SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

AUDIT

SECURITY ASSESSMENT

10. May, 2024

FOR

AXIOME

SolidProof_io

@solidproof_io

Introduction

SolidProof.io is a brand of the officially registered company FutureVisions Deutschland, based in Germany. We mainly focus on Blockchain Security, such as Smart Contract Audits and KYC verification for project teams.
Solidproof.io assesses potential security issues in the smart contracts implementations, reviews for inconsistencies between the code base and the whitepaper/documentation, and provides suggestions for improvement.

Disclaimer

SolidProof.io reports are not, nor should they be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should they be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io does not cover testing or auditing the integration with external contracts or services (such as Unicrypt, Uniswap, PancakeSwap, etc.).

SolidProof.io Audits does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed or any indication of the technology proprietors. SolidProof Audits should not be used to make decisions about investment or involvement with any particular project. These reports do not provide investment advice, nor should they be leveraged as investment advice.

SolidProof.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof’s position is that each company and individual is responsible for their own due diligence and continuous security. SolidProof does not claim any guarantee of the security or functionality of the technology we agree to analyze.

Project Overview

Summary

Project Name	Axiome
Website	https://axiome.pro
About the project	Axiome is a DeFi ecosystem built on its own L1 solution, Axiome Chain, powered by the unique AXM coin. Our mission is to build a scalable ecosystem of modern DeFi solutions, allowing AXM holders to continuously increase their staking rewards.
Chain	Axiome, powered by Cosmos SDK
Language	Go
Codebase Link	https://github.com/axiome-pro/axm-node
Commit	645184e
Unit Tests	Provided

Social Medias

Telegram	https://t.me/axiomeen
Twitter	https://twitter.com/axiome_pro
Facebook	N/A
Instagram	N/A
Github	https://github.com/axiome-pro
Reddit	N/A
Medium	N/A
Discord	N/A
Youtube	https://www.youtube.com/@axiome_ru
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Changelog
v1.0	10. May 2024	• Layout Project • Automated- /Manual-Security Testing • Summary

This audit report provides a detailed security analysis of the Go codebase used in the project, particularly focusing on potential vulnerabilities to external malicious interference with the program's functions. This analysis did not cover functional testing (or unit testing) of the program's logic. Therefore, we cannot assure complete logical correctness of the code, as we did not perform functional tests on it. This includes the internal calculations in the algorithms implemented in the codebase.

Scope of Work

We aim to conduct a comprehensive security assessment of the provided repository, including a fork of the Cosmos SDK and custom modifications made by the Axiome team. This assessment aims to identify and mitigate potential security vulnerabilities, ensure code integrity, and verify the robustness of modifications to support secure, reliable, and efficient operations.

The repository consists of:

- The original Cosmos SDK source code
- Custom modifications and enhancements made to the SDK
- Integration points and dependencies with external systems or libraries

Repository	Commit
https://github.com/axiome-pro/axm-node	645184e

Imported packages

Used code from other Frameworks. More are imported indirectly.

- cosmossdk.io/api v0.7.2
- cosmossdk.io/client/v2 v2.0.0-beta.1
- cosmossdk.io/collections v0.4.0
- cosmossdk.io/core v0.11.0
- cosmossdk.io/depinject v1.0.0-alpha.4
- cosmossdk.io/errors v1.0.1
- cosmossdk.io/log v1.3.0
- cosmossdk.io/math v1.2.0
- cosmossdk.io/store v1.0.2
- cosmossdk.io/tools/confix v0.1.1
- cosmossdk.io/x/upgrade v0.1.1
- github.com/bits-and-blooms/bitset v1.8.0
- github.com/cockroachdb/errors v1.11.1
- github.com/cometbft/cometbft v0.38.5
- github.com/cosmos/cosmos-db v1.0.0
- github.com/cosmos/cosmos-proto v1.0.0-beta.3
- github.com/cosmos/cosmos-sdk v0.50.3
- github.com/cosmos/go-bip39 v1.0.0
- github.com/cosmos/gogoproto v1.4.11
- github.com/gogo/protobuf v1.3.2
- github.com/golang/mock v1.6.0
- github.com/golang/protobuf v1.5.3
- github.com/gorilla/mux v1.8.0
- github.com/grpc-ecosystem/grpc-gateway v1.16.0
- github.com/hashicorp/go-metrics v0.5.2
- github.com/pkg/errors v0.9.1
- github.com/spf13/cobra v1.8.0
- github.com/spf13/pflag v1.0.5
- github.com/spf13/viper v1.18.2
- github.com/stretchr/testify v1.8.4
- golang.org/x/exp v0.0.0-20231006140011-7918f672742d
- google.golang.org/genproto/googleapis/api v0.0.0-20231120223509-83a465c0220f
- google.golang.org/grpc v1.60.1
- google.golang.org/protobuf v1.32.0
- gopkg.in/yaml.v3 v3.0.1
- gotest.tools/v3 v3.5.1

