BuyBurnBonus
Audit Report

Thu Jul 18 2024

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

ScaleBit

BuyBurnBonus Audit Report

1 Executive Summary

1.1 Project Information

Description	The BBB Double Pump protocol is a new protocol that supports 314 contract-to-coin transactions as well as Swap transactions.
Type	DeFi
Auditors	ScaleBit
Timeline	Mon Jul 15 2024 - Thu Jul 18 2024
Languages	Solidity
Platform	BSC
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/BBBCOIN77/bbb
Commits	486261b74a5008aed91e530bb511620f178e87b2

1.2 Files in Scope

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

ID	File	SHA-1 Hash
BBB	BBB.sol	b89977ea3d25d98456ac380b92a78c272f971662

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	3	0	3
Informational	0	0	0
Minor	1	0	1
Medium	0	0	0
Major	2	0	2
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 mike to identify any potential issues and vulnerabilities in the source code of the BuyBurnBonus 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
BBB-1	The Swap Operation Lacks Slippage Protection	Major	Acknowledged
BBB-2	Centralization Risk	Major	Acknowledged
BBB-3	`autoReleaseFrom` Should Directly Return If `elapsedPeriods` = 0	Minor	Acknowledged

3 Participant Process

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

Owner

- The owner can call the `setRatio` function to set various ratios for different fee types and actions.
- The owner can call the `withdrawOtherAssets` function to withdraw other assets from the contract.
- The owner can call the `batchGrantRole` and `batchRevokeRole` functions to manage roles in batch.

Users

- Users can call the `autoReleaseFrom` function to trigger the auto-release mechanism if conditions are met.
- Users can call the `getReserves` function to view the amount of ETH and tokens in the contract.
- Users can call the `getAmountOut` function to estimate the amount of tokens or ETH to receive when buying or selling.

Liquidity Providers

- Liquidity providers can call the `addLiquidity` function to add liquidity to the contract.
- Liquidity providers can call the `removeLiquidity` function to remove liquidity from the contract once the liquidity lock period has passed.
- Liquidity providers can call the `extendLiquidityLock` function to extend the liquidity lock period.

4 Findings

BBB-1 The Swap Operation Lacks Slippage Protection

Severity: Major

Status: Acknowledged

Code Location:

BBB.sol#1505-1531

Descriptions:

The protocol swaps ETH to another token Within the functions `_buy` and `_sell`. The amount of tokens that are swapped out is calculated by

```
if (buy_) {
        return (value * reserveToken) / (reserveETH + value);
    } else {
        return (value * reserveETH) / (reserveToken + value);
    }
```

However, there isn't any slippage protection in either the `_buy` or `_sell` function. It means that users may experience a high volatile of price change during the swap execution. Consequently, the protocol may be vulnerable to sandwich attacks.

Suggestion:

It is recommended to set the minimum acceptable quantity in the parameters and compare it after the swap is completed.

Resolution:

It is acknowledged by the dev team.

BBB-2 Centralization Risk

Severity: Major

Status: Acknowledged

Code Location:

BBB.sol#1471-1478;

BBB.sol#1801-1806

Descriptions:

The provided functions `setRatio` and `removeLiquidity` exhibit centralized risks due to their reliance on specific roles for executing critical operations. This can lead to single points of failure or potential abuse of power.

In the event of an administrator's private key being compromised or malicious intent, substantial losses could occur.

Suggestion:

Implement measures to mitigate the centralized risk, such as:

- Multi-Signature Wallets: Require multiple signatures for executing the privileged function. This distributes control and reduces the risk associated with a single point of failure.
- Timelocks: Introduce a timelock mechanism that delays the execution of the privileged function, allowing time for stakeholders to review and potentially intervene.

Resolution:

It is acknowledged by the dev team.

BBB-3 `autoReleaseFrom` Should Directly Return If `elapsedPeriods` = 0

Severity: Minor

Status: Acknowledged

Code Location:

BBB.sol#1712

Descriptions:

In the `autoReleaseFrom` function, `elapsedPeriods` is calculated based on the timestamp, and related `fee` is taken if there are some `elapsedPeriods`.

However, it lacks the check of `elapsedPeriods == 0`; if so, the `autoReleaseFrom` should return directly.

Suggestion:

Adding the check and skipping the following logic is suggested if `elapsedPeriods == 0`.

Resolution:

It is acknowledged by the dev team.

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.