Security Audit Report

Axelar solana-ampd

Axelar node

by ARDA

on April 15, 2026

Table of Contents

Disclaimer **3**

Terminology **4**

Objective **4**

Summary **5**

Code Issues & Recommendations **6**

C1: Gateway messages approved in V0 transactions are never validated by ampd 6

C2: Lookup-table accounts are missed during transaction parsing 8

C3: Attacker can permanently block verifier set rotations 10

C4: An attacker can stuck votes on poll messages by including a fake message in the poll 12

C5: ”filter_map” on account keys can silently shift indexes 14

C6: Misleading function name "is_instruction_from_gateway_program” 15

C7: Gateway check unnecessary and inconsistent with other chains 16

Disclaimer

The report makes no statements or warranties, either expressed or implied, regarding the security of the code, the information herein or its usage. It also cannot be considered as a sufficient assessment regarding the utility, safety and bugfree status of the code, or any other statements.

This report does not constitute legal or investment advice. It is for informational purposes only and is provided on an "as-is" basis. You acknowledge that any use of this report and the information contained herein is at your own risk. The authors of this report shall not be liable to you or any third parties for any acts or omissions undertaken by you or any third parties based on the information contained herein.

Terminology

Code: The code with which users interact.

Inherent risk: A risk for users that comes from a behavior inherent to the code's design.

Inherent risks only represent the risks inherent to the code's design, which are a subset of all the possible risks. **No inherent risk doesn't mean no risk.** It only means that no risk inherent to the code's design has been identified. Other kind of risks could still be present. For example, the issues not fixed incur risks for the users, or the upgradability of the code might also incur risks for the users.

Issue: A behavior unexpected by the users or by the project, or a practice that increases the chances of unexpected behaviors to appear.

Critical issue: An issue intolerable for the users or the project, that must be addressed.

Major issue: An issue undesirable for the users or the project, that we strongly recommend to address.

Medium issue: An issue uncomfortable for the users or the project, that we recommend to address.

Minor issue: An issue imperceptible for the users or the project, that we advise to address for the overall project security.

Objective

Our objective is to share everything we have found that would help assessing and improving the safety of the code:

1. The **inherent risks** of the code, labelled R1, R2, etc.
2. The **issues** in the **code**, labelled C1, C2, etc.
3. The **issues** in the **testing** of the code, labelled T1, T2, etc.
4. The **issues** in the **other** parts related to the code, labelled O1, O2, etc.
5. The **recommendations** to address each issue.

Summary

Blockchain: Axelar

Initial scope

- **Repository:** https://github.com/axelarnetwork/axelar-amplifier
- **Commit:** `dde8792d56538acf5e51e433bd9c3e1c3454bf49`
- **Path(s):**
  `./ampd-handlers/src/solana ./ampd-handlers/src/bin/solana ./packages/axelar-wasm-std/src/msg_id/base_58_solana_event_index.rs ./packages/axelar-wasm-std/src/address.rs`

Final scope

- **Repository:** https://github.com/axelarnetwork/axelar-amplifier
- **Commit:** `9ce604c2554536fcf46c8ea161a9bb2663a0f0bf`
- **Path(s):**
  `./ampd-handlers/src/solana ./ampd-handlers/src/bin/solana ./packages/axelar-wasm-std/src/msg_id/base_58_solana_event_index.rs ./packages/axelar-wasm-std/src/address.rs`

0 inherent risk in the final scope

0 issue in the final scope

7 issues reported in the initial scope and 0 remaining in the final scope.

Severity	Reported			Remaining		
	Code	Test	Other	Code	Test	Other
Critical	4	0	0	0	0	0
Major	0	0	0	0	0	0
Medium	0	0	0	0	0	0
Minor	3	0	0	0	0	0

Code Issues & Recommendations

C1: Gateway messages approved in V0 transactions are never validated by ampd

Severity: Critical **Status:** Fixed

Location

```
ampd-handlers/src/solana/mod.rs
    tx
```

Description

Current behavior: In order to verify that a message was emitted as an event by Solana Gateway, ampd fetches the transaction from Solana RPC client, by calling the method `self.client.get_transaction`.

