Security Audit Report

XLink - Solana Endpoint

May 2025

V202411

Executive Summary

CoinFabrik was asked to audit the contracts for **Brotocol's Solana Endpoint** project.

During this audit we found one medium severity issue and one low severity issue. Also, one enhancement was proposed.

The medium issue was fixed and the low issue was acknowledged. The enhancement was implemented.

Scope

The audited files are from the git repository located at https://github.com/Brotocol-xyz/xlink, in the `./packages/contracts/bridge-solana/` directory. The audit is based on the commit `477946ccaf57b15e462182a0725a5fc2868c143c`. Fixes were reviewed on commit `92a0a3f33b0b2b35c0e61670c390dd20077feaaa`.

The scope for this audit includes and is limited to the following files:

- `./programs/bridge-registry/src/lib.rs`: Manages access control roles, token configurations (fees, limits), pause status, and the mint/burn authority for specific bridgeable tokens.
- `./programs/bridge-endpoint/src/lib.rs`: Processes user requests to send tokens, verifying and executing incoming messages from other chains to release tokens.

No other files in this repository were audited. Its dependencies are assumed to work according to their documentation. Also, no tests were reviewed for this audit.

Findings

In the following table we summarize the security issues we found in this audit. The severity classification criteria and the status meaning are explained below. This table does not include the enhancements we suggest to implement, which are described in a specific section after the security issues.

Each severity label is detailed in the Severity Classification section. Additionally, the statuses are explained in the Issues Status section.

Id	Title	Severity	Status
ME-01	Inability to Update Roles when MAX_ROLES is Reached	Medium	**Resolved**
LO-01	Off-Chain Event Processing DoS via Event Spamming	Low	**Acknowledged**

Critical Severity Issues

No issues found.

High Severity Issues

No issues found.

Medium Severity Issues

ME-01 Inability to Update Roles when MAX_ROLES is Reached

Location

- `./programs/bridge-registry/src/lib.rs: 791`

Classification

- CWE-703: Improper Check or Handling of Exceptional Conditions[1]

Description

The `grant_role_internal` function first checks `require!(self.roles.len() < MAX_ROLES, BridgeError::MaxRolesReached);` before attempting to grant a new role. If the number of `RoleEntry` structs in the roles vector is equal to `MAX_ROLES`, this check will fail even if the intention is to grant an additional role to an existing account already in the roles list (which would only modify an existing `RoleEntry.role` bitmask, not increase `self.roles.len()`).

If the roles list is full (i.e., `self.roles.len() == MAX_ROLES`), it becomes impossible to grant new roles to accounts that already have some roles, effectively preventing role updates for existing privileged accounts.

[1] https://cwe.mitre.org/data/definitions/703.html

Recommendation

The check should only apply if a new `RoleEntry` needs to be pushed to the vector.

Status

Resolved. Fixed according to the recommendation.

Low Severity Issues

LO-01 Off-Chain Event Processing DoS via Event Spamming

Location

- `./programs/bridge-endpoint/src/lib.rs: 593-714`

Classification

- CWE-770: Allocation of Resources Without Limits or Throttling[2]

Description

The `bridge_endpoint` program exposes two publicly callable functions, `send_message` and `send_message_with_token`, that emit `SendMessageEvent` and `SendMessageWithTokenEvent` respectively. Any user can call these functions.

While there is an on-chain execution cost for the caller (transaction fees, and token value for `send_message_with_token`), the emitted events are typically consumed by off-chain systems or listeners that react to bridge activities. A malicious actor could repeatedly call these functions to generate a high volume of events.

If the off-chain systems responsible for processing these events are not designed to handle a large volume of events efficiently, or if they incur a significant cost per event processed, then such spamming could lead to increased operational costs, delayed processing for legitimate events, and Denial of Service (DoS).

The severity is reduced due to on-chain contracts themselves correctly validating payload sizes.

