UI to upload smart contract code to analyze, report, stamp and deploy.

4.2 Network Stress Testing

Background

Different types of blockchain networks have their own performance and scalability issues. It is mostly due to the implications a public blockchain has compared to a permissioned blockchain.

Bitcoin is the public blockchain with the most complicated scalability problem, as well the most constrained protocol to modify; the Ethereum network can be implemented as a public blockchain as well as a permissioned blockchain. Also, blockchain types such as Ethereum have different benefits and drawbacks.

Hyperledger Besu is a permissioned blockchain type, for that reason the impact and performance on a public blockchain as well as its scalability compared to a non-permissioned blockchain should be considered. The performance and scalability in Hyperledger Besu are completely different from both Bitcoin and Ethereum since it is a permissioned blockchain. This allows Besu to have different types of nodes with their own responsibilities and allows them to configure a network of nodes that can scale independently of each other.

Proposal

Evaluate the performance and scalability of the LACChain network, examining the scalability issues it will eventually have to face, including Consistency, Availability, and Partition tolerance. Also compare with other types of blockchain networks which are not permissioned networks and do not have the same scalability issues.

Since permissioned networks will not have the same degree of decentralization as a public network would have, the main objective of this proposal is to implement a test environment and run stress tests over a private LACChain network.

After, develop a client to perform a collection of simple tests in the network, which involves coding and installing a chain-code, instantiating accounts, and then send thousands of invoke-requests to the network. The invoke-request means make an asset transfer from account A to account B and execute multiple types of different smart contracts, as well as permissioning on-chain contracts.

Advantages

The main advantage of performing stress tests on a decentralized network is to provide statistics that provide theoretical and empirical support for the stability of the blockchain network, as well as identifying problems in the architecture and topology of the nodes that comprise it.

The following is a list of problems that are expected to be identified after testing:

- **Bottlenecks:** (frequently used parts of the code that require optimization) which adversely affect the network's performance; these occur when a single node significantly reduces a capacity of the entire blockchain.

- **Resource management errors:** For example, absence of synchronization, and consequently, misoperation of the node, as well as data corruption.
- **Configuration errors:** Outline where performance constraints can exist in Hyperledger Besu when tested with specific use cases, and how does different node configuration affect the network's performance.
- **Limitations:** Given the current topology, identify the limits of the consensus and communication protocols within the network.

Architecture

The architecture of the proposal will consist of deploying a private LACChain network, at least with the following resources:

- 4 Subnets
- 3 nodes for each subnet, distributed:
    - 1 network with only validator nodes
    - 1 network with only boot nodes
    - 2 networks with mixed nodes

Each subnet should have a monitor that can deploy more nodes on demand from the master controller. The master controller will receive all the data from each network and determine if it is necessary to deploy more nodes, as well as the tests to be performed.

A coordinator for each subnet will oversee stress tests execution on each node.

Some of the tests are:

- Deploy a Contract
- Send a regular transaction
- Execute a Contract Method
- Change the node's public IP address
- Remove a Peer from the Besu network
- Send a Private Transaction

Figure 2 shows the architecture diagram previously described.