However, this method does not support “versioned” transactions, including V0 transactions where accounts can be provided through a lookup table.

Consequently, any cross-chain message occurring in a V0 transaction will be invisible to the ampd verifier, leading to a `Vote::NotFound` for events that actually succeeded on-chain. In turn, these messages will never be routed to their destination.

Expected behavior: The ampd handler should be able to fetch and verify both legacy and V0 transactions, which are both used on Solana mainnet. According to Solana's RPC documentation (link), fetching V0 transactions can be done by providing an explicit argument `maxSupportedTransactionVersion`: *"If no `maxSupportedTransactionVersion` value is set, then only legacy transactions will be allowed in the RPC response. Therefore, your RPC requests WILL fail if any version 0 transactions are returned."*

Worst consequence: Messages sent via the Gateway in V0 transactions are never forwarded to the destination chain. An attacker aware of this could deliberately trigger operations through V0 transactions to prevent specific cross-chain operations from being verified, for example:

- Blocking verifier set rotations
- Blocking cross-chain liquidations.

Recommendation

We recommend replacing the call to RPC's method `get_transaction` by a call to `get_transaction_with_config`, specifying `max_supported_transaction_version: Some(0)`.

Moreover, we suggest adding a test witnessing that ampd supports voting for V0 transactions. To make the test as comprehensive as possible, we can mix the use of static accounts, ALT read-only accounts and ALT written-only accounts.

C2: Lookup-table accounts are missed during transaction parsing

Severity: Critical **Status:** Fixed

Location

`ampd-handlers/src/solana/mod.rs`
`is_instruction_from_gateway_program`

Description

Current behavior: When parsing a transaction from RPC in `tx`, the list of accounts involved in the transaction (`account_keys`) is derived from `raw_message.account_keys` (link), which contains only the static (non-lookup-table) keys of the transaction.

Later, `is_instruction_from_gateway_program` uses `instruction.program_id_index` to look up the program address in this `account_keys` vector (link).

However, Solana V0 transactions (link) can reference additional accounts via Address Lookup Tables. Such accounts' indexes are not in `raw_message.account_keys`. Rather, these indexes can be found in the transaction metadata field `loaded_addresses`, and the indexing starts from `account_keys.len()`.

Therefore, if the Gateway program's address is resolved through a lookup table rather than being in the static account keys, its `program_id_index` will exceed `account_keys.len()`, causing `is_instruction_from_gateway_program` to incorrectly return `false`:

```
if program_id_index >= account_keys.len() {
   return false;
}
```

Consequently, any cross-chain message occurring in a V0 transaction will be invisible to the ampd verifier, leading to a `Vote::NotFound` for events that actually succeeded on-chain. In turn, these messages will never be routed to their destination.

Expected behavior: The ampd handler should resolve all accounts referenced in the transaction, including those from lookup tables, so that `program_id_index` lookups work correctly for both legacy and V0 transactions.

Worst consequence: Messages sent via the Gateway in V0 transactions are never forwarded to the destination chain. An attacker aware of this could deliberately trigger operations through V0 transactions to prevent specific cross-chain operations from being verified, for example:

- Blocking verifier set rotations
- Blocking cross-chain liquidations.

Recommendation

When constructing `account_keys` from the transaction, we suggest also adding the accounts from lookup tables involved in the transaction. These accounts can be found in `meta.loadedAddresses.writable` and `meta.loadedAddresses.readonly`.

C3: Attacker can permanently block verifier set rotations

Severity: Critical **Status:** Fixed

Location

`ampd-handlers/src/solana/mod.rs`
`verify`

Description

Current behavior: When verifying a Solana transaction, the ampd handler checks whether the transaction failed (`tx.err.is_some()`) and immediately returns `Vote::FailedOnChain` (link), before performing any validation on the transaction content — i.e., before checking whether the instruction exists at the given index, whether it corresponds to a self-CPI event emitted by the Gateway program, or whether the event data matches the verifier set rotation.

