



XRPL Integration Security Assessment (Phase 3)

Axelar

Version 1.0 – April 22, 2025

1 Executive Summary

Synopsis

During the April of 2025, Axelar engaged NCC Group to conduct a follow-up security assessment of the XRP Ledger's integration with the Axelar Amplifier protocol. This project provides the necessary support for bridging tokens between XRP and connected chains on the Axelar network, and outbound General Message Passing (GMP) from XRPL to other supported chains. One consultant conducted this review in 4 person-days.

Scope

NCC Group's evaluation included the following paths from the <https://github.com/commonprefix/axelar-amplifier> repository on commit [765d82d](#). This phase of the review was guided by the difference between the [latest commit hash](#) and the [previously reviewed commit hash](#):

- XRPL MultiSig Prover: <https://github.com/commonprefix/axelar-amplifier/tree/xrpl/contracts/xrpl-multisig-prover>
- XRPL Gateway: <https://github.com/commonprefix/axelar-amplifier/tree/xrpl/contracts/xrpl-gateway>
- XRPL Voting Verifier: <https://github.com/commonprefix/axelar-amplifier/tree/xrpl/contracts/xrpl-voting-verifier>

In addition the following 2 paths from the <https://github.com/axelarnetwork/axelar-amplifier> repository on commit [90f60bd](#) were reviewed:

- XRPL types: <https://github.com/axelarnetwork/axelar-amplifier/tree/main/packages/xrpl-types>
- XRPL-specific ampd components: <https://github.com/axelarnetwork/axelar-amplifier/tree/main/ampd>

Finally, the [deploy-multisig.js](#) file was reviewed for common programming and security pitfalls.

Limitations

The NCC Group team focused their review on the scope above, paying attention to common programming pitfalls and security issues. The consultant achieved good coverage on the in-scope changes.

Key Findings

The assessment uncovered 4 findings. Notably:

- Finding "Using Deprecated CosmWasm Response Structure"
- Finding "Duplicate Messages Are Still Routed"

Strategic Recommendations

After addressing the findings presented in this review, NCC Group strongly recommends performing a search for all Todos, evaluating their priority, and resolving them.



2 Dashboard

Target Data

Name	XRPL Integration with the Amplifier Protocol
Type	Source Code
Platforms	Rust, JavaScript
Environment	CosmWasm


Engagement Data

Type	Implementation Review
Method	Code-Assisted
Dates	2025-04-15 to 2025-04-21
Consultants	1
Level of Effort	4 person-days





Targets

XRPL MultiSig Prover	https://github.com/commonprefix/axelar-amplifier/tree/xrpl/contracts/xrpl-multisig-prover on commit 765d82d
XRPL Gateway	https://github.com/commonprefix/axelar-amplifier/tree/xrpl/contracts/xrpl-gateway on commit 765d82d
XRPL Voting Verifier	https://github.com/commonprefix/axelar-amplifier/tree/xrpl/contracts/xrpl-voting-verifier on commit 765d82d
XRPL types	https://github.com/axelarnetwork/axelar-amplifier/tree/main/packages/xrpl-types on commit 90f60bd
XRPL-specific ampd components	https://github.com/axelarnetwork/axelar-amplifier/tree/main/ampd on commit 90f60bd
MultiSig deployer script	deploy-multisig.js

Finding Breakdown

Critical issues	0
High issues	0
Medium issues	2 
Low issues	0
Informational issues	2 
Total issues	4

Category Breakdown

Error Reporting	1 
Security Improvement Opportunity	2  
Session Management	1 



Component Breakdown

xrpl-gateway	1	<div></div>
xrpl-multisig-prover	3	<div><div></div><div></div><div></div></div>

Critical High Medium Low Informational



3 Table of Findings

For each finding, NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors.

Title	Status	ID	Risk
Using Deprecated CosmWasm Response Structure	Risk Accepted	XEF	Medium
Duplicate Messages Are Still Routed	Reported	7AT	Medium
MultiSig Prover Contract Should Account for Reserve Fee Volatility	Fixed	YDV	Info
Incorrect Error Code	Reported	X6A	Info



4 Finding Details

Medium

Using Deprecated CosmWasm Response Structure

Overall Risk	Medium	Finding ID	NCC-E010021-XEF
Impact	Undetermined	Component	xrpl-multisig-prover
Exploitability	Undetermined	Category	Security Improvement Opportunity
		Status	Risk Accepted

Impact

Using deprecated API may result in unexpected behavior in the release builds.

