FYEO

Security Code Review of Axelar - Stellar Integration

Axelar Foundation

January 2025
Version 1.0

Presented by:

FYEO Inc.

PO Box 147044
Lakewood CO 80214
United States

Security Level
Public

TABLE OF CONTENTS

Executive Summary

Overview

Axelar Foundation engaged FYEO Inc. to perform a Security Code Review of Axelar - Stellar Integration.

The assessment was conducted remotely by the FYEO Security Team. Testing took place on December 21 - January 09, 2025, and focused on the following objectives:

- To provide the customer with an assessment of their overall security posture and any risks that were discovered within the environment during the engagement.

- To provide a professional opinion on the maturity, adequacy, and efficiency of the security measures that are in place.

- To identify potential issues and include improvement recommendations based on the results of our tests.

This report summarizes the engagement, tests performed, and findings. It also contains detailed descriptions of the discovered vulnerabilities, steps the FYEO Security Team took to identify and validate each issue, as well as any applicable recommendations for remediation.

Key Findings

The following issues have been identified during the testing period. These should be prioritized for remediation to reduce the risk they pose:

- FYEO-AX-SOR-01 – Event Signers Rotated - missing `newSignersData`

- FYEO-AX-SOR-02 – Missing token manager compared to evm implementation

- FYEO-AX-SOR-03 – Token service EVM discrepancies

- FYEO-AX-SOR-04 – Code without implementation

- FYEO-AX-SOR-05 – Gas service doesn't check gas token

- FYEO-AX-SOR-06 – No check of minimum signers in contract

- FYEO-AX-SOR-07 – Weak validation for threshold of signers

Based on our review process, we conclude that the reviewed code implements the documented functionality.

Scope and Rules of Engagement

The FYEO Review Team performed a Security Code Review of Axelar - Stellar. The following table documents the targets in scope for the engagement. No additional systems or resources were in scope for this assessment.

The source code was supplied through a public repository at
https://github.com/axelarnetwork/axelar-cgp-stellar with the commit hash
ab4361c58daffebd099ab386910b55a4d56d152f.

Remediations were provided with the commit hash 1b77756e62f35f7ac016dab7fbba9e2a06375e87.

Files included in the code review

```
axelar-cgp-soroban/
├── contracts/
│   ├── axelar-gas-service/
│   │   ├── src/
│   │   │   ├── contract.rs
│   │   │   ├── error.rs
│   │   │   ├── event.rs
│   │   │   ├── interface.rs
│   │   │   ├── lib.rs
│   │   │   └── storage_types.rs
│   │   └── Cargo.toml
│   ├── axelar-gateway/
│   │   ├── src/
│   │   │   ├── auth.rs
│   │   │   ├── contract.rs
│   │   │   ├── error.rs
│   │   │   ├── event.rs
│   │   │   ├── executable.rs
│   │   │   ├── interface.rs
│   │   │   ├── lib.rs
│   │   │   ├── messaging_interface.rs
│   │   │   ├── storage_types.rs
│   │   │   ├── testutils.rs
│   │   │   └── types.rs
│   │   └── Cargo.toml
│   ├── axelar-operators/
│   │   ├── src/
│   │   │   ├── contract.rs
│   │   │   ├── error.rs
│   │   │   ├── event.rs
│   │   │   ├── lib.rs
```