Note for Investors: We have only audited the Go dependencies listed in the above scope. We have not reviewed any additional dependencies related to the project that are not included in our audit scope, and we cannot comment on their security. We are not responsible for any security issues arising from these unreviewed dependencies.

Audit Information

Vulnerability & Risk Level

Risk represents the probability that a certain source threat will exploit vulnerability and the impact of that event on the organization or system. The risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon aspossible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the audit of the Cosmos SDK Layer 1 blockchain repository, meticulous attention was dedicated to identifying security vulnerabilities, ensuring code quality, and verifying adherence to both specifications and best practices. Our audit was conducted by a seasoned team of penetration testers and blockchain developers with extensive experience in the Cosmos ecosystem and smart contract security.

Each file within the repository was subjected to a thorough manual examination. While automated tools were employed, their usage was strategically limited to augment the efficiency and effectiveness of the manual review process rather than replace it.

Methodology

The auditing process follows a routine series of steps:

- Specification Review: Our team meticulously reviewed all relevant documentation, specifications, and guidelines provided initially. This ensured a profound understanding of the blockchain's architecture, scope, and intended functionalities.

- Manual Code Inspection: The source code was examined line by line. This intensive review aimed to uncover potential security vulnerabilities that could be exploited maliciously.

- Specification Conformance: The code was rigorously compared against the provided specifications to confirm its fidelity in performing as described, ensuring the implementation accurately reflected the intended design and requirements.

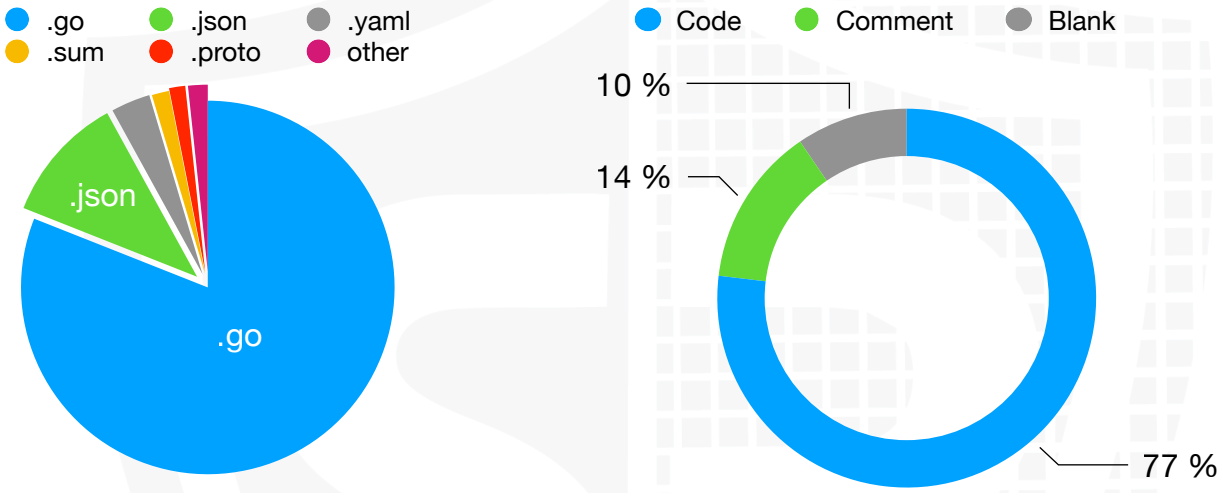
- Test Coverage Analysis: We analyzed to ascertain the extent of the test cases' coverage over the codebase. This involved determining how much code was executed during these tests to identify untested paths.

