Echo Bridge II

Audit Report

MOVEBIT

✉

contact@bitslab.xyz

https://twitter.com/movebit_

Sat Jan 18 2025

Echo Bridge II Audit Report

1 Executive Summary

1.1 Project Information

Description	A bridge smart contract to cross-assets.
Type	Bridge
Auditors	MoveBit
Timeline	Wed Jan 15 2025 - Fri Jan 17 2025
Languages	Move
Platform	Aptos,Movement
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/echo-proto/bridge-aptos
Commits	d1466688ba18fef0da3311f0b022fa747d03104f 134748b352cb59b2846720fd529662378de69934

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
BRI	sources/bridge.move	91050c7300ba3db666bba04f8af56c4ae565aaae
TCV2	sources/token_config_v2.move	705084f31b3fdd6dc293f6dfd11d8bf8d9daa98c
TCO	sources/token_config.move	ecb23d2ae1750f714a40c4305a24be786f6c056f
ABT	sources/abtc.move	102c776aeb0d658ca2691b9dd1230c1e41644c87
CON	sources/constants.move	0912bdf88904f563124f88a9bb65d9adf60ab6cf
MES	sources/message.move	17bbe8c9ba16382165c0b09e0439fc0862ef815c
VAU	sources/vault.move	81f67d126ced17f3c57daa46e77dd34723ee629c
BTE	sources/bridgg_test.move	8015fffb463f526118ca2ca790275ea9823f23c9
COM	sources/committee.move	916ec6f2c764b1bea7fe6e26a5a955c57f2243e4
UTI	sources/utils.move	f0b31c0e08f1ee4ae5fb5057371a77485cc7a467
ESV	sources/eth_sig_verifier.move	6fd834f08146b43484a496b6871afaf723ec87bc

ITA	sources/iterable_table.move	43de307b92792962592cf7fef1674c3a05718c89
LIM	sources/limiter.move	23065587185958afa221c8b65b56d10488a6d9c9
CCO	sources/chain_config.move	e3927693cc359e027bcf774e1d0a0217c6518775

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	9	5	4
Informational	0	0	0
Minor	4	1	3
Medium	4	3	1
Major	1	1	0
Critical	0	0	0

1.4 MoveBit Audit Breakdown

MoveBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow by bit operations
- Number of rounding errors
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting
- Unchecked CALL Return Values
- The flow of capability
- Witness Type

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Formal Verification(Optional)

Perform formal verification for key functions with the Move Prover.

(4) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by Echo to identify any potential issues and vulnerabilities in the source code of the Echo Bridge smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 9 issues of varying severity, listed below.

ID	Title	Severity	Status
VAU-1	Incorrect Order of Assignment in Reward Calculation	Major	Fixed
VAU-2	Missing Validation for PrizeCoinType Consistency	Medium	Fixed
VAU-3	Potential Precision Loss	Medium	Acknowledged
VAU-4	Missing Existence Check for User State	Medium	Fixed
VAU-5	Incorrect Event Data in UserUpdated Emission	Medium	Fixed
VAU-6	Unused Constant `ERR_PERIOD_NOT_FOUND`	Minor	Fixed
VAU-7	Lack of Event Emission for Configuration Updates	Minor	Acknowledged
VAU-8	Code Optimization	Minor	Acknowledged
VAU-9	Code Optimization	Minor	Acknowledged

3 Participant Process

Here are the relevant actors with their respective abilities within the Echo Bridge Smart Contract :

Admin

- The Admin can update the submitter through `update_submitter()` .
- The Admin can update the fee receipt through `update_fee_receipt()` .
- The Admin can set the minimum amount through `set_min_amount()` .
- The Admin can pause and unpause the deposit through `set_deposit_paused()` .
- The Admin can pause and unpause the withdraw through `set_withdraw_paused()` .
- The Admin can set new fee percentage through `setFeePercentage()` .
- The Admin can set new token minimum amount through `setTokenMinAmount()` .
- The Admin can set new fee receiver through `setFeeRecipient()` .
- The Admin can set new submitter through `updateSubmitterlist()` .
- The Admin can add the vote power of a committee through `addCommitteeStake()` .

Submitter

- The Submitter can update the committees through `update_committees()` .
- The Submitter can add a new token type through `add_token()` .
- The Submitter can update the limit through `update_limit()` .
- The Submitter can bridge the user's assets of other networks to this bridge and mint to the user through `bridge()` .
- The Submitter can verify the provided message and signatures using the BridgeCommittee contract through `verifyMessageAndSignatures()` .

User

- The User can withdraw their assets of this network and burn from the user through `withdraw()` .
- The User can claim rewards through `claim()` .

4 Findings

VAU-1 Incorrect Order of Assignment in Reward Calculation

Severity: Major

Status: Fixed

Code Location:

sources/vault.move#390

Descriptions:

The line `user_reward.claimed_amount = user_reward.accumulated_prize;` pre-assigns `user_reward.claimed_amount` to `user_reward.accumulated_prize` before calculating the `period.claimed_prize`. As a result, the calculation `period.claimed_prize + (user_reward.accumulated_prize - user_reward.claimed_amount)` always evaluates to zero, leading to incorrect updates of the `period.claimed_prize`.