Files included in the code review

```
│     │     │     └── storage_types.rs
│     │     └── Cargo.toml
│     ├── example/
│     │     ├── src/
│     │     │     ├── contract.rs
│     │     │     ├── event.rs
│     │     │     ├── lib.rs
│     │     │     └── storage_types.rs
│     │     └── Cargo.toml
│     ├── interchain-token/
│     │     ├── src/
│     │     │     ├── contract.rs
│     │     │     ├── error.rs
│     │     │     ├── event.rs
│     │     │     ├── interface.rs
│     │     │     ├── lib.rs
│     │     │     └── storage_types.rs
│     │     └── Cargo.toml
│     ├── interchain-token-service/
│     │     ├── src/
│     │     │     ├── abi.rs
│     │     │     ├── contract.rs
│     │     │     ├── error.rs
│     │     │     ├── event.rs
│     │     │     ├── executable.rs
│     │     │     ├── interface.rs
│     │     │     ├── lib.rs
│     │     │     ├── storage_types.rs
│     │     │     ├── token_handler.rs
│     │     │     └── types.rs
│     │     └── Cargo.toml
│     └── upgrader/
│           ├── src/
│           │     ├── contract.rs
│           │     ├── error.rs
│           │     └── lib.rs
│           └── Cargo.toml
└── packages/
      ├── axelar-soroban-std/
      │     ├── src/
      │     │     ├── interfaces/
      │     │     │     ├── testdata/
      │     │     │     │     ├── contract_non_trivial_migration.rs
```

Files included in the code review

```
│   │   │   │   │   ├── contract_trivial_migration.rs
│   │   │   │   │   └── mod.rs
│   │   │   │   ├── mod.rs
│   │   │   │   ├── operatable.rs
│   │   │   │   ├── ownable.rs
│   │   │   │   └── upgradable.rs
│   │   │   ├── address.rs
│   │   │   ├── error.rs
│   │   │   ├── events.rs
│   │   │   ├── lib.rs
│   │   │   ├── testutils.rs
│   │   │   ├── token.rs
│   │   │   ├── traits.rs
│   │   │   ├── ttl.rs
│   │   │   └── types.rs
│   │   └── Cargo.toml
│   └── axelar-soroban-std-derive/
│       ├── src/
│       │   └── lib.rs
│       └── Cargo.toml
```

Table 1: Scope

Technical Analyses and Findings

During the Security Code Review of Axelar - Stellar, we discovered:

- 3 findings with LOW severity rating.

- 4 findings with INFORMATIONAL severity rating.

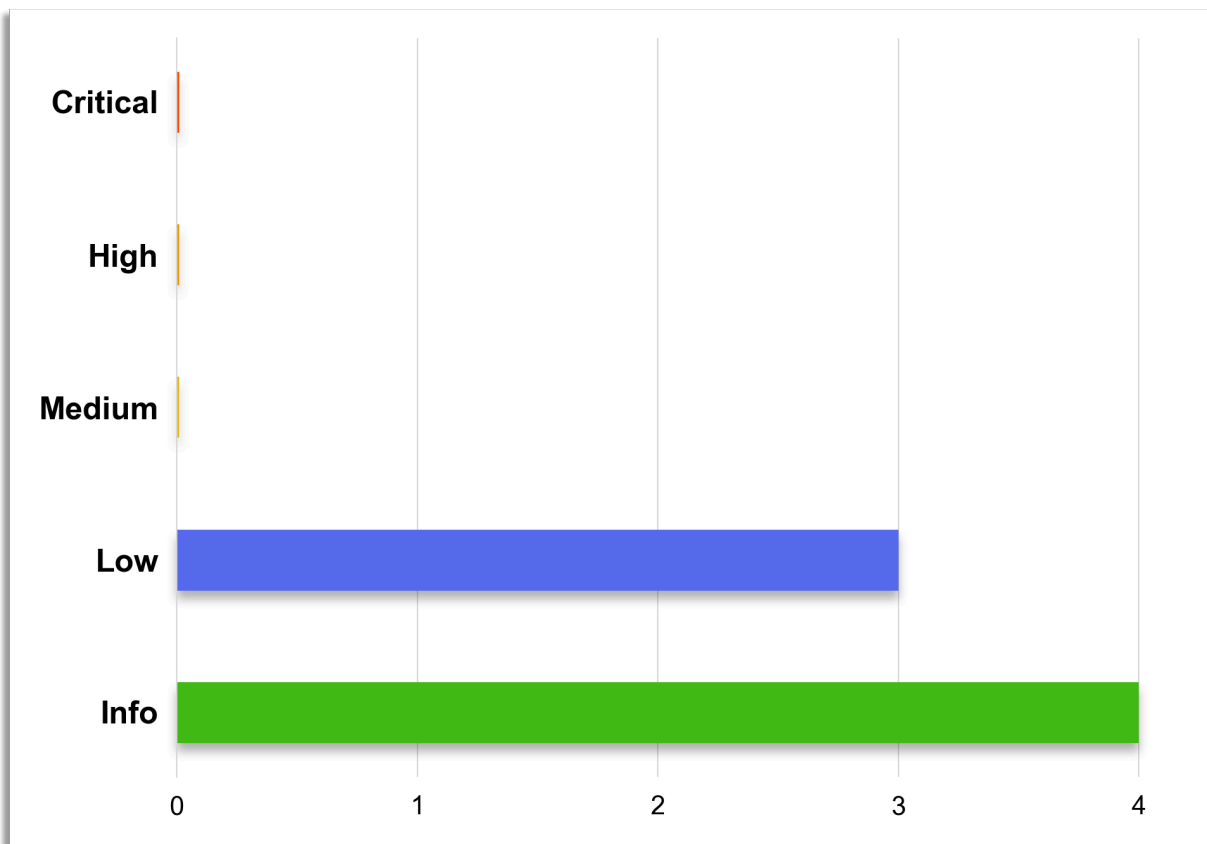The following chart displays the findings by severity.