- Symbolic Execution: This technique was applied to analyze how different inputs affect the code execution paths. It helped understand the conditions under which various program parts would execute, revealing potential vulnerabilities.

Metrics

Codebase Composition Overview

This report section analyzes the project's codebase, primarily composed of Go (81%), with extensive comments and documentation, especially in .proto files. The distribution shows 77% code, 14% comments, and 10% blank lines, indicating a well-documented and maintainable structure.



Extension	Total Code	Total Comment	Total Blank	Percent
.go	138.085	24.417	17.097	81
.json	19.675	0	0	11
.yaml	5.853	19	1126	3.4
.sum	2.591	25	0	1.5
.proto	2.333	745	570	1.4
other				

Cyclomatic Complexity

Cyclomatic Complexity is a software metric used to quantify the complexity of a program. This metric, introduced by Thomas J. McCabe in 1976, measures the number of linearly independent paths through a program's source code. In practical terms, it is determined by the control flow graph of the program, where nodes represent blocks of code that do not contain any control flow keywords (like if, for, switch, case, etc.), and edges represent the flow of execution. The cyclomatic complexity is calculated using the formula: $V(G)=E-N+2P$

Top 25, Highest Cyclomatic Complexity Score

Score	Function	File and Line Number
187	(*fastReflection_Validator).ProtoMethods	api/axiome/staking/v1beta1/staking.pulsar.go:2926
164	(*fastReflection_GenesisState).ProtoMethods	api/axiome/distribution/v1beta1/genesis.pulsar.go:4638
154	(*fastReflection_GenesisState).ProtoMethods	api/axiome/vote/v1beta1/genesis.pulsar.go:818
130	(*fastReflection_GenesisState).ProtoMethods	api/axiome/staking/v1beta1/genesis.pulsar.go:724
128	(*fastReflection_Params).ProtoMethods	api/axiome/staking/v1beta1/staking.pulsar.go:10000
127	(*fastReflection_Info).ProtoMethods	api/axiome/referral/v1beta1/types.pulsar.go:479
115	(*fastReflection_Poll).ProtoMethods	api/axiome/vote/v1beta1/types.pulsar.go:2561
93	file_axiome_staking_v1beta1_query_proto_init	api/axiome/staking/v1beta1/query.pulsar.go:16365
89	(*fastReflection_ProposalHistoryRecord).ProtoMethods	api/axiome/vote/v1beta1/types.pulsar.go:1247
84	(*fastReflection_MsgCreateValidator).ProtoMethods	api/axiome/staking/v1beta1/tx.pulsar.go:368
84	(*fastReflection_Params).ProtoMethods	api/axiome/distribution/v1beta1/distribution.pulsar.go:374
80	(*fastReflection_RedelegationEntry).ProtoMethods	api/axiome/staking/v1beta1/staking.pulsar.go:8458
80	(*fastReflection_UnbondingDelegationEntry).ProtoMethods	api/axiome/staking/v1beta1/staking.pulsar.go:7751
79	(*fastReflection_Module).ProtoMethods	api/axiome/staking/module/v1/module.pulsar.go:462