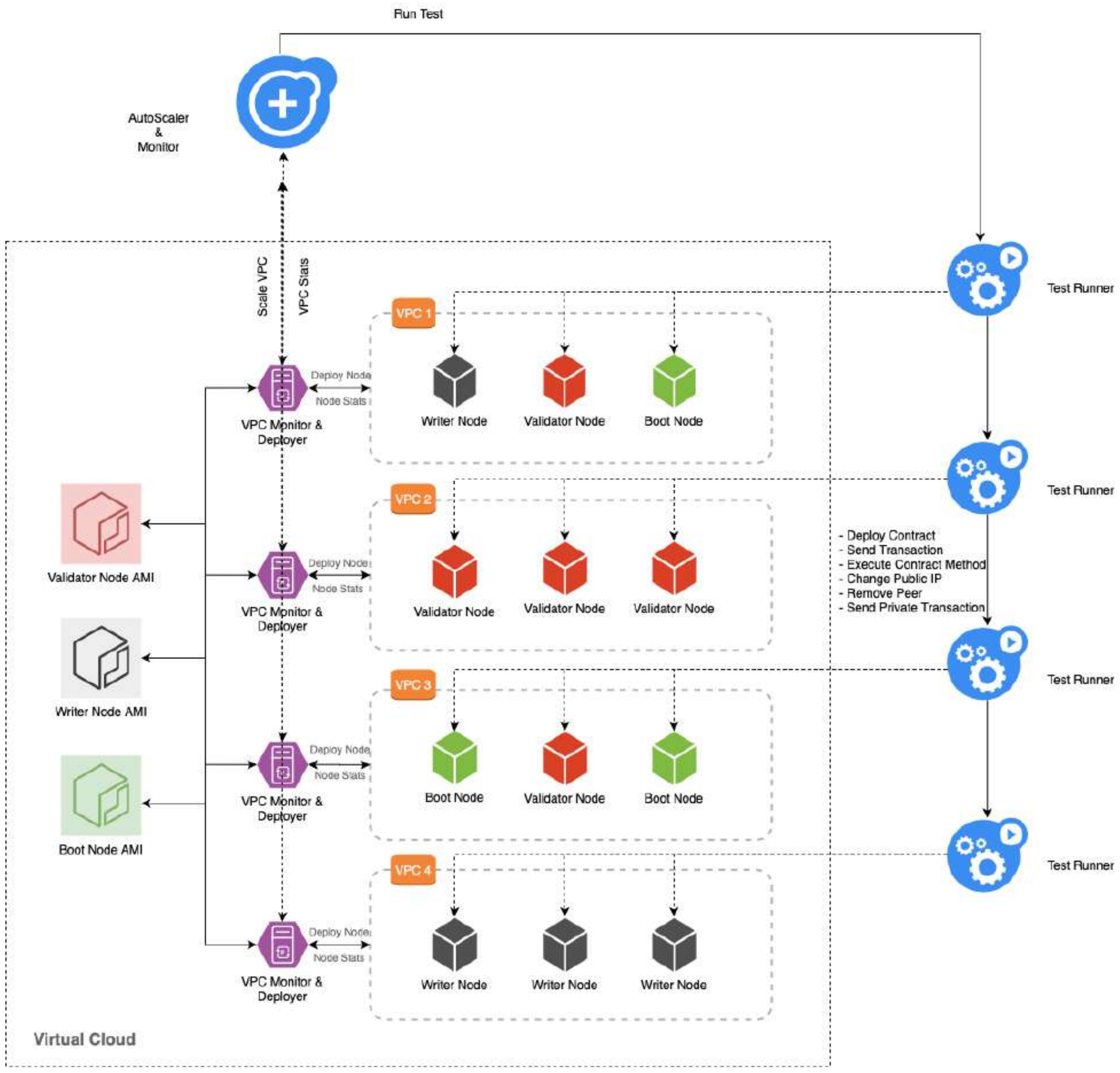

Figure 2. Stress testing architecture

Technical milestones:

- Deploy a virtual private LACChain network
    - Bootstrap current blockchain data of LACChain.
    - Deploy an isolated LACChain network topology considering the minimal requirements of the architecture.
    - **Deliverable**: Isolated private LACChain network.
- Develop a stress testing suite
    - Develop a collection of scripts to test and stress the cloned LACChain network, including:
        - Sending multiple raw transactions
        - Deployment of smart contracts
        - Contract call functions
        - Contracts with large amount of data
        - Contracts with complex loop functions
        - Increasing and reducing nodes (boot, validators, writers)
        - Removing Peers
        - Changing public IP node addresses
    - **Deliverable**: Scripts for every test case.
- Develop a master controller
    - Code a master controller to run scripts in every node and monitor all resources in the network, including CPU, Memory, Traffic, Locks, Errors, Collisions.
    - Generate reports and statistics of use.
    - **Deliverable**: A program to control and orchestrate all stress tests and result reports, including recommendations and enhancements.

4.3 Node Scoring

Background

Hyperledger Besu is a public-permissioned blockchain that is part of the core of the LACChain network and within 3 types of nodes mainly operate: validators, boot, and writers. The validator nodes run the consensus protocol (IBFT 2), its function is to validate the transactions that will be included in the generated blocks.

On the other hand, there are the boot nodes that allow the writer nodes to connect to the network and discover other nodes deployed within the network. Finally, there are the writing nodes, which function as the access point to the network, sending transactions, displaying contracts and consuming information.

On the other hand, Hyperledger Besu blockchain provides both local and on-chain permissioning:

- Local permissioning is done at the node level. To implement it, a permission configuration file is used. As the permissions are local, they do not impact the network. This is useful on how the node works — which are independent of the rest of the network. It is also required to protect nodes if something wrong happens.
- On-chain permissioning, on the other hand, is coded within smart contracts. On-chain permissioning is network-wide, and all nodes can read and update it. On-chain permissioning can only be modified or updated with coordination. Also, once it is updated, it is applied across the network.

