Echo Bridge Smart Contract

Audit Report

contact@bitslab.xyz

https://twitter.com/movebit_

Wed Jul 24 2024

Echo Bridge Smart Contract Audit Report

1 Executive Summary

1.1 Project Information

Description	A cross-chain bridge project.
Type	Bridge
Auditors	MoveBit
Timeline	Thu Jul 18 2024 - Fri Jul 19 2024
Languages	Move, Solidity
Platform	Aptos,Ethereum
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/echo-proto/bridge-aptos https://github.com/echo-proto/bridge-b2
Commits	https://github.com/echo-proto/bridge-aptos: 4b87863f1b1a4b9398b16a712a4d74741810c52d 24f31c8bc98b2611c670991c511b0bbd9e92355e 75366c45ced2de86842def697f734c9865f9ec62 https://github.com/echo-proto/bridge-b2: 9fc4516477b41f2ea6d2b1bb3f84d1a98e374906 b1a099227f4b031cdd4fddf487cfca4cc78a2276 64a1fadfd00b1e3c9038d6dcfb0a4fde05f42a41

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MOV	Move.toml	fe53c30a4c9dd02590054f5f69e775dd8cb8e29e
BRI	sources/bridge.move	b4cb8eb378d6ee89bdbe7cb4e6ebd525811ac767
TCO	sources/token_config.move	ecb23d2ae1750f714a40c4305a24be786f6c056f
ABT	sources/abtc.move	16e524f1cb139e80a27d10630bd6416105c15efd
CON	sources/constants.move	f4fca4f5ef7ba79ab7c4b42c45ae970c34c906c0
MES	sources/message.move	17bbe8c9ba16382165c0b09e0439fc0862ef815c
COM	sources/committee.move	b047aee662d54c0307ed2279d15e4f7a1a7ddfde
UTI	sources/utils.move	5b54203fb96eb559ba611d2b744885fd86a1db44
ESV	sources/eth_sig_verifier.move	6fd834f08146b43484a496b6871afaf723ec87bc
ITA	sources/iterable_table.move	43de307b92792962592cf7fef1674c3a05718c89
LIM	sources/limiter.move	dd47d57649b78cff5ecf05b91e95aa81608aa48c

BLI	contracts/BridgeLimiter.sol	9f87bca758c64b62b7de4e7d74a3d36c45aa2c41
BCO	contracts/BridgeCommittee.sol	c7745bc7de5f30308b7043a6273c4901229616ee
MVE	contracts/utils/MessageVerifier.sol	a56cd6481881b2184bf7356f318de9532c3e0204
CUP	contracts/utils/CommitteeUpgradeable.sol	bc6de563a7fe5d2772b5203da30c788c9c98c45e
BUT	contracts/utils/BridgeUtils.sol	839980343574b473bdc1bdfebf474b18d3928333
BVA	contracts/BridgeVault.sol	3cb2af8ad97b7a25388acdf1d83f8d9b32787f43
BCO1	contracts/BridgeConfig.sol	8ab92f39c2532bc499ce2d9dab60ce4fba26ecb8
IBL	contracts/interfaces/IBridgeLimiter.sol	946a99029b3f76b318b06ca45ba2cab9050166c7
IBC	contracts/interfaces/IBridgeConfig.sol	c7e038017c181bf67b8ac19e6476d6db2216a04c
IBV	contracts/interfaces/IBridgeVault.sol	6380077f22f11b5ab39339bfc0f62d6803ce9bb9
IEB	contracts/interfaces/IEchoBridge.sol	33f2661e069537f4ce79fc2c967f37b24e0c8cb5
IWBTC9	contracts/interfaces/IWBTC9.sol	5114eac50367110ec3c54d7ace209be527ff471f
IBC1	contracts/interfaces/IBridgeCommittee.sol	4e98dcd8ef07ee55c1eafb8c8d7adf503d1f09ff

EBR	contracts/EchoBridge.sol	7b1e8a6c18dc7ea05997cc88161415d356c24bf3

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	7	7	0
Informational	1	1	0
Minor	5	5	0
Medium	1	1	0
Major	0	0	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

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, 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 7 issues of varying severity, listed below.

ID	Title	Severity	Status
BCO-1	Lack of Events Emit	Minor	Fixed
BLI-1	Lack of Token Type Mapping	Minor	Fixed
BRI-1	Invalid `token_type` Assertion	Minor	Fixed
BRI-2	Redundant Token Type Check	Minor	Fixed
COM-1	Incorrect Comment	Informational	Fixed
EBR-1	Repeatedly Throwing Same Exceptions	Minor	Fixed
LIM-1	First Transaction Amount Not Recorded	Medium	Fixed

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 minimun amount through `setTokenMinAmount()` .
- The admin can set new fee receiver throguh `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 verifies 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()` .

4 Findings

BCO-1 Lack of Events Emit

Severity: Minor

Status: Fixed

Code Location:

contracts/BridgeConfig.sol#159,151

Descriptions:

The contract lacks appropriate events for monitoring `setFeePercentage()` and `setFeeRecipient()` operations, which could make it difficult to track sensitive actions or detect potential issues.

Suggestion:

It is recommended to emit events for these functions.

Resolution:

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

BLI-1 Lack of Token Type Mapping

Severity: Minor

Status: Fixed

Code Location:

contracts/BridgeLimiter.sol#22 23

Descriptions:

These mappings lack the ability to determine the type of token, which may result in the subsequent cross-chain circulation restrictions not being able to be constrained based on the token type.

```
// total limit in USD (8 decimal precision) (e.g. 1000_00000000 => 1000 USD)
    mapping(uint8 chainID => uint128 totalLimit) public chainLimits;
    mapping(uint8 chainID => uint32 oldestHourTimestamp) public oldestChainTimestamp;
```

Suggestion:

It is recommended to add each token type as a key.

Resolution:

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

BRI-1 Invalid `token_type` Assertion

Severity: Minor

Status: Fixed

Code Location:

sources/bridge.move#440

Descriptions:

In the `withdraw` function, the check for `token_type` occurs after the fee calculation. However, during the fee calculation, `fee(token_type)` retrieves the `token_configs` corresponding to `token_type`. If there are no `token_configs` for the given `token_type`, it will abort, rendering the subsequent check for `token_type` ineffective.

```
    let fee = ((original_amount as u128) * (fee(token_type) as u128) / (FEE_DENOMINATOR as u128) as u64);
    let amount = original_amount - fee;

    let bridge_res = borrow_global_mut<Bridge>(@echo);
    assert!(simple_map::contains_key(&bridge_res.token_configs, &token_type), ERR_BRIDGE_INVALID_TOKEN_TYPE);
```

Suggestion:

It is recommended to check the `token_type` before the fee calculation.

Resolution:

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

BRI-2 Redundant Token Type Check

Severity: Minor

Status: Fixed

Code Location:

sources/bridge.move#409

Descriptions:

In the `bridge` function, the `token_type` is checked before transferring tokens to the user under the `token_type == constants::ubtc_token_type()` branch with the following assertion:

```
assert!(
    token_type == constants::token_ubtc(),
    error::invalid_argument(ERR_BRIDGE_INVALID_TOKEN_TYPE)
);
```

However, since this check is inside the branch and `constants::ubtc_token_type()` is equal to `constants::token_ubtc()`, we consider this check to be redundant.

Suggestion:

It is recommended to confirm it aligns with your design.

Resolution:

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

COM-1 Incorrect Comment

Severity: Informational

Status: Fixed

Code Location:

sources/committee.move#100

Descriptions:

There is an incorrect comment in the `update_committees` function. The comment on line 100 reads `make sure pub key is part of the committee`, but this section is actually for adding a new pub key to the committee. It should ensure that the new pub key is not already in the committee.

```
// and make sure pub key is part of the committee
assert!(!simple_map::contains_key(&self.members, &pubkey), ERR_INVALID_SIGNATURE);
```

Suggestion:

It is recommended to modify the comment as: `make sure pub key is not part of the committee`.

Resolution:

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

EBR-1 Repeatedly Throwing Same Exceptions

Severity: Minor

Status: Fixed

Code Location:

contracts/EchoBridge.sol#257

Descriptions:

Repeatedly throwing exceptions in this function

```
function _transferTokensFromVault(
    uint8 sendingChainID,
    uint8 tokenID,
    address recipientAddress,
    uint256 amount
) private whenNotPaused limitNotExceeded(sendingChainID, amount) {
    //...
    // update amount bridged
    limiter.recordBridgeTransfers(sendingChainID, amount);
}
```

In both the modifier `limitNotExceeded` and the function `recordBridgeTransfers()`, the following exception checking is performed, which here leads to code duplication

```
require(
        !willAmountExceedLimit(chainID, amount),
        "BridgeLimiter: amount exceeds rolling window limit"
    );
```

Suggestion:

It is recommended to optimise this code for your needs.

Resolution:

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

LIM-1 First Transaction Amount Not Recorded

Severity: Medium

Status: Fixed

Code Location:

sources/limiter.move#53

Descriptions:

The function `amountUnderLimit` is intended to determine whether the transaction amount of the same token on the same chain exceeds the limit within a day. However, during the first transaction, the transaction amount is not recorded, which allows the transaction amount of the same token to exceed the limit within the same day.

```
if (!simple_map::contains_key(&self.transfer_records, &key)) {
        simple_map::add(&mut self.transfer_records, key, 0);
    }
```

Suggestion:

It is recommended to add the amount of the first transaction into the `transfer_records`.

Resolution:

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

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.