CERTIK

Certik Report for FirmaChain

Contents

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Verification Services Agreement between CertiK and FirmaChain (the “Company”), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, the “Agreement”). This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK’s prior written consent.

About CertiK

CertiK is a technology-led blockchain security company founded by Computer Science professors from Yale University and Columbia University built to prove the security and correctness of smart contracts and blockchain protocols.

CertiK, in partnership with grants from IBM and the Ethereum Foundation, has developed a proprietary Formal Verification technology to apply rigorous and complete mathematical reasoning against code. This process ensures algorithms, protocols, and business functionalities are secured and working as intended across all platforms.

CertiK differs from traditional testing approaches by employing Formal Verification to mathematically prove blockchain ecosystem and smart contracts are hacker-resistant and bug-free. CertiK uses this industry-leading technology together with standardized test suites, static analysis, and expert manual review to create a full-stack solution for our partners across the blockchain world to secure 6.2B in assets. For more information: https://certik.org.

Executive Summary

FirmaChain is an electronic contract platform built on the blockchain, leveraging the Cosmos SDK. FirmaChain delivers a unique solution to the existing problems in the electronic contract market, and fundamentally improves work efficiency for the end users.

To that end, the sole objective of the audit is to verify FirmaChain's implementation of the auth and contract modules against the specifications, and discover potential issues and vulnerabilities in the codebase. The audit was conducted through a series of thorough security assessments, the goal of which is to help the said project protect their users by finding and fixing known vulnerabilities that could cause unauthorized access, loss of funds, cascading failures, and/or other vulnerabilities. Alongside each security finding, recommendations on fixes and best practices have also been given.

Testing Summary

SECURITY LEVEL

This report has been prepared as a product of an audit request by FirmaChain. The audit was conducted to discover issues and vulnerabilities in the project's source code.

TYPE	Chain Audit
SOURCE CODE	https://github.com/FirmaChain/FirmaChain/tree/ae8ef4b32e0f500357ab73b98d281daba5c6c30c
PLATFORM	Cosmos SDK v0.37.4
LANGUAGE	Golang
REQUEST DATE	June xx, 2020
REVISION DATE	July 06, 2020
METHODS	Dynamic Analysis, Static Analysis, and Manual Review

Review Notes

Scope of Work

- The audit work was strictly scoped to a specific commit of the source code per the agreement
- Modules within the scope include: auth module; contract module
- State transitions in each module were carefully verified against their specification
- Go programming best practices were enforced to improve general performance and minimize the chances of run-time panicking

Audit Approach

Our audit approach revolves around ensuring that the security model in FirmaChain is done in a secure and functionally correct manner so that it aids the encapsulation of the modules of the blockchain and helps safeguard the application against unintentional state changes. Apart from assessing the security model, best practices in Go programming will also be applied. The practices include:

- Correct simulation implementation for fuzzy testing to avoid incorrect assumptions
- Secure module inter-dependency instantiation on a need-to-know basis
- Proper and meaningful definition of application invariants

Following the unique structural properties and security models of a Cosmos SDK application, our audit approach largely favors modularity and encapsulation in code design. At a high level we analyze each object by their interfaces and references to other objects. This ultimately ensures that the same security properties can be extended to new objects added to the system, which in return minimizes the attack surface of the application down to the implementation of specific objects.

Audit Revision

The FirmaChain team took our Exhibits into account and decided to proceed optimizing their codebase according to our recommendations. After multiple commits, a second tree branch was assessed accessible at the following link:

- FirmaChain/FirmaChain:v0.1.0 [0b755de3b10553d9f9921c793194d52ce9e7be47]

The changes to the codebase were evaluated and represented in the Exhibits that follow wherever applicable.

Findings

Exhibit 1

TITLE	TYPE	SEVERITY	LOCATION
Incomplete Implementation of `AppModule`	Ineffectual Code	Informational	Reference

[INFORMATIONAL] Description:

The `AppModule` interface for module `auth` has not been fully implemented. It's extremely important to make sure each module correctly implements the `AppModule` interface, with a reference to `AppModuleBasic` embedded in it.

Revision:

`x/auth/module.go` was removed in 0668bf6. We consider this exhibit has been fully attended to.

Exhibit 2

TITLE	TYPE	SEVERITY	LOCATION
Missing Store/Module Name in REST Route	Ineffectual Code	Informational	Reference

[INFORMATIONAL] Description:

The url for a HTTP request should always start with the name of the module.

Revision:

Query handler function estimateGasHandler was replaced with QueryEstimateGasHandlerFn in 0668bf6. We recommend prefixing the rest route /txs/estimate_gas with a store name, but it doesn't impose any security threats nonetheless. We consider this exhibit has been fully attended to.

Exhibit 3

TITLE	TYPE	SEVERITY	LOCATION
Missing Sanity Check on `BaseReq`	Ineffectual Code	Informational	Reference

[INFORMATIONAL] Description:

After parsing the request, it's recommended to run `Sanitize` and `ValidateBasic` on the underlying `BaseReq` to check the validity of the request.

Revision:

This exhibit was fixed in 3ae1575 and hence we consider the issue raised originally was addressed in full.

Exhibit 4

TITLE	TYPE	SEVERITY	LOCATION
Improper REST Response Post Processing	Ineffectual Code	Informational	Reference

[INFORMATIONAL] Description:

For handling POST requests, create a response message and pass it to `WriteGenerateStdTxResponse` for further request processing.

Revision:

This exhibit was not addressed in the fixes. Considering the original finding does not impose any substantial security threats, we conclude the exhibit has been fully attended to.

Exhibit 5

TITLE	TYPE	SEVERITY	LOCATION
Missing Specifier in REST Route	Ineffectual Code	Informational	Reference

[INFORMATIONAL] Description:

A specifier for adding contracts should be appended to the url of the HTTP request.

Revision:

Handler function for adding contracts was replaced with AddContractHandlerFn in 0668bf6. We recommend adding a specifier for adding contracts to the url of the HTTP request, but it doesn't impose any security threats nonetheless. We consider this exhibit has been fully attended to.

Exhibit 6

TITLE	TYPE	SEVERITY	LOCATION
Unconventional Naming in CLI Query Command	Ineffectual Code	Informational	Reference

[INFORMATIONAL] Description:

Query command getter functions should be prefixed with `GetCmd` and include the name of the command.

Revision:

GetContract was removed from x/contract/client/cli/query.go in 3ae1575. We consider this exhibit has been fully attended to.

Exhibit 7

TITLE	TYPE	SEVERITY	LOCATION
Improper Handler Implementation	Ineffectual Code	Informational	Reference

[INFORMATIONAL] Description:

- Before calling the keeper's setter function to actually perform the state transition, make sure stateful validity checks are performed on the message. For instance, in a transfer message you'd want to check the sending account has enough funds to actually perform the transfer
- Before returning, the handler function generally needs to emit one or multiple events via the EventManager held in the ctx

Revision:
This exhibit was fixed in f0341ab and hence we consider the issue raised originally addressed in full.

Exhibit 8

TITLE	TYPE	SEVERITY	LOCATION
Improper Custom Error Registration	Ineffectual Code	Informational	Reference

[INFORMATIONAL] Description:

Consider registering custom errors with sdkerrors.Register , instead of creating individual methods.

Revision:

This exhibit was not addressed in the fixes. Considering the original finding does not impose any substantial security threats, we conclude the exhibit has been fully attended to.