Cryptonics

Sifchain - Continuous Audit Report

July 6, 2021

Cryptonics Consulting S.L.
Ramiro de Maeztu 7
46022 Valencia
SPAIN

https://cryptonics.consulting/

Table of Contents

Disclaimer

THE CONTENT OF THIS AUDIT REPORT IS PROVIDED “AS IS”, WITHOUT REPRESENTATIONS AND WARRANTIES OF ANY KIND.

THE AUTHORS AND THEIR EMPLOYER DISCLAIM ANY LIABILITY FOR DAMAGE ARISING OUT OF, OR IN CONNECTION WITH, THIS AUDIT REPORT.

THIS AUDIT REPORT IS NOT A SECURITY WARRANTY, INVESTMENT ADVICE, OR AN ENDORSEMENT OF THE CLIENT OR ITS PRODUCTS. THIS AUDIT DOES NOT PROVIDE A SECURITY OR CORRECTNESS GUARANTEE OF THE AUDITED SOFTWARE.

Introduction

Purpose of this Report

Cryptonics Consulting has been engaged to perform a continuous security audit of Sifchain. This current audit report covers the migration of the Sifchain **Blockchain node software** to version 0.42 of Cosmos SDK.

The objectives of the audit are as follows:

1. Determine the correct functioning of the system, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the code submitted in the following branch in a public GitHub repository:

https://github.com/Sifchain/sifnode/tree/feature/cosmos-0.42

with the latest commit no: `9597894b9628ea5eebf41fd2ef7c7533f4022be2`

The files in scope for the audit process:

Methodology

The audit has been performed by a mixed team of smart contract and full-stack auditors.

The following steps were performed:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line by line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under- / overflow issues
    c. Key management vulnerabilities
    d. Permissioning issues
    e. Logic errors
4. Report preparation

The results were then discussed between the auditors in a consensus meeting and integrated into this joint report.

Functionality Overview

The submitted code implements the Sifchain blockchain node software, including the native Continuous Liquidity Pool implementation and the Peggy Ethereum bridge.

How to read this Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending**, **Acknowledged** or **Resolved**. Informational notes do not have a status, since we consider them optional recommendations.

Note, that audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria for each module, in the corresponding findings section.

Note, that high complexity or lower test coverage generally equates to a higher remaining risk. Test coverage is of particular importance since certain bugs are more easily detected in unit testing than a security audit and vice versa.

Notes on Architecture / Functionality

Sifchain CLP

Apart from the code implementation security review, the audit team has performed a review of the currently supported functionality and architecture.

The Sifchain CLP architecture builds directly on Cosmos SDK (https://cosmos.network/sdk) and uses the Cosmos-native primitives and the same terminology. The Cosmos SDK version used in the audited code was `v0.42`.

Liquidity pools in Sifchain provide liquidity for pairings of external assets to Sifchain's native token.

IMPORTANT NOTICE: The audit does not evaluate the merits and validity of the economic model. It merely verifies that the implementation matches the specification.

Peggy Ethereum Bridge

Apart from the code implementation security review, the audit team has performed a review of the currently supported functionality and architecture. The following a number of informal observations:

- **Ethereum Transaction Relaying:**
  The relayer currently does not relay data from transactions directly from Ethereum to the peg zone. Instead, it subscribes to events emitted through the smart contracts. This is an efficient way to deal with this but may lead to inconsistencies if events are missed.
- **Finality Considerations: (UPDATE: Resolved)**
  There is also currently no evidence of a solution to the finality mismatch between the Ethereum and Cosmos chain. Tendermint consensus used by Cosmos provides instant finality through a byzantine voting protocol. Ethereum, on the other hand, provides probabilistic consensus, which means that blocks can become state and transactions may be rolled back retrospectively initially.
  Peg zones usually act as a "finality gadget buffer" in this. Whilst it is clear that the finality gadget is intended for future versions, the current design does not lend itself to implement this without major refactoring.

  UPDATE: The team has introduced a 30-block delay before submitting events to Cosmos chain to address this issue

- **Supported Assets:**

Currently, ETH and ERC-20 tokens are supported. Whilst focusing on ERC-20 assets is a good choice, care must be taken with malicious implementations, copycat tokens, and compatible standards, such as ERC-777 tokens that may inject arbitrary code through callbacks.