Description

The `cosmwasm-std` crate's `SubMsgResponse` struct's [documentation](#) warns that using the `data` field is deprecated as of version 2.0, and new implementations should use the `msg_responses` field instead. The github.com/commonprefix/axelar-amplifier/ repository uses the `cosmwasm-std` crate on version (at least) 2.1.4. Yet, MultiSig contract's `start_multisig_reply()` implementation uses the deprecated field:

```
pub fn start_multisig_reply(deps: DepsMut, reply: Reply) -> Result<Response, ContractError> {
    let config = CONFIG.load(deps.storage)?;
    #[allow(deprecated)]
    // TODO: use `msg_responses` instead when the cosmwasm vm is updated to 2.x.x
    let data = reply
        .result
        .clone()
        .into_result()
        .map_err(ParseReplyError::SubMsgFailure)?
        .data
        .ok_or_else(|| ParseReplyError::ParseFailure("missing reply data".to_owned()))?;
```

If the `submessages` library does not fill the `data` field in the future, the response will be empty, and the MultiSig Prover contract's reply function will return with an error instead of processing the response in the `msg_responses` field. It is important to keep abreast of the latest updates in the dependencies to prevent unexpected behavior.

Recommendation

Process the `msg_responses` field in the response, instead of the `data` field.

Location

<https://github.com/commonprefix/axelar-amplifier/blob/765d82d09e0652d40500c4ed333d37a0c5f02897/contracts/xrpl-multisig-prover/src/contract/reply.rs#L22>

Retest Results

2025-04-22 – Not Fixed

The client indicated that [other MultiSig Prover contracts](#) also use this deprecated API. Since this issue needs to be fixed for all of these implementations, this finding is marked as *Risk Accepted* at this stage.



Duplicate Messages Are Still Routed

Overall Risk Medium

Impact Medium

Exploitability Medium

Finding ID NCC-E010021-7AT

Component xrpl-gateway

Category Session Management

Status Reported

Impact

Failure to filter duplicated messages and payloads may lead to routing an interchain transfer or a call contract message multiple times.

Description

In the following code snippet from the `route_incoming_messages()` function, an interchain transfer or a call contract message are prepared, their gas fee amounts are counted (if they were successful), and at the end a list of successful messages is emitted:

```
pub fn route_incoming_messages(  
    storage: &mut dyn Storage,  
    config: &Config,  
    verifier: &xrpl_voting_verifier::Client,  
    msgs_with_payload: Vec<WithPayload<XRPLMessage>>,  
) -> Result<Response, Error> {  
    // ... variable initialization ...  
    for (status, msgs) in &msgs_by_status {  
        let mut msgs_to_route = Vec::new();  
        for msg in msgs {  
            let message_with_payload = match msg.message.clone() {  
                XRPLMessage::InterchainTransferMessage(interchain_transfer_message) => {  
                    // ... prepare the message ...  
  
                    if status == &VerificationStatus::SucceededOnSourceChain {  
                        state::count_gas(  
                            storage,  
                            &interchain_transfer_message.tx_id,  
                            &token_id,  
                            interchain_transfer_message.gas_fee_amount.clone(),  
                        )  
                        .change_context(Error::State)?;  
                    }  
  
                    message_with_payload  
                }  
                XRPLMessage::CallContractMessage(call_contract_message) => {  
                    // ... prepare the payload and the CallContract message ...  
  
                    if status == &VerificationStatus::SucceededOnSourceChain {  
                        state::count_gas(  
                            storage,  
                            &call_contract_message.tx_id,  
                            &gas_token_id,  
                            call_contract_message.gas_fee_amount.clone(),  
                        )  
                        .change_context(Error::State)?;  
                    }  
                }  
            }  
            msgs_to_route.push(msg);  
        }  
    }  
    emit_successful_messages(msgs_to_route);  
    Result::Ok(Response::new())  
}
```



```

        Some(message_with_payload)
    }
    => {
        return Err(report!(Error::UnsupportedIncomingMessage(
            msg.message.to_owned()
        )))
    }
};

if let Some(MessageWithPayload {
    message: msg,
    payload,
}) = message_with_payload
{
    msgs_to_route.push(msg.clone());
    events.extend(vec![
        into_route_event(status, msg.clone()),
        Event::from(XRPLGatewayEvent::ContractCalled {
            msg,
            payload: payload.into(),
        }),
    ]);
}
route_msgs.extend(filter_routable_messages(*status, &msgs_to_route));
}

let router = Router::new(config.router.clone());
Ok(Response::new()
    .add_messages(router.route(route_msgs))
    .add_events(events))