Suggestion:

It is recommended to reorder the operations to ensure `user_reward.claimed_amount` is updated after `period.claimed_prize` is calculated.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

VAU-2 Missing Validation for PrizeCoinType Consistency

Severity: Medium

Status: Fixed

Code Location:

sources/vault.move#284

Descriptions:

The `add_prize()` function allows depositing reward tokens of a specific type (`PrizeCoinType`) into the vault. However, there is no validation to ensure that the `PrizeCoinType` matches the vault's predefined `prize_coin_type` . This could lead to a situation where tokens of an incorrect type are deposited, potentially causing inconsistency or operational issues within the vault.

Suggestion:

Add a validation step to check that the `PrizeCoinType` matches the vault's `prize_coin_type` before allowing the deposit. If they do not match, abort the operation. Example of validation logic:

```
assert!(type_info::type_name<PrizeCoinType>() == vault.prize_coin_type,
ERR_INVALID_PRIZE_TOKEN);
```

Resolution:

This issue has been fixed. The client has adopted our suggestions.

VAU-3 Potential Precision Loss

Severity: Medium

Status: Acknowledged

Code Location:

sources/vault.move#548

Descriptions:

In the `calculate_prize()` method, in certain scenarios, such as when the elapsed time is too short, the reward calculation may result in 0.

Suggestion:

It is recommended to add a coefficient to improve precision.

VAU-4 Missing Existence Check for User State

Severity: Medium

Status: Fixed

Code Location:

sources/vault.move#476

Descriptions:

The statement `let user_state = smart_table::borrow_mut(&mut vault.users_state, user);` assumes the user key exists in the `vault.users_state table`. If the key does not exist, this could result in a runtime error. Missing a pre-check for the existence of the key makes the code fragile and prone to unexpected crashes.

Suggestion:

It is recommended to perform a check to ensure the user key exists in the `vault.users_state` table before borrowing the mutable reference.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

VAU-5 Incorrect Event Data in UserUpdated Emission

Severity: Medium

Status: Fixed

Code Location:

sources/vault.move#467

Descriptions:

The `UserUpdated` event emission contains incorrect data. When the event is emitted, the variables `new_reward_amount` and `new_total_amount` are set to zero, which does not reflect the actual state of the user or vault.

```
else {
        // The user first deposit.
        smart_table::add(&mut vault.users_state, user, UserState {
          total_amount: amount,
          reward_amount,
          last_updated,
        });
      }
```

When the following code is executed, `new_reward_amount` and `new_total_amount` are both 0, which will cause errors in subsequent event releases.

Suggestion:

It is recommended to check the status assignment of this part of the code:

```
else {
        // The user first deposit.
        smart_table::add(&mut vault.users_state, user, UserState {
          total_amount: amount,
          reward_amount,
          last_updated,
        });
        }
//Assign values to new_reward_amount and new_total_amount
```

Resolution:

This issue has been fixed. The client has adopted our suggestions.

VAU-6 Unused Constant `ERR_PERIOD_NOT_FOUND`

Severity: Minor

Status: Fixed

Code Location:

sources/vault.move#24

Descriptions:

The constant `ERR_PERIOD_NOT_FOUND` is defined but not used anywhere in the codebase. Unused constants introduce unnecessary clutter, making the code harder to maintain and understand.

Suggestion:

Remove the unused constant if it is not intended to be used in the future. Alternatively, if the constant is part of a planned feature, add comments to indicate its purpose and when it will be implemented.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

VAU-7 Lack of Event Emission for Configuration Updates

Severity: Minor

Status: Acknowledged

Code Location:

sources/vault.move#304,314,324

Descriptions:

The functions `update_total_cap()`, `update_user_cap()`, and `update_claim_after_period()` are critical for managing vault configurations, including total cap, user cap, and claim-after-period settings. However, these functions do not emit events to reflect these state changes. The absence of event emissions makes it difficult to trace and monitor these significant updates, hindering transparency and accountability in the contract's operations.

Suggestion:

Emit events for each of these functions to log critical state changes. Include details such as the `vault_id`, previous and new values of the updated parameters, and the admin who initiated the change.

VAU-8 Code Optimization

Severity: Minor

Status: Acknowledged

Code Location:

sources/vault.move#334

Descriptions:

According to the logic, all periods are consecutive. Therefore, when a condition is not met for one period, the subsequent periods will also not meet the condition. Hence, a `break` can be used instead of `continue`.

Suggestion:

It is recommended to use `break` instead of `continue`.

VAU-9 Code Optimization

Severity: Minor

Status: Acknowledged

Code Location:

sources/vault.move#415

Descriptions:

Lines 434-440 ensure that when increasing the reward_amount, it cannot exceed vault.total_cap. After verifying user_cap, the reward_amount will only decrease further, making it even less likely to exceed vault.total_cap.

Suggestion:

It is recommended to replace lines 457-462 with:

```
user_state.reward_amount = user_state.reward_amount + reward_amount;
```

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.

MOVEBIT VERIFIED
MOVEBIT