Figure 1: Findings by Severity

Findings

The *Findings* section provides detailed information on each of the findings, including methods of discovery, explanation of severity determination, recommendations, and applicable references.

The following table provides an overview of the findings.

Finding #	Severity	Description
FYEO-AX-SOR-01	Low	Event Signers Rotated - missing `newSignersData`
FYEO-AX-SOR-02	Low	Missing token manager compared to evm implementation
FYEO-AX-SOR-03	Low	Token service EVM discrepancies
FYEO-AX-SOR-04	Informational	Code without implementation
FYEO-AX-SOR-05	Informational	Gas service doesn't check gas token
FYEO-AX-SOR-06	Informational	No check of minimum signers in contract
FYEO-AX-SOR-07	Informational	Weak validation for threshold of signers

Table 2: Findings Overview

Technical Analysis

The source code has been manually validated to the extent that the state of the repository allowed. The validation includes confirming that the code correctly implements the intended functionality.

Conclusion

Based on our review process, we conclude that the code implements the documented functionality to the extent of the reviewed code.

Technical Findings

General Observations

The Axelar Cross-chain Gateway Protocol for Stellar is a set of contracts designed to closely mirror the logic of the EVM implementation of Axelar's Cross-chain Gateway Protocol, specifically adapted for use on the Stellar network.

The overall quality of the code is strong: it is well-structured, well-documented, and covered by unit tests.

The bridging logic mirrors that of the existing EVM implementation, while also addressing Soroban-specific cases, such as handling storage expiration, the authorization model, and the upgrade process.

During the audit, a few minor discrepancies were found between the Soroban implementation and the EVM version. These discrepancies do not impact the security of the contracts. While the validator logic generally follows the EVM implementation, it could be further strengthened by implementing stricter signer requirements but as this is set up from the Axelar side this should not affect the security if deployed correctly.

In conclusion, the security aspects of the protocol are well handled.

Event Signers Rotated - missing `newSignersData`

Finding ID: FYEO-AX-SOR-01
Severity: Low
Status: Remediated

Description

Some events have different structure compared to EVM implementation.

This has been omitted due to Stellar's strict event size limit. The team requested a limit increase and Stellar has accepted. The team will investigate the number of signers at which including newSignersData would cause the emission of this event to fail under the increased limit.