On-chain permissioning based on smart contracts is done through a DApp developed by Pegasys at the node and account level. Registering each one in the whitelist requires specifying the enode data, such as: public key, IP address and TCP port. However, more specific data such as the type of node is not included to allow a better categorization of the network topology.

The current topology of the LACChain network allows a certain level of isolation, and is maintained under the assumption that there are no direct connections between writer nodes and validator nodes. This restriction is respected with the relationship of trust that exists between the members of the network, but without any type of technical restriction.

<u>Proposal</u>

The objective of this proposal is to modify the smart permissioning contracts to consider more information in the registration process of each node and account inside the LACChain network. Also, it is intended to use these same contracts to create a classification method based on the type of node and route the network connections, maintaining stricter connections and ensuring the topology.

Because contracts deployed on a blockchain network like Besu are immutable and cannot be updated, it is necessary to change the address of the permissioning contracts with that of the new deployed contracts.

Finally, the permissions rules should be added considering the node type in the permissions list in the new contracts. The added rules must be according to the connection current topology.

Architecture

Figure 3 shows the architecture for the node scoring process.

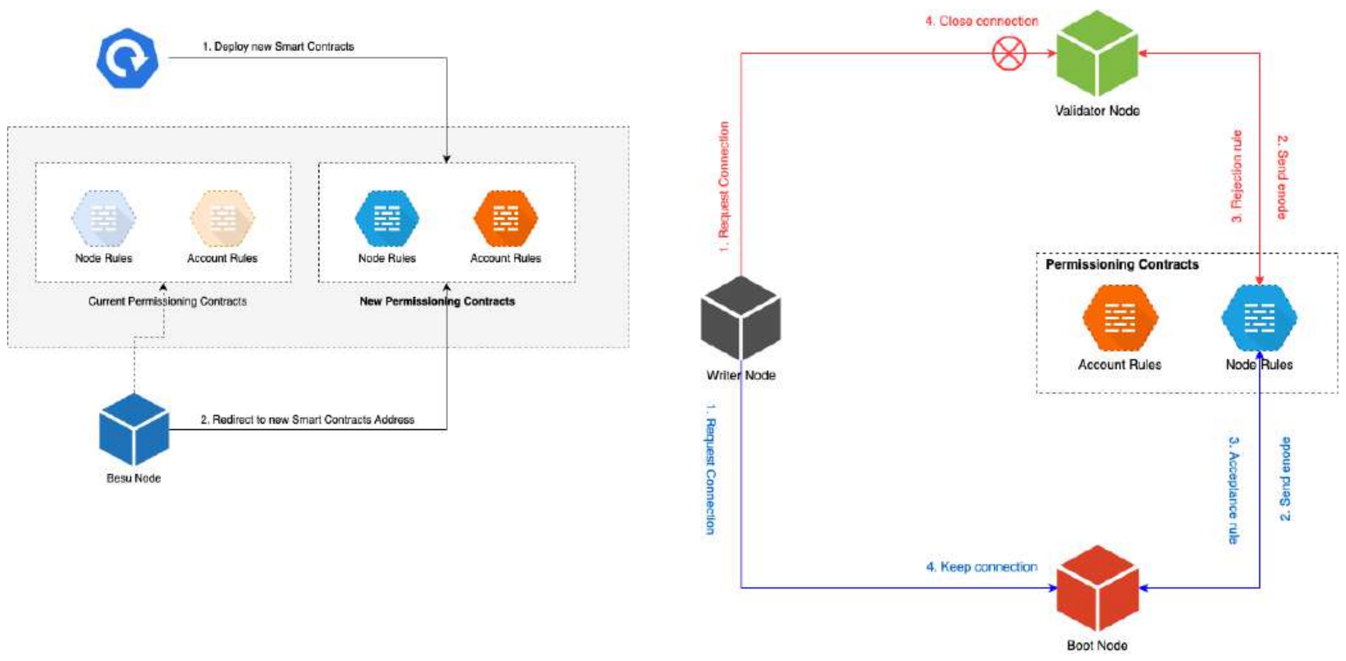

Figure 3. Node scoring architecture.

Technical milestones:

- Modify Permissioning Smart Contracts
    - Modify the current permissioning smart contracts from Pegasys to allow registering more information of nodes and accounts, including the node type and common name instead of IP address in the enode.
    - Deploy modified smart contract for permissioning and redirect all Besu nodes to the new address.
    - Modify permissioning rules to consider the node type for allowing TCP/UDP connections.
    - **Deliverable**: New Permissioning Smart Contracts and its corresponding address.