Score	Function	File and Line Number
79	(*fastReflection_GenesisState).ProtoMethods	api/axiome/referral/v1beta1/genesis.pulsar.go:517
78	(*fastReflection_Params).ProtoMethods	api/axiome/slashing/v1beta1/slashing.pulsar.go:1055
78	(*fastReflection_ValidatorSigningInfo).ProtoMethods	api/axiome/slashing/v1beta1/slashing.pulsar.go:379
78	(*fastReflection_ValidatorSigningInfo).ProtoMethods	api/axiome/slashing/v1beta1/slashing.pulsar.go:379
76	(*fastReflection_QueryValidatorDistributionInfoResponse).ProtoMethods	api/axiome/distribution/v1beta1/query.pulsar.go:1728
75	(*fastReflection_Description).ProtoMethods	api/axiome/staking/v1beta1/staking.pulsar.go:1993
75	(*fastReflection_CommunityPoolSpendProposalWithDeposit).ProtoMethods	api/axiome/distribution/v1beta1/distribution.pulsar.go:6509
73	(*fastReflection_CurrentResponse).ProtoMethods	api/axiome/vote/v1beta1/query.pulsar.go:2640
71	(*fastReflection_Proposal).ProtoMethods	api/axiome/vote/v1beta1/types.pulsar.go:397
70	(*fastReflection_StakeAuthorization).ProtoMethods	api/axiome/staking/v1beta1/authz.pulsar.go:404
69	file_axiome_staking_v1beta1_staking_proto_init	api/axiome/staking/v1beta1/staking.pulsar.go:15545

For the project under review, the Average Cyclomatic Complexity score is **3.22**, indicating a low overall complexity and suggesting that the codebase is relatively straightforward to manage and maintain.

Overall tested functions

In the comprehensive audit conducted on the project, both automated tools and manual inspection techniques scrutinized **8974** functions.

Call graph

The call graph generated during the audit is extensive and detailed, reflecting the complex interactions within the codebase. Due to its size and complexity, it is too large to be effectively displayed within the confines of this report.

Audit Results

Final Words & Comments

The Axiome Layer 1 blockchain audit, powered by Cosmos SDK, has been completed with great diligence and attention to detail. It is pleasing to report that no critical security issues were discovered during the review. This finding underscores the robustness of the codebase and the efficacy of the development practices employed.

Dependency Review

Our analysis identified 15 potential vulnerabilities in 15 dependencies. Four were deemed significant enough to be explicitly listed in the issues table for immediate attention. These dependencies pose potential security risks and should be addressed promptly to mitigate any adverse impacts on the system's security and stability.

Error Handling

Throughout the audit, several instances of unhandled errors were noted. Addressing these unhandled errors is crucial to enhancing the system's reliability and tolerance. Proper error handling ensures the system can gracefully manage unexpected conditions without compromising ongoing operations or security.

Codec Recommendation

We observed the usage of NewAminoCodec within the codebase. To align with current best practices and ensure future compatibility, it is recommended to transition to NewLegacyAmino. This change will improve the codebase's maintainability and enhance its performance and security characteristics.

Scope of Review

During this audit, certain aspects, such as unused constants, functions, exported functions, global variables, and parameters, were excluded from the scope of our review. While these elements were not directly evaluated, we suggest revisiting these areas in future audits to ensure that they do not introduce security concerns or affect the overall quality of the codebase.

Critical issues

No critical issues

High issues

#1 | Improper Input Validation

File	Severity	Location	Status
Main	High	Dependency	Open

Description:
The "ValidateVoteExtensions" helper function in Cosmos SDK may allow incorrect voting power assumptions. This issue affects github.com/cosmos/cosmos-sdk versions 0.50.0-alpha.0 through 0.50.4.

Changelog Cosmos SDK 0.50.5:
https://github.com/cosmos/cosmos-sdk/releases/tag/v0.50.5

Advisory:
Upgrade to 0.50.6. More information under:
https://github.com/advisories/GHSA-95rx-m9m5-m94v

#2 | Error may be not nil

File	Severity	Location	Status
Main	High	x/staking/keeper/points.go:46 x/staking/types/ref_hooks.go:44	Open

Description:
Reports instances where variables might have nil or an unexpected value because of the associated error that is not checked for being non-nil, as in v, err := foo().

#3 | Constant condition

File	Severity	Location	Status
Main	High	x/staking/client/cli/tx.go:516 x/staking/keeper/delegation.go:1096 x/staking/keeper/query_utils.go:44	Open

Description:
Condition is always false.

