Echo Bridge Movement

Audit Report

contact@bitslab.xyz

https://twitter.com/movebit_

Tue Nov 26 2024

Echo Bridge Movement Audit Report

1 Executive Summary

1.1 Project Information

Description	A bridge smart contract to cross-assets to Movement.
Type	Bridge
Auditors	MoveBit
Timeline	Wed Nov 20 2024 - Fri Nov 22 2024
Languages	Move
Platform	Movement
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/echo-proto/bridge-movement
Commits	aa00f6f087880b8769cf2f78e5bdb2cc23167f5c 1d8ff0621c1a6ee21aa672312414ff4950f2a672 67b0d63d425c6cf70909de6c8030e7396e09b22b 77b6039bc1427141528a5c07a760e84bad1396a5

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	d5c2168f82af3c547523db3c2c8200e8c2fb9c98
TCO	sources/token_config.move	ab75ee49f9bf2b84aee93df3f8a2b422cb353834
CON	sources/constants.move	6a032f0eb9cff8f3d132534f825fe3427775833a
BTE	sources/bridge_test.move	7b4d3d923e0819a8209333b8c21813605fe75b58
MES	sources/message.move	3460caf113d72f33d6f9315d15bccb41923f2e18
EBT	sources/ebtc.move	0783c30c9bee9f52bc1c075bf1d32e1a274b8192
COM	sources/committee.move	a849c8646de5ec20b7b42c4d567ac8450bcc1fc0
UTI	sources/utils.move	310781562a2be94fefe1b4ecfd2444187e036087
ESV	sources/eth_sig_verifier.move	fa263b91920be06ae7dcbd952503008daed6a8a5
ITA	sources/iterable_table.move	43de307b92792962592cf7fef1674c3a05718c89
LIM	sources/limiter.move	23065587185958afa221c8b65b56d10488a6d9c9

CFA	sources/coin_factory.move	6f7917fdbdb3c1b1ca56e65b2914e22177762c88
CCO	sources/chain_config.move	87b1adac2556287e75f0dd71c0bf8feeb992ede3
BRI	sources/bridge.move	5dddb15de547e261d9651595715b01233d536c9d
CON	sources/constants.move	41da1695a96cf87842a6032b64262201104bc6ec
MOV1	sources/movebtc.move	3b0fbc05f19723e00cfe35a5e45664989a0d24c7
BTE	sources/bridge_test.move	9ffa41cedecb3e6005be099b0307567b6881395b

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	5	4	1
Informational	1	1	0
Minor	2	2	0
Medium	1	0	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 Protocol to identify any potential issues and vulnerabilities in the source code of the Echo Bridge Smart Contract 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 5 issues of varying severity, listed below.

ID	Title	Severity	Status
BRI-1	`chain_state_update()` and `token_state_update()` Function Lacks Permission Check	Major	Fixed
BRI-2	Error Code Optimization	Minor	Fixed
BRI-3	`init_chain_mint` Function Visibility Useless	Minor	Fixed
BRI-4	Unused Code	Informational	Fixed
EBT-1	Centralization Risk	Medium	Acknowledged

3 Participant Process

Here are the relevant actors with their respective abilities within the Echo Bridge Smart Contract 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 set the fee through `set_fee()` .
- 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 pause the crossing chain function based on the chain id through `chain_state_update()` .
- The admin can set a new token minimum amount through `setTokenMinAmount()` .
- The admin can set a new fee receiver through `setFeeRecipient()` .
- The admin can set a new submitter through `updateSubmitterlist()` .
- The admin can add the voting 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()` .

User

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

4 Findings

BRI-1 `chain_state_update()` and `token_state_update()` Function Lacks Permission Check

Severity: Major

Status: Fixed

Code Location:

sources/bridge.move#223

Descriptions:

The `chain_state_update()` and `token_state_update()` function lacks permission checks, and the parameters can modify the suspended state of the cross-chain and token contract, which will put the contract at risk.

```
// chain start or suspend
public entry fun chain_state_update(chain_type: u8, paused: bool) acquires Bridge {
    let bridge_res = borrow_global_mut<Bridge>(@echo);
    assert!(
        simple_map::contains_key(&bridge_res.chain_configs, &chain_type),
        error::invalid_argument(ERR_BRIDGE_NOT_EXISTENT_CHAIN_TYPE)
    );
    set_chain_paused(
        simple_map::borrow_mut(&mut bridge_res.chain_configs, &chain_type),
        paused
    )
}
...
    public entry fun token_state_update(chain_type: u8, token_type: u8, paused: bool)
acquires Bridge {

        let bridge_res = borrow_global_mut<Bridge>(@echo);
        chain_config::set_token_state(&mut bridge_res.chain_configs, chain_type,
token_type, paused);
    }
```

Suggestion:

It is recommended to add permission check for admin, add `admin() == signer::address_of(account)` .

Resolution:

The customer took our advice and fixed the issue.

BRI-2 Error Code Optimization

Severity: Minor

Status: Fixed

Code Location:

sources/bridge.move#339,683

Descriptions:

The error code does not match the description or reference.

```
assert!(seq_num == expected_seq_num, ERR_BRIDGE_UNEXPECTED_SEQ);
assert!(exists<ChainTotalMint>(@echo),
error::invalid_argument(ERR_BRIDGE_NOT_INIT_CHAIN_MINT),
);
```

Suggestion:

It is recommended to change the error code to:

`error::invalid_argument(ERR_BRIDGE_UNEXPECTED_SEQ)` and

`error::not_found(ERR_BRIDGE_NOT_INIT_CHAIN_MINT)` .

Resolution:

The customer took our advice and fixed the issue.

BRI-3 `init_chain_mint` Function Visibility Useless

Severity: Minor

Status: Fixed

Code Location:

sources/bridge.move#607

Descriptions:

The `init_chain_mint` function does not do any other setup after it is called. It is used to initialize the function `init_module()`.

Suggestion:

It is recommended to make the `init_chain_mint()` function a private function.

BRI-4 Unused Code

Severity: Informational

Status: Fixed

Code Location:

sources/bridge.move#36,44

Descriptions:

The code is not used after being defined and does not participate in the execution of the contract.

```
const ERR_BRIDGE_INVALID_TOKEN_TYPE: u64 = 13;
const ERR_BTC_TYPE_EXISTENT: u64 = 21;
const ERR_BTC_TYPE_NOT_EXISTENT: u64 = 22;
const ERR_BTC_SYMBOL_EXISTENT: u64 = 23;
```

Suggestion:

It is recommended to delete the unused code.

EBT-1 Centralization Risk

Severity: Medium

Status: Acknowledged

Code Location:

sources/ebtc.move#45

Descriptions:

Centralization risk was identified in the smart contract.

- 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 set the fee through `set_fee()`.
- 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 pause the crossing chain function based on the chain id through `chain_state_update()`.
- The admin can set a new token minimum amount through `setTokenMinAmount()`.
- The admin can set a new fee receiver through `setFeeRecipient()`.
- The admin can set a new submitter through `updateSubmitterlist()`.
- The admin can add the voting power of a committee through `addCommitteeStake()`.

Suggestion:

It is recommended to take measures to reduce the risk of centralization.

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.