Findings

Summary

The Sifchain node software was found to contain the issues summarized in the following table and detailed below:

No	Description	Severity	Status
1	parser.go: Incorrect prefix comparison in function EthereumEventToEthBridgeClaim()	Minor	Pending
2	x/clp/keeper/Calculations.go: Token decimal adjustment is always executed	Minor	Pending
3	Outdated indirect dependency with multiple known security vulnerabilities	Minor	Pending
4	Consider using a mock address for ETH	Informational	-
5	Node package inclusion in Solidity files breaks build system	Informational	-
6	Provide formal documentation on why the number 30 Ethereum blocks were chosen as safe enough finality for Sifchain	Informational	-
7	Missing default case in switch statement	Informational	-

Code Quality Criteria

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	High	-
Level of Documentation	High	-
Test Coverage	Medium	-

Detailed Findings

1. parser.go: Incorrect prefix comparison in function EthereumEventToEthBridgeClaim()

Severity: Minor

On line 56, function `EthereumEventToEthBridgeClaim()` only checks that symbol contains `defaultEthereumPrefix`, but does not actually verify that it starts with it.

Recommendation

Replace `strings.Contains()` with `strings.HasPrefix()`.

Status: Pending

2. `x/clp/keeper/Calculations.go`: Token decimal adjustment is always executed

Severity: Minor

In functions `calcLiquidityFee()` and `calcSwapResult()`, there are checks to see if the number of decimals of external tokens needs to be normalized. However, the actual adjustment code is executed in either case, due to the following incorrect if statement:

```
if adjustExternalToken {
                if toRowan {
                        X = X.Mul(sdk.NewUintFromBigInt(normalizationFactor.RoundInt().BigInt()))
                        x = x.Mul(sdk.NewUintFromBigInt(normalizationFactor.RoundInt().BigInt()))
                } else {
                        Y = Y.Mul(sdk.NewUintFromBigInt(normalizationFactor.RoundInt().BigInt()))
                }
        } else {
                if toRowan {
                        X = X.Mul(sdk.NewUintFromBigInt(normalizationFactor.RoundInt().BigInt()))
                        x = x.Mul(sdk.NewUintFromBigInt(normalizationFactor.RoundInt().BigInt()))
                } else {
                        Y = Y.Mul(sdk.NewUintFromBigInt(normalizationFactor.RoundInt().BigInt()))
                }
        }
```

Recommendation

Remove the else statement.

Status: Pending

3. Outdated indirect dependency with multiple known security vulnerabilities

Severity: Minor

The codebase uses an outdated version of the `ecdt` library (versions 3.3.13). This version includes a number of known vulnerabilities, including TLS authentication issues, TCP proxy discovery issues, and missing password length verifications.

Recommendation

Update the dependency.

Status: Pending

4. Consider using a mock address for ETH

Severity: Informational

In the contracts, for referencing an ETH value transfer the token address is set to 0x0, however, this is already the default value mappings. While there's no security implication as is, a better option would be to choose a mock address for ETH, like `0xeeeee....` to avoid any possible confusion with uninitialized addresses.

Recommendation

Change the ETH token address

5. Node package inclusion in Solidity files breaks build system

Severity: Informational

The smart contract course files include OpenZeppelin dependencies imported as npm packages with relative path names. For example:

```
import "../../node_modules/openzeppelin-solidity/contracts/math/SafeMath.sol";
```

This breaks the functionality of most up-to-date development frameworks, including Truffle used in the project, leading to the project not currently building on a clean installation.

Recommendation

Import npm dependencies directly, since they are discoverable from Truffle:

```
import "openzeppelin-solidity/contracts/math/SafeMath.sol";
```

6. Provide formal documentation on why the number 30 Ethereum blocks were chosen as safe enough finality for Sifchain

Severity: Informational

Provide more explicit documentation of the security assumptions that made Sifchain arrive at the fact that 30 blocks are safe finality for the purposes of the Sifchain application.