#3 | Deprecated element

File	Severity	Location	Status
Main	High	app/params/config.go:52	Open

Description:
Deprecated: This method is supported for backward compatibility only and will be removed in a future release. Use SetPurpose and SetCoinType instead.

Docs: https://pkg.go.dev/github.com/cosmos/cosmos-sdk/types@v0.50.3#Config.SetFullFundraiserPath

#4 | Deprecated element

File	Severity	Location	Status
Main	High	x/staking/keeper/keeper.go:136 x/staking/keeper/keeper.go:148	Open

Description:
Deprecated: IntProto defines a Protobuf wrapper around an Int object. Deprecated: Prefer to use math.Int directly. It supports binary Marshal and Unmarshal.

Docs: https://pkg.go.dev/github.com/cosmos/cosmos-sdk/types@v0.50.3#IntProto

#5 | Loop with Unreachable Exit Condition

File	Severity	Location	Status
Main	High	Dependency	Open

Description:
In the package google.golang.org/protobuf versions prior to 1.33.0, the "protojson.Unmarshal" function can enter an infinite loop when unmarshaling certain forms of invalid JSON. This condition can occur when unmarshaling into a message which contains a "google.protobuf.Any" value, or when the "UnmarshalOptions.DiscardUnknown" option is set.

Advisory:
Upgrade to 1.33.0
https://pkg.go.dev/vuln/GO-2024-2611
https://github.com/advisories/GHSA-8r3f-844c-mc37

#6 | Improper Restriction of Excessive Authentication Attempts

File	Severity	Location	Status
Main	High	Dependency	Open

Description:
JumpServer is an open-source bastion host. When users enable MFA and use a public key for authentication, the Koko SSH server does not verify the corresponding SSH private key. An attacker could exploit a vulnerability by utilizing a disclosed public key to attempt brute-force authentication against the SSH service. This issue has been patched in versions 3.6.5 and 3.5.6. There are no known workarounds for this issue.

Advisory:
Upgrade to 0.21.0

https://github.com/jumpserver/jumpserver/security/advisories/GHSA-jv3c-27cv-w8jv

Medium issues

#1 | Insecure Default Initialization of Resource

File	Severity	Location	Status
Main	Medium	Dependency	Open

Description:
Dependency go:github.com/cometbft/cometbft:v0.38.5 is vulnerable

A default configuration in CometBFT is small for common use cases and may prevent the slashing mechanism from working in specific cases. The default values for EvidenceParams.MaxAgeNumBlocks and EvidenceParams.MaxAgeDuration consensus parameters may not be sufficient for common use cases to provide coverage for the entire unbonding period for a chain (Staking.UnbondingTime). Suppose the conditions of both parameters are exceeded. In that case, evidence may be prematurely expired and considered no longer valid, potentially allowing for unpunished Byzantine behavior if the evidence is discovered outside that window.

Advisory:
Upgrade to 1.0.0-alpha.2

#2 | Potential nil dereference

File	Severity	Location	Status
Main	Medium	x/distribution/keeper/delegation.go:120	Open

Description:
Receiver „rewards“ might be „nil“ in a call.

NOTE: Add operates under the invariant that coins are sorted by denominations.

CONTRACT: Add will never return Coins where one has a non-positive amount. In other words, IsValid will always return true.

Docs: https://pkg.go.dev/github.com/cosmos/cosmos-sdk/types@v0.50.3#DecCoins.Add

Low issues

#1 | Type is not defined

File	Severity	Location	Status
Main	Low	X/vote/types/types.go:185 X/vote/keeper/keeper.go:519	Open

Description - Types *Poll_CanValidate and *Poll_MinStatus are not defined.

Informational issues

No informational issues

Legend for the Issue Status

Attribute or Symbol	Meaning
Open	The issue is not fixed by the project team.
Fixed	The issue is fixed by the project team.
Acknowledged(ACK)	The issue has been acknowledged or declared as part of business logic.

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY