DuckChain Bridge
Audit Report

Tue Oct 15 2024

contact@bitslab.xyz https://twitter.com/scalebit_

ScaleBit

DuckChain Bridge Audit Report

1 Executive Summary

1.1 Project Information

Description	DuckChain is the Layer2 cross-chain bridge of EVM
Type	L2
Auditors	ScaleBit
Timeline	Sun Sep 29 2024 - Fri Oct 11 2024
Languages	Solidity
Platform	Ethereum
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/DuckChainTonL2/duck-bridge-contract
Commits	23ee96a7817dac19c4609adf9d5df96ba9e76d36 b0e51257b1382cf9f0397367ca70fb239f189c29 78dd34c1bc470408c4e9b8891050792fef09c5ce a20a7b35bbe26b6399b0e6f23e76b2e73de5f600

1.2 Files in Scope

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

ID	File	SHA-1 Hash
L2B	contracts/Layer2Bridge.sol	81d96fbb4ccf341fa2c0daf501179e5f54431338
IL2BERC2	contracts/interfaces/ILayer2BridgeERC20.sol	48a8a8ae5ca30c89380ccbae960cb05bd76e4cfd
ERC2TW	contracts/ERC20TokenWrapped.sol	658fce0e9df49f43d0529dfae6015c1c55dd700b
L2BERC2	contracts/Layer2BridgeERC20.sol	89e46fa649566fb00b9522f5adc4741014494f48

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	3	3	0
Informational	0	0	0
Minor	3	3	0
Medium	0	0	0
Major	0	0	0
Critical	0	0	0

1.4 ScaleBit Audit Breakdown

ScaleBit 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
- Number of rounding errors
- Unchecked External Call
- Unchecked CALL Return Values
- Functionality Checks
- Reentrancy
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic issues
- Gas usage
- Fallback function usage
- tx.origin authentication
- Replay attacks
- Coding style issues

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) 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 DuckChain Bridge to identify any potential issues and vulnerabilities in the source code of the DuckChain 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 3 issues of varying severity, listed below.

ID	Title	Severity	Status
L2B-1	Redundant Code	Minor	Fixed
L2B-2	`proposeAdminAddressList` Not Removed	Minor	Fixed
L2B-3	Unnecessary Check	Minor	Fixed

3 Participant Process

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

Deployer

- Deployer initializes the `__AccessControl_init()/__Pausable_init()` function through the `initialize()` function, sends the `DEFAULT_ADMIN_ROLE` permission to the deployer through the `_grantRole()` function, and sets the `SUPER_ADMIN` permission to `_superAdminAddress` .

Admin

- superAdminAddress can set `superAdminAddress` via `setSuperAdminAddress` function.
- superAdminAddress can set `normalAdminAddress` via `setNormalAdminAddress` function.
- superAdminAddress and normalAdminAddress can add `proposeAdminAddressSupported` via `addProposeAdminAddress` function.
- superAdminAddress and normalAdminAddress can delete `proposeAdminAddressSupported` via `delProposeAdminAddress` function.
- superAdminAddress and normalAdminAddress can add `reviewAdminAddressSupported` via `addReviewAdminAddress` function.
- superAdminAddress and normalAdminAddress can delete `reviewAdminAddressSupported` via `delReviewAdminAddress` function.
- superAdminAddress and normalAdminAddress can add `tokenWrappedAddress` via `addERC20TokenWrapped` function.
- superAdminAddress and normalAdminAddress can set the block time limit through the `setBlockTimeLimit` function.
- `proposeAdminAddressSupported` can propose bridge transaction proposals through the `propose` function.
- `reviewAdminAddressSupported` can review and execute bridge transaction proposals through the `review` function.
- superAdminAddress and normalAdminAddress can set the pause state through the `pause/unpause` function.

- superAdminAddress can set the handling fee through the `setBridgeSettingsFee` function.
- superAdminAddress and normalAdminAddress can set the blacklist token address through the `setBlackListERC20Token` function.

User

- User can initiate a withdrawal request by destroying the `bridgeERC20Address` token through the `burnERC20Token` function.
- User can initiate a `withdrawId` request by transferring Native tokens through the `lockNativeToken` function.
- User can execute a `withdrawId` request through the `claim` function and emit an event based on the value recorded by `withdrawId`.
- User can get the bridge fee through the `getBridgeFee` function.

4 Findings

L2B-1 Redundant Code

Severity: Minor

Status: Fixed

Code Location:

contracts/Layer2Bridge.sol#517

Descriptions:

The permission verification `reviewAdminAddressSupported[msg.sender]` is performed in both the `review` function and the mintERC20Token internal function. It does not cause security issues, but it is an unnecessary check because the Layer2Bridge.mintERC20Token function is only used in `review`.

```
function review(
    uint256 proposalId,
    uint256 chainId,
    bytes32 txHash,
    bool ifProposalValid
) public {
    require(reviewAdminAddressSupported[msg.sender], "Illegal permissions");
...
else {
  mintERC20Token(
      proposal.txHas
      proposal.token
      proposal.to,
      proposal.amoun
  );
...
    function mintERC20Token(
        bytes32 txHash,
        address token,
        address to,
        uint256 amount
    ) internal whenNotPaused {
        require(reviewAdminAddressSupported[msg.sender], "Illegal permissions");
```

Suggestion:

It is recommended to remove duplicate permission checks.

L2B-2 `proposeAdminAddressList` Not Removed

Severity: Minor

Status: Fixed

Code Location:

contracts/Layer2Bridge.sol

Descriptions:

`proposeAdminAddressList` and `reviewAdminAddressList` are not deleted in `delProposeAdminAddress` and `delReviewAdminAddress` functions.

```
proposeAdminAddressList.push(_account);
```

Suggestion:

It is recommended to delete the corresponding array.

Resolution:

The variables are deleted accordingly in the fix code.

```
for (uint256 i = 0; i < length; i++) {
        if (reviewAdminAddressList[i] == _account) {
          // Move the last element to the place of the element to delete
          reviewAdminAddressList[i] = reviewAdminAddressList[length - 1];
          // Remove the last element
          reviewAdminAddressList.pop();
          break;
        }
      }
```

L2B-3 Unnecessary Check

Severity: Minor

Status: Fixed

Code Location:

contracts/Layer2Bridge.sol#271

Descriptions:

there are duplicate checks in the `delProposeAdminAddress` and `delReviewAdminAddress` functions There is no need to check `==true` .

```
    require(
        reviewAdminAddressSupported[_account] == true,
        "Current address is not exist"
    );
```

Suggestion:

It is recommended to remove duplicate checks.

Resolution:

Fix duplicate check, and change to AccessControlUpgradeable library.

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.