7. Missing default case in switch statement

Severity: Informational

The switch statement for processing Ethereum event data in `cmd/ebrelayer/relayer/ethereum.go:158-168` is lacking a default case. It is always recommended to add default cases for clarity and dealing with unexpected data.

Recommendation

Add a default case.

Appendix A: Files in scope

```
app
├── app.go
├── app_test.go
├── encoding.go
├── export.go
├── genesis.go
├── poolChange.go
├── prefix.go
├── release-20210501000000.go
├── setupHandlers.go
└── test_helpers.go
cmd
├── ebrelayer
│   ├── Dockerfile
│   ├── contract
│   │   ├── abi.go
│   │   └── main.go
│   ├── relayer
│   │   ├── cosmos.go
│   │   ├── cosmos_test.go
│   │   ├── ethereum.go
│   │   ├── ethereum_test.go
│   │   ├── listMissedEvent.go
│   │   ├── network.go
│   │   └── network_test.go
│   ├── replay.go
│   ├── txs
│   │   ├── parser.go
│   │   ├── parser_test.go
│   │   ├── registry.go
│   │   ├── registry_test.go
│   │   ├── relayToCosmos.go
│   │   ├── relayToEthereum.go
│   │   ├── signature.go
│   │   ├── signature_test.go
│   │   └── test_common.go
│   └── types
│       ├── events.go
│       └── types.go
├── sifcrg
│   ├── Dockerfile
│   └── main.go
├── sifgen
│   └── main.go
└── sifnoded
    ├── Dockerfile
    ├── cmd
    │   ├── add_genesis_validator_test.go
    │   ├── clpadmin.go
    │   ├── genaccounts.go
    │   ├── gentx.go
    │   ├── migrate.go
    │   ├── migrate_test.go
    │   ├── oracle.go
    │   ├── root.go
    │   └── testdata
    │       └── v039_exported_migrated_state.json
    └── main.go

x
├── clp
│   ├── README.md
│   ├── client
│   │   ├── cli
│   │   │   ├── flags.go
│   │   │   ├── query.go
│   │   │   ├── tx.go
```

```
│   │   │   └── tx_test.go
│   │   └── rest
│   │       ├── query.go
│   │       ├── rest.go
│   │       └── tx.go
│   ├── clp_suite_test.go
│   ├── genesis.go
│   ├── genesis_test.go
│   ├── handler.go
│   ├── handler_test.go
│   ├── keeper
│   │   ├── calculations.go
│   │   ├── common_test.go
│   │   ├── executors.go
│   │   ├── grpc_query.go
│   │   ├── keeper.go
│   │   ├── keeper_test.go
│   │   ├── liquidityprovider.go
│   │   ├── msg_server.go
│   │   ├── params.go
│   │   ├── pool.go
│   │   ├── pureCalculation.go
│   │   ├── querier.go
│   │   ├── querier_test.go
│   │   ├── table_test.go
│   │   ├── validatorWhiteList_test.go
│   │   └── validatorWhitelist.go
│   ├── legacy
│   │   ├── v39
│   │   │   └── genesis.go
│   │   └── v42
│   │       └── migrate.go
│   ├── module.go
│   ├── test
│   │   └── test_common.go
│   └── types
│       ├── asset.go
│       ├── codec.go
│       ├── errors.go
│       ├── events.go
│       ├── expected_keepers.go
│       ├── genesis.go
│       ├── genesis.pb.go
│       ├── keys.go
│       ├── keys_test.go
│       ├── msgs.go
│       ├── msgs_test.go
│       ├── params.go
│       ├── params.pb.go
│       ├── querier.go
│       ├── querier.pb.go
│       ├── querier.pb.gw.go
│       ├── tx.pb.go
│       ├── types.go
│       └── types.pb.go
├── dispensation
│   ├── Flow-Claim.md
│   ├── Flow-Distribute.md
│   ├── Readme-Claim.md
│   ├── Readme-Distrubute.md
│   ├── Steps-AuthorizedRunner.md
│   ├── Steps-Claimer.md
│   ├── Steps-Investor.md
│   ├── UserFlow-Claim.md
│   ├── UserFlow-Distribute.md