```

The `count_gas()` function, excerpted below, tracks transactions by their transaction ids (`tx_id`) and returns `Ok()` early without updating the gas counter if a transaction has already been accounted:

```

pub fn count_gas(
    storage: &mut dyn Storage,
    tx_id: &HexTxHash,
    token_id: &TokenId,
    gas: XRPLPaymentAmount,
) -> Result<(), Error> {
    if gas.is_zero() || gas_counted(storage, tx_id)? {
        return Ok(());
    }

    increment_gas(storage, token_id, gas)?;
    mark_gas_counted(storage, tx_id)?;
    Ok(())
}

```

Since the `gas_counted()` function silently returns `Ok()` for duplicated transactions, the caller (`route_incoming_messages()` function) will not catch this error scenario and continue to include the message in the list to be routed.

Severity of this finding may be lower if the router implements a mechanism to deduplicate messages.

Recommendation

Filter incoming messages that have already been routed before sending them to the router.

Location

<https://github.com/commonprefix/axelar-amplifier/blob/765d82d09e0652d40500c4ed33d37a0c5f02897/contracts/xrpl-gateway/src/contract/execute.rs#L56-L150>



MultiSig Prover Contract Should Account for Reserve Fee Volatility

Overall Risk	Informational	Finding ID	NCC-E010021-YDV
Impact	Undetermined	Component	xrpl-multisig-prover
Exploitability	Low	Category	Security Improvement Opportunity
		Status	Fixed

Impact

Failure to account for the XRP Ledger's base reserve fee volatility may result in the MultiSig account's removal from the ledger.

Description

The MultiSig prover contract maintains the base reserve that is required to keep the MultiSig account alive on the XRP Ledger. This amount is modifiable using the `ExecuteMsg::UpdateXrpLReserves` command, which can only be executed by the admin and governance accounts.

It is key to ensure that the `new base reserve` is not set too low, as that may accidentally lead to the removal of the MultiSig account from the ledger.

Historically, to facilitate adoption the base reserve has been lowered over time¹, and is currently at 1 XRP on the mainnet. MultiSig contract owners should monitor this parameter via the `server_info` API and update the configuration as needed. In addition, a relatively recent blog post on the xrpl.org warns that the validators' split over this base value may cause fee volatility, and provides two recommendations:

Instead of hard-coding reserve requirements or transaction costs, look up the necessary settings using the fee API method.

Consider holding more XRP than the minimum, so that your account still meets the requirements if reserves go back up.

The first measure is already implemented. The second measure could be accounted for in the `new_base_reserve` and the `new_owner_reserve` values that the admins set.

Recommendation

(If not already done) consider putting a process in place to monitor the base and owner reserve fees and update the configuration promptly upon any changes. Consider including a cushion to account for the volatility that may be caused by the consensus layer.

Location

<https://github.com/commonprefix/axelar-amplifier/blob/765d82d09e0652d40500c4ed333d37a0c5f02897/contracts/xrpl-multisig-prover/src/contract/execute.rs#L202-L216>

Retest Results

2025-04-22 – Fixed

The client indicated that they have implemented the following measures:

1. <https://xrpl.org/blog/2021/reserves-lowered>, and <https://xrpl.org/blog/2024/lower-reserves-are-in-effect>.



Yeah, we're already fetching reserve information via API: <https://github.com/axelarnetwork/axelar-contract-deployments/blob/fefc8d75047e46ddf143b395e6e48df4cceb6d5/cosmwasm/utils.js#L617-L619>. Changing the reserve fee requires an amendment (i.e., network fork). We'll need to track amendments anyways since a lot of the assumptions we make could change when a new amendment passes.

As such, this finding is marked as fixed.



Incorrect Error Code

Overall Risk Informational

Impact None

Exploitability None

Finding ID NCC-E010021-X6A

Component xrpl-multisig-prover

Category Error Reporting

Status Reported

Impact

Returning an incorrect error message may hinder development.

Description

In the following code snippet from the `xrpl_multisig.rs` file, the result of the `checked_sub()` call may underflow not overflow:

```
pub fn num_of_tickets_to_create(storage: &dyn Storage) -> Result<u32, ContractError> {
    let available_tickets = AVAILABLE_TICKETS.load(storage)?;
    let available_ticket_count = u32::try_from(available_tickets.len())
        .map_err(|_| ContractError::TooManyAvailableTickets)?;

    if available_ticket_count > MAX_TICKET_COUNT {
        return Err(ContractError::TooManyAvailableTickets);
    }

    MAX_TICKET_COUNT
        .checked_sub(available_ticket_count)
        .ok_or(ContractError::Overflow)
}
```

Recommendation

Update the error to `ContractError::Underflow`.

Location

https://github.com/commonprefix/axelar-amplifier/blob/765d82d09e0652d40500c4ed333d37a0c5f02897/contracts/xrpl-multisig-prover/src/xrpl_multisig.rs#L408



5 Finding Field Definitions

The following sections describe the risk rating and category assigned to issues NCC Group identified.

Risk Scale

NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. The risk rating is NCC Group's recommended prioritization for addressing findings. Every organization has a different risk sensitivity, so to some extent these recommendations are more relative than absolute guidelines.

Overall Risk

Overall risk reflects NCC Group's estimation of the risk that a finding poses to the target system or systems. It takes into account the impact of the finding, the difficulty of exploitation, and any other relevant factors.

Rating	Description
Critical	Implies an immediate, easily accessible threat of total compromise.
High	Implies an immediate threat of system compromise, or an easily accessible threat of large-scale breach.
Medium	A difficult to exploit threat of large-scale breach, or easy compromise of a small portion of the application.
Low	Implies a relatively minor threat to the application.
Informational	No immediate threat to the application. May provide suggestions for application improvement, functional issues with the application, or conditions that could later lead to an exploitable finding.

Impact

Impact reflects the effects that successful exploitation has upon the target system or systems. It takes into account potential losses of confidentiality, integrity and availability, as well as potential reputational losses.

Rating	Description
High	Attackers can read or modify all data in a system, execute arbitrary code on the system, or escalate their privileges to superuser level.
Medium	Attackers can read or modify some unauthorized data on a system, deny access to that system, or gain significant internal technical information.
Low	Attackers can gain small amounts of unauthorized information or slightly degrade system performance. May have a negative public perception of security.

Exploitability

Exploitability reflects the ease with which attackers may exploit a finding. It takes into account the level of access required, availability of exploitation information, requirements relating to social engineering, race conditions, brute forcing, etc, and other impediments to exploitation.

Rating	Description
High	Attackers can unilaterally exploit the finding without special permissions or significant roadblocks.



Rating	Description
Medium	Attackers would need to leverage a third party, gain non-public information, exploit a race condition, already have privileged access, or otherwise overcome moderate hurdles in order to exploit the finding.
Low	Exploitation requires implausible social engineering, a difficult race condition, guessing difficult-to-guess data, or is otherwise unlikely.

Category

NCC Group categorizes findings based on the security area to which those findings belong. This can help organizations identify gaps in secure development, deployment, patching, etc.

Category Name	Description
Access Controls	Related to authorization of users, and assessment of rights.
Auditing and Logging	Related to auditing of actions, or logging of problems.
Authentication	Related to the identification of users.
Configuration	Related to security configurations of servers, devices, or software.
Cryptography	Related to mathematical protections for data.
Data Exposure	Related to unintended exposure of sensitive information.
Data Validation	Related to improper reliance on the structure or values of data.
Denial of Service	Related to causing system failure.
Error Reporting	Related to the reporting of error conditions in a secure fashion.
Patching	Related to keeping software up to date.
Session Management	Related to the identification of authenticated users.
Timing	Related to race conditions, locking, or order of operations.



6 Contact Info

The team from NCC Group has the following primary members:

- Parnian Alimi – Consultant
parnian.alimi@nccgroup.com
- Javed Samuel – Practice Director, Cryptography Services
javed.samuel@nccgroup.com

The team from Axelar has the following primary member:

- Nikolas Kamarinakis
nikolas@commonprefix.com