This means that if `verify_verifier_set` is called with a `message_id` pointing to any arbitrary failed transaction, all honest ampd verifiers will vote `Vote::FailedOnChain` , regardless of whether the transaction has any relation to the Gateway.

This combines with two properties of the `voting-verifier` contract:

1. `verify_verifier_set` is permissionless and accepts an arbitrary `message_id` with the target `new_verifier_set` (link).
2. Verifier set status is keyed by `new_verifier_set.hash()` only — not by `message_id` (link). Furthermore, `is_confirmed()` returns `true` for `FailedOnSourceChain` (link), permanently blocking any future re-verification of the same verifier set content (link).

Expected behavior: The ampd handler should only return `Vote::FailedOnChain` for messages, not for verifier set rotations, because a verifier set rotation is not tied to a unique transaction.

Worst consequence: An attacker can permanently block verifier set rotations across all chains using this approach in ampd handler (e.g. Solana, EVMs, and Stellar).

Example:

- Attacker creates a failing transaction on Solana.
- He calls `verify_verifier_set(failing_tx_id, target_verifier_set)` at Voting Verifier.
- Honest ampd verifiers fetch the transaction, see it failed, and vote `Vote::FailedOnChain`.
- A quorum of `FailedOnChain` votes is reached, and the verifier set status becomes `FailedOnSourceChain`.
- Any future call to `verify_verifier_set` with the same verifier set content hits `is_confirmed() == true` and returns early, permanently blocking the rotation.

Recommendation

For verifier set rotations, if `tx.err.is_some()`, we recommend returning the vote `Vote::NotFound` instead of `Vote::FailedOnChain`.

This way, if a poll is created in Voting Verifier with a failed transaction for a verifier set rotation, then it will always be possible to create a new poll with a correct transaction.

C4: An attacker can stuck votes on poll messages by including a fake message in the poll

Appeared in intermediary commit: `d4ee8f866c7da12f5a4e62cf8c10f666df1795e2`

Severity: Critical **Status:** Fixed

Location

```
ampd-handlers/src/solana/mod.rs
    finalized_txs
```

Description

Current behavior: Recall that ampd votes on polls (i.e., sets of messages), and these polls are created in the Voting Verifier. Messages in a poll are identified by message IDs, which include a transaction's signature, and based on this signature ampd retrieves the actual transaction from the Solana RPC.

However, if a poll contains a single message whose signature has no corresponding on-chain transaction, then ampd fails to vote on the entire poll, i.e. on all other messages. This is problematic because (i) anybody can create a poll with a message whose signature has no corresponding on-chain transaction, and (ii) as long as this poll has not expired, the voting on all messages of this poll is stuck, preventing validation of these messages and their execution at destination.

Below we provide more details.

1) If a single RPC call to fetch a message's transaction fails, then the entire poll voting fails, and so no vote is cast for for all messages in the poll:

- In ampd `finalized_txs`, all per-transaction RPC results are collected using `collect::<Result<Vec<_>>>()?` (link). Therefore, if a single RPC call returns an error, the function `finalized_txs` returns `Err`.
- This error propagates through the voting handler, as it uses `?` on the result of `finalized_txs` (link). Therefore, no vote is cast for any message in the poll.

2) Anybody can create a poll containing a message that will make the corresponding PRC call fail:

- On the Solana RPC side, `getTransaction` returns null when a signature does not exist (see Solana docs).
- In `tx`, we call the Rust client's `get_transaction_with_config`, which returns `ClientResult<EncodedConfirmedTransactionWithStatusMeta>` (not `Option`), so a null result fails deserialization.

3) For some time, no poll can be created with the any of the messages from the DoS poll:

In Voting Verifier, when a poll is created with new messages, they go in `InProgress` status until the poll expires (this depends on the `block_expiry` parameter, currently around 1 min on Axelar).