│   ├── alias.go
│   ├── claims.md
│   ├── client
│   │   ├── cli
│   │   │   ├── query.go
│   │   │   └── tx.go
│   │   └── rest
│   │       └── tx.go
│   ├── genesis.go
│   ├── genesis_test.go
│   ├── handler.go
│   ├── handler_test.go
```

```
│   ├── keeper
│   │   ├── distribution.go
│   │   ├── distributionRecords.go
│   │   ├── distributionRecords_test.go
│   │   ├── distributions_test.go
│   │   ├── executors.go
│   │   ├── executors_test.go
│   │   ├── grpc_querier.go
│   │   ├── keeper.go
│   │   ├── keeper_test.go
│   │   ├── msg_server.go
│   │   ├── querier.go
│   │   ├── querier_test.go
│   │   ├── upgrade.go
│   │   ├── upgrade_test.go
│   │   ├── userclaim.go
│   │   └── userclaim_test.go
│   ├── module.go
│   ├── test
│   │   └── test_common.go
│   ├── types
│   │   ├── codec.go
│   │   ├── errors.go
│   │   ├── events.go
│   │   ├── expected_keepers.go
│   │   ├── genesis.go
│   │   ├── keys.go
│   │   ├── legacy
│   │   │   └── types.go
│   │   ├── msgs.go
│   │   ├── msgs_test.go
│   │   ├── querier.go
│   │   ├── query.pb.go
│   │   ├── records.go
│   │   ├── tx.pb.go
│   │   ├── types.go
│   │   └── types.pb.go
│   └── utils
│       ├── parser.go
│       └── parser_test.go
├── ethbridge
│   ├── alias.go
│   ├── client
│   │   ├── cli
│   │   │   ├── query.go
│   │   │   └── tx.go
│   │   ├── module_client.go
│   │   └── rest
│   │       └── rest.go
│   ├── genesis.go
│   ├── genesis_test.go
│   ├── handler.go
│   ├── handler_test.go
│   ├── keeper
│   │   ├── cethReceiverAccount.go
│   │   ├── cethReceiverAccount_test.go
│   │   ├── grpc_query.go
│   │   ├── keeper.go
│   │   ├── keeper_test.go
│   │   ├── msg_server.go
│   │   ├── peggyTokens.go
│   │   ├── querier.go
│   │   └── querier_test.go
│   ├── legacy
│   │   ├── v39
│   │   │   └── genesis.go
│   │   └── v42
│   │       └── migrate.go
│   ├── module.go
│   ├── test
│   │   └── test_helpers.go
│   └── types
│       ├── claim.go
│       ├── codec.go
│       ├── errors.go
│       ├── ethereum.go
│       ├── events.go
```

```
│   ├── expected_keepers.go
│   ├── flags.go
│   ├── keys.go
│   ├── msgs.go
│   ├── querier.go
│   ├── query.pb.go
│   ├── test_common.go
│   ├── tx.pb.go
│   ├── types.go
│   └── types.pb.go
└── oracle
    ├── genesis.go
    ├── genesis_test.go
    ├── keeper
    │   ├── adminAccount.go
    │   ├── adminAccount_test.go
    │   ├── keeper.go
    │   ├── keeper_test.go
    │   ├── validatorWhiteList.go
    │   └── validatorWhiteList_test.go
    ├── legacy
    │   ├── v39
    │   │   └── genesis.go
    │   └── v42
    │       └── genesis.go
    ├── module.go
    └── types
        ├── claim.go
        ├── codec.go
        ├── errors.go
        ├── expected_keepers.go
        ├── genesis.go
        ├── keys.go
        ├── prophecy.go
        └── types.pb.go

proto
└── sifnode
    ├── clp
    │   └── v1
    │       ├── genesis.proto
    │       ├── params.proto
    │       ├── querier.proto
    │       ├── tx.proto
    │       └── types.proto
    ├── dispensation
    │   └── v1
    │       ├── query.proto
    │       ├── tx.proto
    │       └── types.proto
    ├── ethbridge
    │   └── v1
    │       ├── query.proto
    │       ├── tx.proto
    │       └── types.proto
    └── oracle
        └── v1
            └── types.proto
```