This is remediated by the following fix: https://github.com/axelarnetwork/axelar-cgp-stellar/pull/184

Proof of Issue

1. Signers Rotated - missing newSignersData

File name: contracts/axelar-gateway/src/event.rs

Line number: 32

```
pub fn rotate_signers(env: &Env, epoch: u64, signers_hash: BytesN<32>) {
    let topics = (Symbol::new(env, "signers_rotated"), epoch, signers_hash);
    env.events().publish(topics, ());
}
```

Evm:
https://github.com/axelarnetwork/axelar-gmp-sdk-solidity/blob/432449d7b330ec6edf5a8e0746644a253486ca87/contracts/governance/BaseWeightedMultisig.sol#L160C4-L160C71

```
emit SignersRotated(newEpoch, newSignersHash, newSignersData);
```

Severity and Impact Summary

Missing fields in events may create challenges during integration.

Recommendation

It is recommended to follow the existing implementation as a standard.

Missing token manager compared to EVM implementation

Finding ID: FYEO-AX-SOR-02
Severity: Low
Status: Acknowledged

Description

Missing Token Manager. The team has not implemented this and are still considering how to implement a Token Managers in Stellar ITS at the time of audit.

Proof of Issue

File name: contracts/interchain-token-service/src/contract.rs
Line number: 197

```
fn deploy_interchain_token(
        env: &Env,
        caller: Address,
        salt: BytesN<32>,
        token_metadata: TokenMetadata,
        initial_supply: i128,
        minter: Option<Address>,
    ) -> Result<BytesN<32>, ContractError> {
```

Evm:
https://github.com/axelarnetwork/interchain-token-service/blob/a7439cd108cf9809636951fcb6153dd7b565
17c9/contracts/InterchainTokenService.sol#L373

```
_deployTokenManager(tokenId, TokenManagerType.NATIVE_INTERCHAIN_TOKEN,
abi.encode(minter, tokenAddress));
```

Severity and Impact Summary

Discrepancies with existing implementation may create challenges during integration.

Recommendation

It is recommended to follow the existing implementation as a standard.

Token service EVM discrepancies

Finding ID: FYEO-AX-SOR-03
Severity: Low
Status: Remediated

Description

Some events have different structure compared to EVM implementation. While the token manager is not yet implemented since the team is still considering how to implement the token manager on the Stellar ITS.

This has been remediated by the team via the following pr:

https://github.com/axelarnetwork/axelar-cgp-stellar/commit/1b77756e62f35f7ac016dab7fbba9e2a06375e87

Proof of Issue

1.   No address check (remediated)

File name: contracts/interchain-token-service/src/contract.rs

Line number: 329

```
ensure!(amount > 0, ContractError::InvalidAmount);
```

Evm:
https://github.com/axelarnetwork/interchain-token-service/blob/a7439cd108cf9809636951fcb6153dd7b56517c9/contracts/InterchainTokenService.sol#L1061

```
if (destinationAddress.length == 0) revert EmptyDestinationAddress();
if (amount == 0) revert ZeroAmount();
```

3.   No validation that destination chain is not current chain (remediated)

File name: contracts/interchain-token-service/src/contract.rs

Line number: 723

```
fn deploy_remote_token(
        env: &Env,
        caller: Address,
        deploy_salt: BytesN<32>,
        destination_chain: String,
        gas_token: Token,
    ) -> Result<BytesN<32>, ContractError> {
```

Evm:
https://github.com/axelarnetwork/interchain-token-service/blob/a7439cd108cf9809636951fcb6153dd7b56517c9/contracts/InterchainTokenService.sol#L375

```
if (chainNameHash == keccak256(bytes(destinationChain))) revert
CannotDeployRemotelyToSelf();
```

Severity and Impact Summary

Discrepancies with existing implementation may create challenges during integration.

Recommendation

It is recommended to follow the existing implementation as a standard.

Code without implementation

Finding ID: FYEO-AX-SOR-04
Severity: Informational
Status: Acknowledged

Description

Token contract has unfinished implementation. This is due to the fact that the team has not decided which of these should be implemented at the time of audit.

The team feedback: The token doesn't support these intentionally at the moment since they're not required. We might add these in a future upgrade.

Proof of Issue

File name: contracts/interchain-token/src/contract.rs

Line number: 63

```
fn set_authorized(_env: Env, _id: Address, _authorize: bool) {
        todo!()
    }

    fn authorized(_env: Env, _id: Address) -> bool {
        todo!()
    }

    fn clawback(_env: Env, _from: Address, _amount: i128) {
        todo!()
    }
```

Severity and Impact Summary

Not finished code may lead to undefined behavior.

Recommendation

It is recommended to finalize or remove unnecessary code.

Gas service doesn't check gas token

Finding ID: FYEO-AX-SOR-05
Severity: Informational
Status: Acknowledged

Description

Gas service only checks the token amount but allows any token to be used.

Feedback: Furthermore, there is no risk of a user permanently losing funds when sending an unsupported token, as the Relayer can `refund` the user in this case.

Proof of Issue

File name: contracts/axelar-gas-service/src/contract.rs

Line number: 45

```
ensure!(token.amount > 0, ContractError::InvalidAmount);
```

Severity and Impact Summary

Acceptance of any token in gas service.

The Relayer will only accept payment in supported tokens so this is mitigated by that. Recommendation

It is recommended to specify a supported gas token address.

No check of minimum signers in contract

Finding ID: FYEO-AX-SOR-06
Severity: Informational
Status: Acknowledged

Description

In the contract code it is possible that a weight of a single signer is sufficient to pass a threshold.

However this should be set up on the Axelar Amplifier side and it should not directly affect security if deployed correctly.

Proof of Issue

File name: contracts/axelar-gateway/src/auth.rs

Line number: 215

```
for signer in weighted_signers.signers.iter() {
    ensure!(
        previous_signer < signer.signer,
        ContractError::InvalidSigners
    );

    ensure!(signer.weight != 0, ContractError::InvalidWeight);

    previous_signer = signer.signer;
    total_weight = total_weight
        .checked_add(signer.weight)
        .ok_or(ContractError::WeightOverflow)?;
}

let threshold = weighted_signers.threshold;
ensure!(
    threshold != 0 && total_weight >= threshold,
    ContractError::InvalidThreshold
);
```

Severity and Impact Summary

Even though signers are set and controlled on the axelar side some control of signer size in the contract might be a good idea

Recommendation

It is recommended to validate that the weight of each individual signer is lower than threshold.

Weak validation for threshold of signers

Finding ID: FYEO-AX-SOR-07
Severity: **Informational**
Status: **Acknowledged**

Description

The recommended threshold is 2/3 of total weight but the code only checks for it to be greater than zero. However this should be set up on the Axelar side and should therefore not affect security when deployed correctly.

Proof of Issue

File name: contracts/axelar-gateway/src/auth.rs

Line number: 230

```
ensure!(
    threshold != 0 && total_weight >= threshold,
    ContractError::InvalidThreshold
);
```

Severity and Impact Summary

Weak threshold undermines the decentralized nature of the platform and increases the risk of fraudulent validators.

Recommendation

It is recommended to ensure that the threshold is at least 2/3 of total weight.

References

- https://github.com/axelarnetwork/axelar-gmp-sdk-solidity/blob/main/contracts/gateway/INTEGRATION.md

Our Process

Methodology

FYEO Inc. uses the following high-level methodology when approaching engagements. They are broken up into the following phases.

Figure 2: Methodology Flow

Kickoff

The project is kicked off as the sales process has concluded. We typically set up a kickoff meeting where project stakeholders are gathered to discuss the project as well as the responsibilities of participants. During this meeting we verify the scope of the engagement and discuss the project activities. It's an opportunity for both sides to ask questions and get to know each other. By the end of the kickoff there is an understanding of the following:

- Designated points of contact

- Communication methods and frequency

- Shared documentation

- Code and/or any other artifacts necessary for project success

- Follow-up meeting schedule, such as a technical walkthrough

- Understanding of timeline and duration

Ramp-up

Ramp-up consists of the activities necessary to gain proficiency on the project. This can include the steps needed for familiarity with the codebase or technological innovation utilized. This may include, but is not limited to:

- Reviewing previous work in the area including academic papers

- Reviewing programming language constructs for specific languages

- Researching common flaws and recent technological advancements

Review

The review phase is where most of the work on the engagement is completed. This is the phase where we analyze the project for flaws and issues that impact the security posture. Depending on the project this may include an analysis of the architecture, a review of the code, and a specification matching to match the architecture to the implemented code.

In this code audit, we performed the following tasks:

1. Security analysis and architecture review of the original protocol

2. Review of the code written for the project

3. Compliance of the code with the provided technical documentation

The review for this project was performed using manual methods and utilizing the experience of the reviewer. No dynamic testing was performed, only the use of custom-built scripts and tools were used to assist the reviewer during the testing. We discuss our methodology in more detail in the following sections.

Code Safety

We analyzed the provided code, checking for issues related to the following categories:

- General code safety and susceptibility to known issues

- Poor coding practices and unsafe behavior

- Leakage of secrets or other sensitive data through memory mismanagement

- Susceptibility to misuse and system errors

- Error management and logging

This list is general and not comprehensive, meant only to give an understanding of the issues we are looking for.

Technical Specification Matching

We analyzed the provided documentation and checked that the code matches the specification. We checked for things such as:

- Proper implementation of the documented protocol phases

- Proper error handling

- Adherence to the protocol logical description

Reporting

FYEO Inc. delivers a draft report that contains an executive summary, technical details, and observations about the project.

The executive summary contains an overview of the engagement including the number of findings as well as a statement about our general risk assessment of the project. We may conclude that the overall risk is low but depending on what was assessed we may conclude that more scrutiny of the project is needed.

We report security issues identified, as well as informational findings for improvement, categorized by the following labels:

- Critical

- High

- Medium

- Low

- Informational

The technical details are aimed more at developers, describing the issues, the severity ranking and recommendations for mitigation.

As we perform the audit, we may identify issues that aren't security related, but are general best practices and steps that can be taken to lower the attack surface of the project. We will call those out as we encounter them and as time permits.

As an optional step, we can agree on the creation of a public report that can be shared and distributed with a larger audience.

Verify

After the preliminary findings have been delivered, this could be in the form of the approved communication channel or delivery of the draft report, we will verify any fixes within a window of time specified in the project. After the fixes have been verified, we will change the status of the finding in the report from open to remediated.

The output of this phase will be a final report with any mitigated findings noted.

Additional Note

It is important to note that, although we did our best in our analysis, no code audit or assessment is a guarantee of the absence of flaws. Our effort was constrained by resource and time limits along with the scope of the agreement.

While assessing the severity of the findings, we considered the impact, ease of exploitability, and the probability of attack. This is a solid baseline for severity determination.

The Classification of vulnerabilities

Security vulnerabilities and areas for improvement are weighted into one of several categories using, but is not limited to, the criteria listed below:

Critical – vulnerability will lead to a loss of protected assets

- This is a vulnerability that would lead to immediate loss of protected assets

- The complexity to exploit is low

- The probability of exploit is high

High - vulnerability has potential to lead to a loss of protected assets

- All discrepancies found where there is a security claim made in the documentation that cannot be found in the code

- All mismatches from the stated and actual functionality

- Unprotected key material

- Weak encryption of keys

- Badly generated key materials

- Txn signatures not verified

- Spending of funds through logic errors

- Calculation errors overflows and underflows

Medium - vulnerability hampers the uptime of the system or can lead to other problems

- Insecure calls to third party libraries

- Use of untested or nonstandard or non-peer-reviewed crypto functions

- Program crashes, leaves core dumps or writes sensitive data to log files

Low – vulnerability has a security impact but does not directly affect the protected assets

- Overly complex functions

- Unchecked return values from 3rd party libraries that could alter the execution flow

Informational

- General recommendations