Recommendation

Implement monitoring for event volume. Set up alerts for unusual spikes in event creation that could indicate a spam attack.

[2] https://cwe.mitre.org/data/definitions/770.html

Status

Acknowledged.

Enhancements

These items do not represent a security risk. They are best practices that we suggest implementing.

Id	Title	Status
EN-01	Inconsistent Event Emission Pattern	**Implemented**

EN-01 Inconsistent Event Emission Pattern

Location

- `./programs/bridge-registry/src/lib.rs`

Description

Some public-facing functions (like `set_required_validators`) emit events directly within their own body. Other public functions (like `revoke_role` and `grant_role`) delegate their core logic to internal functions (`revoke_role_internal`, `grant_role_internal`), and it's these internal functions that emit the events.

Recommendation

Standardize the event emission pattern. A common best practice is for the public-facing instruction handler to be responsible for emitting events.

Status

Implemented.

Other Considerations

The considerations stated in this section are not right or wrong. We do not suggest any action to fix them. But we consider that they may be of interest to other stakeholders of the project, including users of the audited contracts, token holders or project investors.

Centralization

The system owner possesses significant privileges, including the ability to unilaterally finalize orders in emergencies and to perform administrative actions like pausing the bridge or upgrading critical program addresses. While this is necessary for operational flexibility and recovery, it also means that the compromise or misuse of the owner account could lead to temporary service disruptions.

Upgrades

The registry program is set in the endpoint program and can be replaced by a new one if the owner makes a transaction.

Privileged Roles

These are the privileged roles that we identified on each of the audited contracts.

Owner

This is the highest-level administrative role, controlling the bridge-registry. The owner can:

- Grant/revoke all other roles.
- Set system parameters like required validators.
- Configure approved tokens, their fees, and limits.
- Pause/unpause the bridge system.
- Transfer ownership of the registry.
- Transfer mint authority for burnable tokens.
- Collect accrued fees.
- Update critical addresses in the bridge-endpoint program (like the registry program address, peg-in address).

APPROVED_ROLE

Can call `create_or_mark_order_finalized` in the bridge endpoint, allowing for emergency finalization or creation of orders. This role is typically assigned to the owner or a trusted administrative account.

RELAYER_ROLE

Authorized to submit validated off-chain messages to the bridge endpoint's `transfer_to_unwrap` function to initiate the release of tokens on Solana.

MINTER_ROLE

Authorized to call `mint_tokens` and `burn_tokens` directly on the bridge registry. In the standard flow, the bridge endpoint program PDA acts as the signer with this role when performing CPIs to the registry.

FINALIZER_ROLE

Signs to finalize the release of non-burnable tokens via the `finalize_unwrap` instruction in the bridge endpoint.

About CoinFabrik

CoinFabrik is a research and development company specialized in Web3, with a strong background in cybersecurity. Founded in 2014, we have worked on over 500 decentralization projects, including EVM-based and other platforms like Solana, Algorand, and Polkadot. Beyond development, we offer security audits through a dedicated in-house team of senior cybersecurity professionals, working on code in languages such as Substrate, Solidity, Clarity, Rust, TEAL, and Stellar Soroban.

Our team has an academic background in computer science, software engineering, and mathematics, with accomplishments including academic publications, patents turned into products, and conference presentations. We actively research in collaboration with universities worldwide, such as Cornell, UCLA, and École Polytechnique in Paris, and maintain an ongoing collaboration on knowledge transfer and open-source projects with the University of Buenos Aires, Argentina. Our management and people experience team has extensive expertise in the field.

Methodology

CoinFabrik was provided with the source code, including automated tests that define the expected behavior. Our auditors spent two weeks auditing the source code provided, which includes understanding the context of use, analyzing the boundaries of the expected behavior of each contract and function, understanding the implementation by the development team (including dependencies beyond the scope to be audited) and identifying possible situations in

which the code allows the caller to reach a state that exposes some vulnerability. Without being limited to them, the audit process included the following analyses.

- Arithmetic errors
- Race conditions
- Reentrancy attacks
- Misuse of block timestamps
- Denial of service attacks
- Excessive gas usage
- Missing or misused function qualifiers
- Needlessly complex code and contract interactions
- Poor or nonexistent error handling
- Insufficient validation of the input parameters
- Incorrect handling of cryptographic signatures
- Centralization and upgradeability

After delivering a report with our findings, the development team had the opportunity to comment on every finding and fix the issues they considered convenient. Once fixed and/or commented, our team ran a second review process to verify that the changes to the code effectively solve the issues found and do not unintentionally add new ones. This report includes the final status after the second review.

Severity Classification

Security risks are classified as follows[3]:

Critical	• Manipulation of governance voting result deviating from voted outcome and resulting in a direct change from intended effect of original results • Direct theft of any user funds, whether at-rest or in-motion, other than unclaimed yield • Direct theft of any user NFTs, whether at-rest or in-motion, other than unclaimed royalties • Permanent freezing of funds • Permanent freezing of NFTs • Unauthorized minting of NFTs • Predictable or manipulable RNG that results in abuse of the principal or NFT • Unintended alteration of what the NFT represents (e.g. token URI, payload, artistic content) • Protocol insolvency
High	• Theft of unclaimed yield • Theft of unclaimed royalties • Permanent freezing of unclaimed yield • Permanent freezing of unclaimed royalties • Temporary freezing of funds • Temporary freezing NFTs

[3] This classification is based on the smart contract Immunefi severity classification system version 2.3. https://immunefi.com/immunefi-vulnerability-severity-classification-system-v2-3/

Medium	• Smart contract unable to operate due to lack of token funds • Block stuffing • Griefing (e.g. no profit motive for an attacker, but damage to the users or the protocol) • Theft of gas • Unbounded gas consumption • Security best practices not followed
Low	• Contract fails to deliver promised returns, but doesn't lose value • Other security issues with minor impact

Issue Status

An issue detected by this audit has one of the following statuses:

- **Unresolved:** The issue has not been resolved.
- **Resolved:** Adjusted program implementation to eliminate the risk.
- **Partially Resolved:** Adjusted program implementation to eliminate part of the risk. The other part remains in the code, but is a result of an intentional decision.
- **Acknowledged:** The issue remains in the code, but is a result of an intentional decision. The reported risk is accepted by the development team.
- **Mitigated:** Implemented actions to minimize the impact or likelihood of the risk.

Disclaimer

This audit report has been conducted on a **best-effort basis within a tight deadline defined by time and budget constraints**. We reviewed only the specific smart contract code provided by the client at the time of the audit, detailed in the Scope section. We do not review other components that are part of the solution: neither implementation, nor general design, nor business ideas that motivate them.

While we have employed the latest tools, techniques, and methodologies to identify potential vulnerabilities, **this report does not guarantee the absolute security of the contracts, as undiscovered vulnerabilities may still exist.** Our findings and recommendations are

suggestions to enhance security and functionality and are not obligations for the client to implement.

The results of this audit are valid solely for the code and configurations reviewed, and any modifications made after the audit are outside the scope of our responsibility. CoinFabrik disclaims all liability for any damages, losses, or legal consequences resulting from the use or misuse of the smart contracts, including those arising from undiscovered vulnerabilities or changes made to the codebase after the audit.

This report is intended exclusively for the **XLink** team and should not be relied upon by any third party without the explicit consent of CoinFabrik. Blockchain technology and smart contracts are inherently experimental and involve significant risk; users and investors should fully understand these risks before deploying or interacting with the audited contracts.

Changelog

Date	Description
2025-05-19	Initial report based on commit `477946ccaf57b15e462182a0725a5fc2868c143c`.
2025-05-27	Reaudit report based on the fixes in commit `92a0a3f33b0b2b35c0e61670c390dd20077feaaa`. LO-01 was added.