F Y E O

Security Code Review of the Axelar - Solana Integration

Axelar Foundation

March 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 engaged FYEO Inc. to perform a Security Code Review of Axelar Solana.

The assessment was conducted remotely by the FYEO Security Team. Testing took place on January 15 - February 19, 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-SOL-01 – Governance: Config can be deleted
- FYEO-AX-SOL-02 – Governance: Initialize is done first come first served
- FYEO-AX-SOL-03 – Its: Missing account checks for InterchainTransfer
- FYEO-AX-SOL-04 – Relayer: Latest processed signature is not updated in config.toml
- FYEO-AX-SOL-05 – Relayer: Potential data loss
- FYEO-AX-SOL-06 – GasService: Authority check missing
- FYEO-AX-SOL-07 – Its: Initialize is done first come first served
- FYEO-AX-SOL-08 – Amplifier: Incomplete test to check for invalid characters
- FYEO-AX-SOL-09 – Amplifier: Missing tests
- FYEO-AX-SOL-10 – GasService: Code duplication

- FYEO-AX-SOL-11 – GasService: No checks on amounts transferred
- FYEO-AX-SOL-12 – Gateway: Dead code
- FYEO-AX-SOL-13 – Gateway: Incorrect names of errors and functions
- FYEO-AX-SOL-14 – Gateway: Missing tests
- FYEO-AX-SOL-15 – Governance: Code Improvements
- FYEO-AX-SOL-16 – Its: Mint and Token account checks
- FYEO-AX-SOL-17 – MultiCall: Incorrect account indexing in MultiCallPayloadBuilder
- FYEO-AX-SOL-18 – Relayer: Incorrect constant value (or name)
- FYEO-AX-SOL-19 – Relayer: No check that compute_budget has not exceeded the limit
- FYEO-AX-SOL-20 – Relayer: No limit on the number of concurrent connections

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 the Axelar - Solana integration. 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/eigerco/solana-axelar/ with the commit hash b6be90dbbe5e987bbce94f6d860064ce878b1d22.

The Relayer was reviewed based on the commit hash a72dbcf479af3d2a36d00adf5934629af7161753.

The Amplifier was reviewed based on the PRs https://github.com/axelarnetwork/axelar-amplifier/pull/744 and https://github.com/eigerco/axelar-amplifier/pull/56.

Remediations were submitted with the following hashes:

Contracts: 59b05c1905c8810269cf93a00abecd235241197c

Relayer: b0fd956176ee743f11e8222f802a73f54f85743e

Amplifier: (solana) 00c7f11e1bdb7df044e9267518b87ca42e056206 and (add-multisig-prover-sol-logic) 1a0c1d9cb24181c506b80cbb5a73ed99c3eed088

Files included in the code review

```
solana-axelar-contracts/
└── solana/
    ├── crates/
    │   ├── axelar-executable/
    │   │   └── src/
    │   │       ├── axelar_payload/
    │   │       │   └── encoding/
    │   │       │     ├── abi_encoding.rs
    │   │       │     ├── borsh_encoding.rs
    │   │       │     └── mod.rs
    │   │       ├── axelar_payload.rs
    │   │       └── lib.rs
    │   ├── axelar-solana-encoding/
    │   │   └── src/
    │   │       ├── types/
    │   │       │   ├── execute_data.rs
    │   │       │   ├── messages.rs
    │   │       │   ├── mod.rs
    │   │       │   ├── payload.rs
    │   │       │   ├── pubkey.rs
    │   │       │   └── verifier_set.rs
    │   │       ├── error.rs
    │   │       ├── hasher.rs
    │   │       └── lib.rs
    │   ├── axelar-solana-gateway-test-fixtures/
    │   │   └── src/
    │   │       ├── base.rs
    │   │       ├── gas_service.rs
    │   │       ├── gateway.rs
    │   │       ├── lib.rs
    │   │       └── test_signer.rs
    │   └── gateway-event-stack/
    │     └── src/
    │         └── lib.rs
    └── programs/
    ├── axelar-solana-gas-service/
    │   └── src/
    │     ├── processor/
    │     │   ├── initialize.rs
    │     │   ├── native.rs
    │     │   └── spl.rs
    │     ├── entrypoint.rs
    │     ├── instructions.rs
```

Files included in the code review

```
│       ├── lib.rs
│       ├── processor.rs
│       └── state.rs
├── axelar-solana-gateway/
│   └── src/
│       ├── processor/
│       │   ├── approve_message.rs
│       │   ├── call_contract.rs
│       │   ├── call_contract_offchain_data.rs
│       │   ├── close_message_payload.rs
│       │   ├── commit_message_payload.rs
│       │   ├── initialize_config.rs
│       │   ├── initialize_message_payload.rs
│       │   ├── initialize_payload_verification_session.rs
│       │   ├── rotate_signers.rs
│       │   ├── transfer_operatorship.rs
│       │   ├── validate_message.rs
│       │   ├── verify_signature.rs
│       │   └── write_message_payload.rs
│       ├── state/
│       │   ├── config.rs
│       │   ├── incoming_message.rs
│       │   ├── message_payload.rs
│       │   ├── signature_verification.rs
│       │   ├── signature_verification_pda.rs
│       │   └── verifier_set_tracker.rs
│       ├── entrypoint.rs
│       ├── error.rs
│       ├── instructions.rs
│       ├── lib.rs
│       ├── processor.rs
│       └── state.rs
├── axelar-solana-governance/
│   └── src/
│       ├── processor/
│       │   ├── gmp/
│       │   │   ├── approve_operator_proposal.rs
│       │   │   ├── cancel_operator_approval.rs
│       │   │   ├── cancel_time_lock_proposal.rs
│       │   │   ├── mod.rs
│       │   │   └── schedule_time_lock_proposal.rs
│       │   ├── execute_operator_proposal.rs
│       │   ├── execute_proposal.rs
```

Files included in the code review

```
│       │   ├── init_config.rs
│       │   ├── mod.rs
│       │   ├── transfer_operatorship.rs
│       │   └── withdraw_tokens.rs
│       ├── state/
│       │   ├── mod.rs
│       │   ├── operator.rs
│       │   └── proposal.rs
│       ├── entrypoint.rs
│       ├── events.rs
│       ├── instructions.rs
│       ├── lib.rs
│       └── sol_types.rs
├── axelar-solana-its/
│   └── src/
│       ├── instructions/
│       │   ├── interchain_token.rs
│       │   ├── minter.rs
│       │   ├── mod.rs
│       │   ├── operator.rs
│       │   └── token_manager.rs
│       ├── processor/
│       │   ├── interchain_token.rs
│       │   ├── interchain_transfer.rs
│       │   ├── mod.rs
│       │   └── token_manager.rs
│       ├── state/
│       │   ├── flow_limit.rs
│       │   ├── mod.rs
│       │   └── token_manager.rs
│       ├── entrypoint.rs
│       ├── executable.rs
│       └── lib.rs
└── axelar-solana-multicall/
    └── src/
        ├── entrypoint.rs
        ├── instructions.rs
        ├── lib.rs
        └── processor.rs
```

ampd/ (PR 744, PR 56)

Files included in the code review

```
└── src/
  ├── config.rs
  ├── encoding/
  │   ├── mod.rs
  │   └── solana.rs
  ├── error.rs
  ├── evm/
  │   └── finalizer.rs
  ├── handlers/
  │   ├── config.rs
  │   ├── mod.rs
  │   ├── solana_verify_msg.rs
  │   └── solana_verify_verifier_set.rs
  ├── lib.rs
  ├── solana/
  │   ├── mod.rs
  │   ├── msg_verifier.rs
  │   └── verifier_set_verifier.rs
packages/
└── axelar-wasm-std/
  └── src/
      └── address.rs



Relayer
├── crates
│   ├── common-serde-utils
│   │   └── src
│   │       └── lib.rs
│   ├── effective-tx-sender
│   │   └── src
│   │       └── lib.rs
│   ├── file-based-storage
│   │   └── src
│   │       └── lib.rs
│   ├── gateway-gas-computation
│   │   └── src
│   │       └── lib.rs
│   ├── rest-service
│   │   ├── src
│   │   │   ├── component.rs
│   │   │   ├── config.rs
```

Files included in the code review

```
│   │   │   ├── endpoints
│   │   │   │   ├── call_contract_offchain_data.rs
│   │   │   │   ├── health.rs
│   │   │   │   └── mod.rs
│   │   │   └── lib.rs
│   │   └── tests
│   │       └── call_contract_offchain_data.rs
│   ├── retrying-solana-http-sender
│   │   └── src
│   │       └── lib.rs
│   ├── solana-axelar-relayer
│   │   ├── build.rs
│   │   └── src
│   │       ├── main.rs
│   │       └── telemetry.rs
│   ├── solana-event-forwarder
│   │   └── src
│   │       ├── component.rs
│   │       ├── config.rs
│   │       └── lib.rs
│   ├── solana-gateway-task-processor
│   │   └── src
│   │       ├── component
│   │       │   └── message_payload.rs
│   │       ├── component.rs
│   │       ├── config.rs
│   │       └── lib.rs
│   └── solana-listener
│       └── src
│           ├── component
│           │   ├── log_processor.rs
│           │   ├── signature_batch_scanner.rs
│           │   └── signature_realtime_scanner.rs
│           ├── component.rs
│           ├── config.rs
│           └── lib.rs
└── xtask
    └── src
        └── main.rs
```

Table 1: Scope

Technical Analyses and Findings

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

- 1 finding with HIGH severity rating.
- 4 findings with MEDIUM severity rating.
- 2 findings with LOW severity rating.
- 13 findings with INFORMATIONAL severity rating.

The following chart displays the findings by severity.

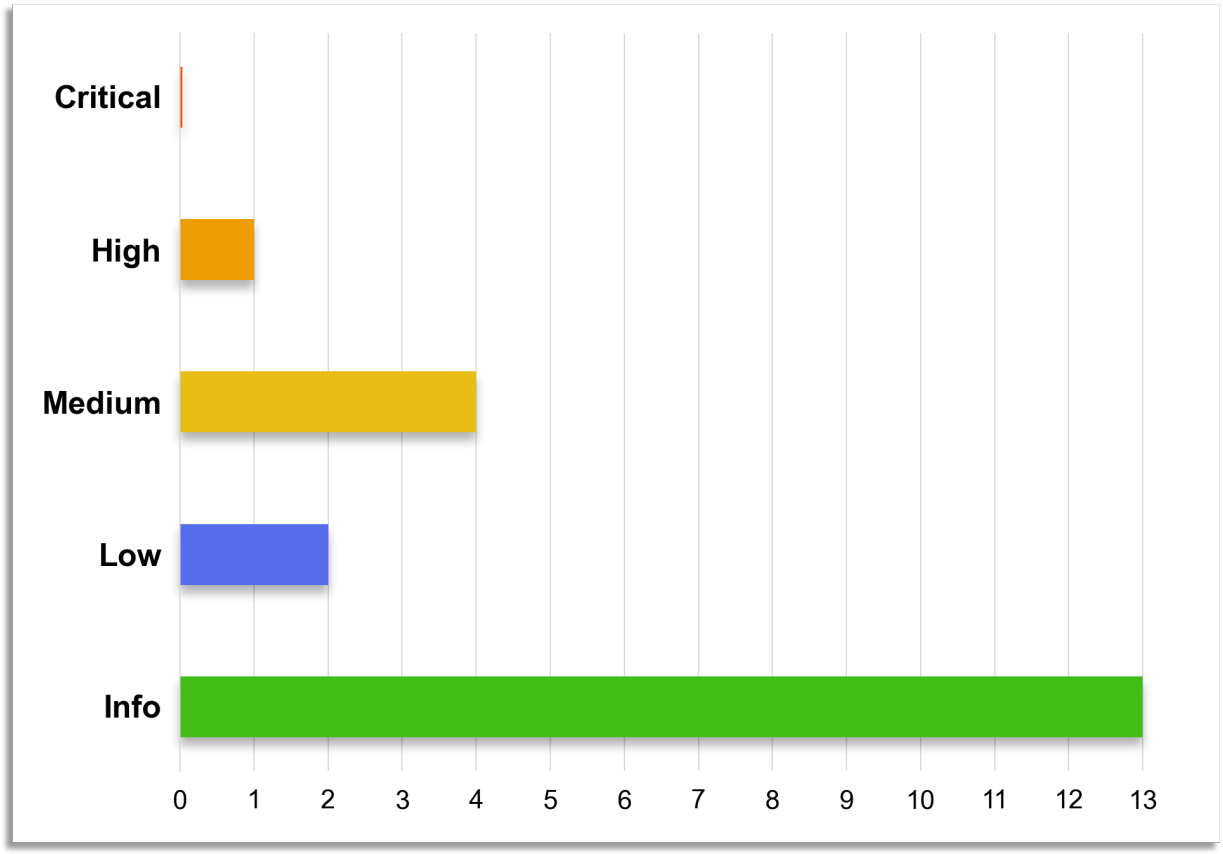

Figure 1: Findings by Severity

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-SOL-01	**High**	Governance: Config can be deleted
FYEO-AX-SOL-02	**Medium**	Governance: Initialize is done first come first served
FYEO-AX-SOL-03	**Medium**	Its: Missing account checks for InterchainTransfer
FYEO-AX-SOL-04	**Medium**	Relayer: Latest processed signature is not updated in config.toml
FYEO-AX-SOL-05	**Medium**	Relayer: Potential data loss
FYEO-AX-SOL-06	**Low**	GasService: Authority check missing
FYEO-AX-SOL-07	**Low**	Its: Initialize is done first come first served
FYEO-AX-SOL-08	**Informational**	Amplifier: Incomplete test to check for invalid characters
FYEO-AX-SOL-09	**Informational**	Amplifier: Missing tests
FYEO-AX-SOL-10	**Informational**	GasService: Code duplication
FYEO-AX-SOL-11	**Informational**	GasService: No checks on amounts transferred
FYEO-AX-SOL-12	**Informational**	Gateway: Dead code
FYEO-AX-SOL-13	**Informational**	Gateway: Incorrect names of errors and functions
FYEO-AX-SOL-14	**Informational**	Gateway: Missing tests
FYEO-AX-SOL-15	**Informational**	Governance: Code Improvements
FYEO-AX-SOL-16	**Informational**	Its: Mint and Token account checks
FYEO-AX-SOL-17	**Informational**	MultiCall: Incorrect account indexing in MultiCallPayloadBuilder
FYEO-AX-SOL-18	**Informational**	Relayer: Incorrect constant value (or name)

FYEO-AX-SOL-19	**Informational**	Relayer: No check that compute_budget has not exceeded the limit
FYEO-AX-SOL-20	**Informational**	Relayer: No limit on the number of concurrent connections

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

Contracts

Axelar Solana Gateway
The Gateway is well structured and implemented with a focus on secure and efficient cross-chain message management. The project uses thorough validation of PDAs (Program Derived Addresses) and signatures, which minimizes the risk of double handling and unauthorized access.

Axelar Solana Gas Service
This module implements gas payment using both SOL and SPL tokens. It efficiently handles fund transfers by accurately emitting events to track transactions. The module includes PDA correctness checks as well as owner and signature validation to help prevent unauthorized transfers.

Axelar Solana Multicall
The Multicall program provides a tool for aggregating multiple calls into a single multicall using encoding and decoding payloads. The project supports both Borsh coding and ABI coding, giving flexibility Merkle tree mechanisms for account and message verification are implemented efficiently.

Axelar Solana Its
The ITS program deals with token deployments and issues tokens in accordance with incoming messages. This program can also track flow limits on tokens and supports a variety of ways to implement with tokens on Solana.

Axelar Solana Governance This program deals with governance related concerns such as the creation of proposals, execution, cancellation etc.

Code style
It would be helpful for the maintenance of the code base to have more separation between account checks and general code. Doing account checks early would make reasoning about the checks easier. Currently checks are somewhat spread out in the code.

Test coverage
Test coverage isn't the best. In most cases negative scenarios are not considered, some features are not tested at all. A few issues are noticed in the project, but most of them are not vulnerabilities but informative in nature.

Amplifier

The pull requests introduce support for Solana verification by adding new handlers, message verification logic, and configuration updates. The main changes include the addition of `SolanaMsgVerifier` and `SolanaVerifierSetVerifier` configurations, implementation of Solana transaction validation, and new modules for processing Solana-related events. The modifications extend the system's capabilities to handle Solana messages and verifier set updates efficiently.

The structure of the implementation follows existing patterns, ensuring consistency across different blockchain integrations. Error handling is well-managed, and message verification includes checks to ensure correctness. Unit tests have been added to validate the new functionality, covering key scenarios such as message validation and verifier set confirmation.

However, several issues were identified:

- Incomplete test for invalid characters in Solana addresses
- Missing unit tests in contracts/multisig-prover/src/encoding/solana.rs
- Unnecessary voting transaction submission
- Missing signature verification in transaction validation

Relayer

The Axelar Solana Relayer project stands out as both well-structured and efficiently designed. It is centered on monitoring Solana transactions in real-time, executing tasks accordingly, and retrying operations in case of failures.

The Relayer uses structured error handling, notably via crates like eyre and tracing, which provide clear and actionable debugging information. Critical operations propagate errors effectively, reducing the likelihood of failures.

The system validates signatures and tracks transactions meticulously, mitigating risks such as double-processing.

Several points require attention to bolster robustness and consistency:

- The latest processed signature is not updated in config.toml. Consequently, restarting the service could lead to reprocessing transactions.
- Potential Data Loss in Memory-Mapped Storage
- Compute Budget Limit Not Enforced

General status

In general, the transaction and retry monitoring mechanisms in Axelar Solana Relayer are thoughtfully implemented. The main challenges are related to maintaining state integrity, preventing potential data inconsistencies, and accurately managing computational budgets.

Status

All the vulnerabilities of medium and above have been addressed and fixed swiftly by the team during the audit. Some outstanding low and informational findings that have not been currently addressed are acknowledged by the team and tracked in their internal issue tracker and will be remediated later by the team.

Governance: Config can be deleted

Finding ID: FYEO-AX-SOL-01
Severity: **High**
Status: **Remediated**

Description

The config account can transfer its entire balance and it may thus fall below rent exemption requirements. This would mean the config gets deleted. And it then becomes possible for anyone to take ownership of the contract by setting themselves as the controller.

Proof of Issue

File name: solana/programs/axelar-solana-governance/src/processor/withdraw_tokens.rs
Line number: 41

```
match program_utils::transfer_lamports(config_pda, receiver, amount)


pub fn transfer_lamports(
    from_account: &AccountInfo<'_>,
    to_account: &AccountInfo<'_>,
    amount_of_lamports: u64,
) -> ProgramResult {
    // Does the from account have enough lamports to transfer?
    if **from_account.try_borrow_lamports()? < amount_of_lamports {
        return Err(ProgramError::InsufficientFunds);
    }
    // Debit from_account and credit to_account
    let mut from_account = from_account.try_borrow_mut_lamports()?;
    let mut to_account = to_account.try_borrow_mut_lamports()?;
    **from_account = from_account
        .checked_sub(amount_of_lamports)
        .ok_or(ProgramError::InsufficientFunds)?;
    **to_account = to_account
        .checked_add(amount_of_lamports)
        .ok_or(ProgramError::InsufficientFunds)?;
    Ok(())
}
```

Severity and Impact Summary

If the config is deleted, it can be re-initialized. Since that is done by a public function, the first caller can become the owner of the contract.

Recommendation

Always make sure to check that the account balance can not drop below rent exemption. Otherwise consider using a separate account to store funds.

Governance: Initialize is done first come first served

Finding ID: FYEO-AX-SOL-02
Severity: **Medium**
Status: **Remediated**

Description

The initializer is public and can be called by anyone. It does set the controller of the contract.

Proof of Issue

File name: solana/programs/axelar-solana-governance/src/processor/init_config.rs
Line number: 20

```
pub(crate) fn process(
    program_id: &Pubkey,
    accounts: &[AccountInfo<'_>],
    mut governance_config: GovernanceConfig,
) -> Result<(), ProgramError> {
    let accounts_iter = &mut accounts.iter();
    let payer = next_account_info(accounts_iter)?;
    let root_pda = next_account_info(accounts_iter)?;
    let system_account = next_account_info(accounts_iter)?;

    // Check: System Program Account
    if !system_program::check_id(system_account.key) {
        return Err(ProgramError::IncorrectProgramId);
    }

    ...
```

Severity and Impact Summary

The first one to call can set themselves as the admin of the contract. This was set as a Medium as the config can be deleted as discussed in another issue.

Recommendation

Restrict this to be an authorized call. A good choice would be using the upgrade_authority_address as the signer.

Its: Missing account checks for InterchainTransfer

Finding ID: FYEO-AX-SOL-03
Severity: **Medium**
Status: **Remediated**

Description

This function is missing its account checks. It was communicated that these checks have already been added in an updated version.

Proof of Issue

File name: solana/programs/axelar-solana-its/src/processor/interchain_transfer.rs
Line number: 60

```
pub fn process_inbound_transfer<'a>(
    message: Message,
    payer: &'a AccountInfo<'a>,
    message_payload_account: &'a AccountInfo<'a>,
    accounts: &'a [AccountInfo<'a>],
    payload: &InterchainTransfer,
) -> ProgramResult {
    let parsed_accounts =
        GiveTokenAccounts::from_account_info_slice(accounts, &(payer,
message_payload_account))?;
    let token_manager = TokenManager::load(parsed_accounts.token_manager_pda)?;
    assert_valid_token_manager_pda(
        parsed_accounts.token_manager_pda,
        parsed_accounts.its_root_pda.key,
        &token_manager.token_id,
        token_manager.bump,
    )?;

    let Ok(converted_amount) = payload.amount.try_into() else {
        msg!("Failed to convert amount");
        return Err(ProgramError::InvalidInstructionData);
    };

    give_token(&parsed_accounts, &token_manager, converted_amount)?;
```

Severity and Impact Summary

The accounts are not thoroughly checked. This could mean that unexpected accounts are passed in.

Recommendation

Make sure to check this function in a fashion similar to the other `GMPPayload` handlers.

Relayer: Latest processed signature is not updated in config.toml

Finding ID: FYEO-AX-SOL-04
Severity: **Medium**
Status: **Remediated**

Description

There are no explicit places where `latest_processed_signature` is updated and written back to the TOML config. This means that when the service restarts, it will always take the old value from TOML.

Proof of Issue

File name: crates/solana-listener/src/component/signature_realtime_scanner.rs
Line number: 86

```
pub(crate) async fn process_realtime_logs(
    config: crate::Config,
    latest_processed_signature: Option<Signature>,
    rpc_client: Arc<RpcClient>,
    mut signature_sender: MessageSender,
) -> Result<(), eyre::Error>
```

Since transaction processing is competitively executed, updating the last_processed_signature should be guaranteed after successful transaction processing, but avoiding data races and possible conflicts.

Suitable place to update last_processed_signature.

Line number: 144

```
        // Process the merged stream
        while let Option::<eyre::Result<SolanaTransaction>>::Some(result) =
            merged_stream.next().await
        {
            match result {
                Ok(log_item) => {
                    // Send the fetched log item
                    signature_sender.send(log_item).await?;
                    /*
                      ====Update last_processed_signature=====
                    */
                }
                Err(err) => {
                    // Handle error in fetch_logs
                    tracing::error!(?err, "Error in merged stream");
                    continue 'outer;
                }
            }
        }
```

Severity and Impact Summary

If the `latest_processed_signature` is not updated to the config.toml file, it may cause already processed signatures to be re-processed when the service is restarted.

Recommendation

Add logic to update `latest_processed_signature` in the `config.toml` file.

Relayer: Potential data loss

Finding ID: FYEO-AX-SOL-05
Severity: **Medium**
Status: **Remediated**

Description

Method `mmap.flush_async()?` initiates flushing modified pages to durable storage, but it will not wait for the operation to complete before returning. The file's metadata (including last modification timestamp) may not be updated.

Proof of Issue

File name: crates/file-based-storage/src/lib.rs
Line number: 90

```
fn set_task_item_id<F>(
    &self,
    task_item_id: &TaskItemId,
    field_mutator: F,
) -> Result<(), io::Error>
where
    F: Fn(&mut InternalState, u128),
{
    let mut mmap = self.mmap.lock().expect("lock should not be poisoned");
    let raw_u128 = task_item_id.0.as_u128();
    let data = bytemuck::from_bytes_mut::<InternalState>(&mut mmap[..]);
    field_mutator(data, raw_u128);
    mmap.flush_async()?;
    drop(mmap);
    Ok(())
}
```

Severity and Impact Summary

If `set_task_item_id()` doesn't persist the data properly, `latest_processed_task_id()` and `latest_queried_task_id()` will return outdated or incorrect values, causing tasks to be re-processed again.

Recommendation

Use synchronous flush (mmap.flush()?).

GasService: Authority check missing

Finding ID: FYEO-AX-SOL-06
Severity: **Low**
Status: **Remediated**

Description

The function has checks for the system account and PDA, but these checks are not sufficient to protect the system from unauthorized initialization of the configuration account.

Proof of Issue

File name: solana/programs/axelar-solana-gas-service/src/processor/initialize.rs
Line numbers: 14

```
    if !system_program::check_id(system_account.key) {
        return Err(ProgramError::InvalidInstructionData);
    }

    let (_, bump) = get_config_pda(program_id, &salt, authority.key);

    // Check: Gateway Config account uses the canonical bump.
    assert_valid_config_pda(bump, &salt, authority.key, config_pda.key)?;
```

Severity and Impact Summary

Authority verification is necessary to prevent unauthorized initialization or overwriting of the configuration. If this account can be created or overwritten by any user, an attacker will be able to install themselves as a controller, negatively impacting the security of the entire system.

Recommendation

Add an authority check.

Its: Initialize is done first come first served

Finding ID: FYEO-AX-SOL-07
Severity: **Low**
Status: **Remediated**

Description

The initializer is public and can be called by anyone. It does set the operator of the program.

Proof of Issue

File name: solana/programs/axelar-solana-its/src/processor/mod.rs
Line number: 124

```
fn process_initialize(program_id: &Pubkey, accounts: &[AccountInfo<'_>]) ->
ProgramResult {
    let account_info_iter = &mut accounts.iter();
    let payer = next_account_info(account_info_iter)?;
    ...
```

Severity and Impact Summary

The first one to call can set themselves as the operator of the program.

Recommendation

Restrict this to be an authorized call. A good choice would be using the upgrade_authority_address as the signer.

Amplifier: Incomplete test to check for invalid characters

Finding ID: FYEO-AX-SOL-08
Severity: **Informational**
Status: **Remediated**

Description

The test includes a valid address, too short and too long, invalid characters - ‘I’ (capital ‘i’), ‘0’ (zero), ‘O’ (capital ‘o’). But Base58 prohibits the use of ‘l’ (lowercase ‘L’) because it can easily be confused with ‘I’ (capital ‘i’)

Proof of Issue

File name: packages/axelar-wasm-std/src/address.rs
Line number: 209

```
fn validate_solana_address() {
        ...
        // Invalid address: contains invalid character '0' (zero)
        let invalid_char_addr = "4f3J7t1HgX1t36k6rph2pYJrWxk9uT1RrB2K3nVHDh8D0";
        assert_err_contains!(
            address::validate_address(invalid_char_addr,
&address::AddressFormat::Base58Solana),
            address::Error,
            address::Error::InvalidAddress(..)
        );

        // Invalid address: contains invalid character 'O'
        let invalid_char_addr2 = "4f3J7t1HgX1t36k6rph2pYJrWxk9uT1RrB2K3nVHDh8DO";
        assert_err_contains!(
            address::validate_address(invalid_char_addr2,
&address::AddressFormat::Base58Solana),
            address::Error,
            address::Error::InvalidAddress(..)
        );
        ...
        // Invalid address: contains invalid character 'I'
        let invalid_char_addr3 = "4f3J7t1HgX1t36k6rph2pYJrWxk9uT1RrB2K3nVHDh8DI";
        assert_err_contains!(
            address::validate_address(invalid_char_addr3,
&address::AddressFormat::Base58Solana),
            address::Error,
            address::Error::InvalidAddress(..)
        );
}
```

Severity and Impact Summary

The issue affects the correctness of address validation but does not introduce a security vulnerability. While this does not impact the validation function’s ability to reject invalid addresses (since bs58::decode should already fail), the absence of a dedicated test creates a gap in test coverage.

Recommendation

Add a test that verifies that an address with ‘l’ (lowercase ‘L’) will be considered invalid.

Amplifier: Missing tests

Finding ID: FYEO-AX-SOL-09
Severity: **Informational**
Status: **Acknowledged**

Description

The current implementation does not contain unit tests to validate its correctness.

Proof of Issue

File name: contracts/multisig-prover/src/encoding/solana.rs

Severity and Impact Summary

Since these functions handle critical cryptographic data encoding and hashing, a lack of test coverage increases the risk of undetected errors, data corruption, or unexpected encoding failures.

Recommendation

Implement tests for - `encode_execute_data()`, - `payload_digest()`, - `to_verifier_set()`, - `to_pub_key()` with PublicKey::Ecdsa and PublicKey::Ed25519, - `to_payload()` with Payload::Messages and Payload::VerifierSet - `to_msg()` - `to_signature()` with Signature::Ecdsa, Signature::EcdsaRecoverable, Signature::Ed25519

GasService: Code duplication

Finding ID: FYEO-AX-SOL-10
Severity: **Informational**
Status: **Remediated**

Description

The functions `process_pay_native_for_contract_call`, `add_native_gas`, `collect_fees_native`, and `refund_native` all contain a repeated block of code that performs configuration PDA validation.

Proof of Issue

File name: solana/programs/axelar-solana-gas-service/src/processor/native.rs
Line numbers: 31, 71, 107, 142

```
    config_pda.check_initialized_pda_without_deserialization(program_id)?;
    let data = config_pda.try_borrow_data()?;
    let config = Config::read(&data).ok_or(ProgramError::InvalidAccountData)?;
    assert_valid_config_pda(config.bump, &config.salt, &config.authority,
config_pda.key)?;
```

Severity and Impact Summary

This does not introduce a direct security vulnerability, but it poses a maintenance risk. If the validation logic needs to be updated, the duplicated code must be updated in multiple locations, increasing the chance of errors or inconsistencies.

Recommendation

Refactor the repeated PDA validation code into a dedicated helper function. This centralizes the validation logic and simplifies future maintenance and auditing of the code.

GasService: No checks on amounts transferred

Finding ID: FYEO-AX-SOL-11
Severity: **Informational**
Status: **Remediated**

Description

Gas add, fee and refund functions for native and SPL tokens do not check incoming gas_fee_amount, amount or fees data.

Proof of Issue

File name: solana/programs/axelar-solana-gas-service/src/processor/native.rs
File name: solana/programs/axelar-solana-gas-service/src/processor/spl.rs

Severity and Impact Summary

Events with zero amount can be created, which in reality do not perform any action.

Recommendation

Add checks for the input values.

Gateway: Dead code

Finding ID: FYEO-AX-SOL-12
Severity: **Informational**
Status: **Acknowledged**

Description

The `verify_eddsa_signature` function is not called anywhere. This may be a feature that was planned but is not used.

Proof of Issue

File name: solana/programs/axelar-solana-gateway/src/state/signature_verification.rs
Line number: 292

```
pub fn verify_eddsa_signature()
```

Severity and Impact Summary

This does not carry any threat.

Recommendation

If a part of the code is not used in the contract - it would be best to remove it.

Gateway: Incorrect names of errors and functions

Finding ID: FYEO-AX-SOL-13
Severity: **Informational**
Status: **Acknowledged**

Description

Function and error names do not match the logic.

Proof of Issue

File name: solana/programs/axelar-solana-gateway/src/lib.rs
Line number: 251

```
solana_program::msg!("Error: Invalid Verifier Set Root PDA ");
```

The `assert_valid_verifier_set_tracker_pda()` and `assert_valid_signature_verification_pda()` functions return the same error text, which looks like a copying error. The message in `assert_valid_signature_verification_pda()` is expected to reflect an error related to the signature verification PDA, such as “Error: Invalid Signature Verification PDA”.

Proof of Issue

File name: solana/programs/axelar-solana-gateway/src/processor/initialize_payload_verification_session.rs
Line number: 70

```
if verification_session_account.lamports() != 0 {
    solana_program::msg!("Error: verification session account is not initialized");
    return Err(ProgramError::AccountAlreadyInitialized);
}
```

The account must be writable and its balance must be 0, which means it has not yet been initialized. However, if the balance is not 0, we get an error that the account has not been initialized, when the error should report that the account has **already been initialized**.

Severity and Impact Summary

This does not carry any threat.

Recommendation

Concise naming will improve the maintainability of the code base.

Gateway: Missing tests

Finding ID: FYEO-AX-SOL-14
Severity: **Informational**
Status: **Acknowledged**

Description

The current implementation does not contain unit tests to validate its correctness.

Proof of Issue

File name: solana/programs/axelar-solana-gateway/src/state/signature_verification.rs

Severity and Impact Summary

Functions such as `process_signature`, `accumulate_threshold`, `mark_slot_done`, and `verify_ecdsa_signature` are either not tested or only partially covered. This limited test coverage risks undetected logical errors in these core verification processes.

Recommendation

Implement tests: - successful and failing cases of `process_signature` - proper accumulation of signature thresholds and transition to valid state - correct behavior of `mark_slot_done` including duplicate slot detection - verification of both valid and invalid ECDSA signatures - negative test scenarios for out-of-bound indexes in slot handling

Proof of Issue

File name: solana/programs/axelar-solana-gateway/tests/integration/initialize_message_payload.rs

Description

Current integration tests cover the positive scenario.

Recommendation

These cases can be additionally tested:

- Payer is not signer or writable
- Message payload account is not writable
- Invalid system program account
- Overflow on adjusted_account_size calculation
- Incorrect PDA calculation

Governance: Code Improvements

Finding ID: FYEO-AX-SOL-15
Severity: **Informational**
Status: **Acknowledged**

Description

These are some minor concerns regarding code clarity.

Proof of Issue

File name: solana/programs/axelar-solana-governance/src/processor/transfer_operatorship.rs
Line number: 51

```
let old_operator = config_data.operator;
config_data.operator = new_operator;
```

The operator is updated in a single step and without providing a signature. This could lead to a loss of access.

File name: solana/programs/axelar-solana-governance/src/processor/init_config.rs
Line number: 47

```
program_utils::init_pda::<GovernanceConfig>(
    payer,
    root_pda,
    program_id,
    system_account,
    governance_config
```

There are no bounds checks on the config. Values such as the minimum proposal ETA may be set with values that can potentially cause issues.

File name: solana/programs/axelar-solana-governance/src/processor/withdraw_tokens.rs
Line number: 25

```
pub(crate) fn process(
    _program_id: &Pubkey,
    accounts: &[AccountInfo<'_>],
    amount: u64,
```

The withdraw function accepts 0 amounts. This will essentially do nothing.

File name: solana/programs/axelar-solana-governance/src/processor/withdraw_tokens.rs
Line number: 29

```
let config_pda = next_account_info(accounts_iter)?;
let receiver = next_account_info(accounts_iter)?;
```

The withdraw function accepts that sender and receiver are the same account.

Severity and Impact Summary

These are not security concerns but the code could be improved to avoid potential issues in the future.

Recommendation

The code could be improved to make it less likely that future updates introduce issues.

Its: Mint and Token account checks

Finding ID: FYEO-AX-SOL-16
Severity: **Informational**
Status: **Acknowledged**

Description

The program enables various ways of 'deploying' interchain tokens. They may be created or may use existing ones. Depending on how tokens are being 'deployed', it would be advisable to check the various authorities. `mint_authority`, `freeze_authority`, `owner (authority)`, `delegate` and possibly `close_authority`.

For the original token program `Tokenkeg` it is advisable to check that the associated token account receiving bridged funds is indeed owned by the correct account as these can have their owner (authority) changed.

Proof of Issue

File name: solana/programs/axelar-solana-its/src/processor/token_manager.rs
Line number: 365

```
fn check_accounts(accounts: &DeployTokenManagerAccounts<'_>) -> ProgramResult {
    if !system_program::check_id(accounts.system_account.key) {
        msg!("Invalid system account provided");
        return Err(ProgramError::IncorrectProgramId);
    }

    if accounts
        .token_manager_pda
        .check_uninitialized_pda()
        .is_err()
    {
        msg!("TokenManager PDA is already initialized");
        return Err(ProgramError::AccountAlreadyInitialized);
    }

    if
spl_token_2022::check_spl_token_program_account(accounts.token_mint.owner).is_err() {
        msg!("Invalid token mint account provided");
        return Err(ProgramError::InvalidAccountData);
    }

    if accounts.token_program.key != accounts.token_mint.owner {
        msg!("Mint and program account mismatch");
        return Err(ProgramError::IncorrectProgramId);
    }

    if !spl_associated_token_account::check_id(accounts.ata_program.key) {
        msg!("Invalid associated token account program provided");
        return Err(ProgramError::IncorrectProgramId);
    }

    Ok(())
}
```

Mint authority is appropriately checked in `handle_mintership()`. The `freeze_authority` is however not checked. For Token2022 the configured extensions could also be considered.

Severity and Impact Summary

While this is only dealing with approved messages, the impact is minimal.

Recommendation

It is recommended to check the implementation against requirements.

MultiCall: Incorrect account indexing in MultiCallPayloadBuilder

Finding ID: FYEO-AX-SOL-17
Severity: **Informational**
Status: **Remediated**

Description

The `MultiCallPayloadBuilder::build()` function assigns `account_start_index` and `account_end_index` based on the number of accounts provided for each instruction. However, in cases where an instruction has zero accounts, `account_start_index` will be equal to `account_end_index`.

Proof of Issue

File name: solana/programs/axelar-solana-multicall/src/instructions.rs
Line numbers: 227

```
        let account_start_index = current_index
            .checked_add(1)
            .ok_or(PayloadError::Conversion)?;
        let account_end_index = account_start_index
            .checked_add(accounts.len())
             .ok_or(PayloadError::Conversion)?;
```

Severity and Impact Summary

If account_start_index == account_end_index, an empty account fragment is passed to `process_multicall()`, this will result in an error.

Recommendation

Add validation before assigning account_start_index and account_end_index.

Relayer: Incorrect constant value (or name)

Finding ID: FYEO-AX-SOL-18
Severity: **Informational**
Status: **Remediated**

Description

The constant name doesn't match the value.

Proof of Issue

File name: crates/retrying-solana-http-sender/src/lib.rs
Line number: 27

```
const TWO_MINUTES: Duration = Duration::from_millis(60 * 1_000);
```

Severity and Impact Summary

A mismatch between the constant name and its value may mislead developers.

Recommendation

Change constant name or value.

Relayer: No check that compute_budget has not exceeded the limit

Finding ID: FYEO-AX-SOL-19
Severity: **Informational**
Status: **Remediated**

Description

If computed_units is close to the upper limit (and you add another 10%), an instruction may be received that will be rejected by the RPC server or validator as invalid.

Proof of Issue

File name: crates/effective-tx-sender/src/lib.rs
Line number: 200

```
let compute_budget = computed_units.saturating_add(top_up);
let ix = ComputeBudgetInstruction::set_compute_unit_limit(
    compute_budget
        .try_into()
        .map_err(eyre::Error::from)
        .map_err(ComputeBudgetError::Generic)?,
);
```

Severity and Impact Summary

Exceeding the limit will cause an error when processing the transaction at the RPC-server or validator level, which will result in rejection of execution and unnecessary waste of funds.

Recommendation

Add a check for exceeding the limit. If exceeded, set unit_limit = MAX_COMPUTE_BUDGET (1_399_850).

Relayer: No limit on the number of concurrent connections

Finding ID: FYEO-AX-SOL-20
Severity: **Informational**
Status: **Remediated**

Description

The REST service accepts an unlimited number of connections, which can lead to overloading.

Proof of Issue

File name: crates/rest-service/src/component.rs
Line number: 80

```
pub fn new(
        config: &Config,
        chain_name: String,
        rpc_client: Arc<RpcClient>,
        amplifier_client: AmplifierCommandClient,
    ) -> Self {
        ...
        let router = Router::new()
            .route(
                endpoints::health::PATH,
                endpoints::health::handlers(),
            )
            .route(
                endpoints::call_contract_offchain_data::PATH,
                endpoints::call_contract_offchain_data::handlers(),
            )
            .with_state(Arc::new(state))
            .layer(DefaultBodyLimit::max(
                config.call_contract_offchain_data_size_limit,
            ))
         ...
}
```

Severity and Impact Summary

An attacker could open too many concurrent connections, eventually causing the service to crash or slow down.

Recommendation

Use tower::limit::ConcurrencyLimitLayer, which limits the number of concurrently processed requests.

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