Only after the poll expires, the message status transitions to `FailedToVerify` and can be re-submitted (link).

In summary, the attacker would create a poll every minute containing a non-existing transaction signature alongside the messages he wants to DoS.

Expected behavior: ampd should be able to vote on all messages of a poll even if some messages have no associated on-chain transaction.

Worst consequence: An attacker can prevent a Solana message from being verified.

Recommendation

In `tx`, if a message's signature has no associated on-chain transaction, we suggest not failing and instead returning `None`. This way the final vote for such messages will be `Vote::NotFound` (which is also the behavior for EVM ampd), and moreover ampd will correctly cast votes for other messages.

Moreover, we would add a unit test witnessing that ampd can still vote if on the poll if the poll contains a signature with no associated transaction.

C5: "filter_map" on account keys can silently shift indexes

Severity: Minor **Status:** Fixed

Location

`ampd-handlers/src/solana/mod.rs`
`tx`

Description

In `tx`, when parsing account keys from the RPC response, the code uses `filter_map` to convert string public keys into `Pubkey` values, and `filter_map` skips any keys that fail to be parsed (link).

In principle, no such parsing failure should occur because all public keys come from a finalized transaction returned by the Solana RPC, and thus should always be valid base58-encoded keys.

However, if any key were to fail parsing, `filter_map` would silently remove it from the vector, shifting all subsequent indexes in `account_keys`. The index shifting could have devastating impact: the Gateway ID `program_id_index` would now reference a wrong, possibly malicious account, which could cosplay the emission of a `CallContract` event.

Recommendation

We suggest replacing `filter_map` with `map` and propagating any parsing error, returning `None` from the `tx` method if any public key fails to parse. This makes the failure explicit rather than silently corrupting the indexing.

C6: Misleading function name "is_instruction_from_gateway_program"

Severity: Minor **Status:** Fixed

Location

`ampd-handlers/src/solana/mod.rs`
`is_instruction_from_gateway_program`

Description

The function `is_instruction_from_gateway_program` checks that the instruction's `program_id_index` resolves to the Gateway address in `account_keys`. However, `program_id_index` identifies the callee (the program being invoked), not the caller (the CPI invoker). The function name suggests it verifies that the instruction originates from the Gateway, which would imply checking the caller. Therefore, the naming is misleading and could cause confusion in a future code change or review.

Note: Fortunately, the method `is_instruction_from_gateway_program` also implicitly enforces that the callee is the Gateway, because it checks that the instruction is an Anchor self-CPI event, which validates that the event PDA has signed the transaction, and this can only happen if the Gateway itself is the CPI caller. So checking that the callee is the Gateway implicitly ensures the caller is also the Gateway.

Recommendation

We recommend renaming the function, e.g. to `is_instruction_to_gateway_program`, to more accurately reflect that it checks the callee (the invoked program), not the caller.

For completeness, we also recommend checking that the caller is the Gateway, e.g. through a new, separate helper `is_instruction_from_gateway_program`.

C7: Gateway check unnecessary and inconsistent with other chains

Severity: Minor **Status:** Fixed

Location

```
ampd-handlers/src/bin/solana/handler.rs
    handle
```

Description

In `handle`, the Solana ampd handler checks whether `source_gateway_address` from the poll event matches `self.gateway_address` and skips the poll if they differ. This check is unnecessary and inconsistent with other chain integrations (EVM, Sui, Stellar), which do not perform it.

The check is redundant because:

- The Voting Verifier contract is deployed per chain with the gateway address stored permanently in its configuration. It is not user-controlled, so the `source_gateway_address` in poll events always matches the chain's gateway.
- The handler already filters out polls for other chains via `should_skip_voting` in `VotingHandler::handle`, which checks `source_chain != self.chain()`.

As a consequence, the `gateway_address` field is stored unnecessarily in the `Handler` struct.

Recommendation

We suggest removing the gateway address comparison from `handle` and the `gateway_address` field from the `Handler` struct, to align with the pattern used by other chain integrations.