Bido

Audit Report

Thu Mar 21 2024

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

ScaleBit

Bido Audit Report

1 Executive Summary

1.1 Project Information

Description	Bido is an BEVM liquid staking protocol.
Type	Staking
Auditors	ScaleBit
Timeline	Thu Mar 14 2024 - Thu Mar 21 2024
Languages	Solidity
Platform	BEVM
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/Bidohub/bido
Commits	871305d74404f09389fff6108ad95c5424841c61

1.2 Files in Scope

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

ID	File	SHA-1 Hash
BID	contracts/Bido.sol	f34bcc640529d3ad799522f0301ed552329ea8ce
WBTC	contracts/WstBTC.sol	8d51a4b6b3f126b5de3d692a7d61bc1a5736ebe7
MBI	contracts/mocks/MockBido.sol	659423a9ace70055bf988d971138a851d6b3c0e1
SBTC	contracts/StBTC.sol	262d14dcdee8cc088e26ec29d314084eb96c82e1
IEIP7SBTC	contracts/interfaces/IEIP712StBTC.sol	8483aa59942fc45bbf067f3ec9f1628b72318870
UST	contracts/common/UnstructuredStorage.sol	c68a393a29a417763eabe06e9d7e29418274f044
M25	contracts/common/Math256.sol	fe9972b0d0069eda3833d995c4dcec65cd8f2f2e

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	6	6	0
Informational	0	0	0
Minor	6	6	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 Bido to identify any potential issues and vulnerabilities in the source code of the Bido 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 6 issues of varying severity, listed below.

ID	Title	Severity	Status
BID-1	Unused Libraries	Minor	Fixed
BID-2	Redundant Checks	Minor	Fixed
BID-3	`_unstake` Emit Wrong Event	Minor	Fixed
BID-4	Use `!= 0` instead of `> 0` for Unsigned Integer Comparison	Minor	Fixed
SBT-1	Uncalled Initialization Function	Minor	Fixed
WBT-1	Immutable Parameters	Minor	Fixed

3 Participant Process

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

Admin

- The Admin can initialize the initial shares only once through `initialize()`.
- The Admin can stop the `stake` and `unstake` function through `pauseStaking()`
- The Admin can resume the `stake` and `unstake` function through `resumeStaking()`.
- The Admin can stop the `stake` `unstake` and `_transferShares` function through `stop()`.
- The Admin can resume the `stake` `unstake` and `_transferShares` function through `resume()`.

User

- The User can stake their assets through `stake()`.
- The User can unstake their assets through `unstake()`.
- The User can swap StBTC to WstBTC through `wrap()`.
- The User can swap WstBTC to StBTC through `unwrap()`.

4 Findings

BID-1 Unused Libraries

Severity: Minor

Status: Fixed

Code Location:

Bido.sol#6

Descriptions:

The `Math256` library is imported into the contract, but it is not used.

Suggestion:

It is recommended to delete unused libraries.

Resolution:

The client has deleted unused libraries.

BID-2 Redundant Checks

Severity: Minor

Status: Fixed

Code Location:

Bido.sol#137

Descriptions:

Before the start of `staking`, you need to call the `initialize` function to initialize the contract. At this time, there is already a fund in the contract, recorded at the address `0xdead`, and no one can withdraw this fund. When calling the `stake` function, you also need to pass a fund, so the funds in the contract at this time are equal to the initial funds plus the funds passed in by the user. This value is always greater than the funds passed in by the user.

Suggestion:

It is recommended to delete redundant require checks.

Resolution:

The client has removed redundant checks.

BID-3 `_unstake` Emit Wrong Event

Severity: Minor

Status: Fixed

Code Location:

Bido.sol#175

Descriptions:

In the `_unstake` function, it will emit the `UnStaked` event with the parameters of the caller and the number of unstakes. However, the unstake amount here uses `msg.value`. In this context, `msg.value` is always 0. which will confuse the users and disturb the on-chain data.

Suggestion:

It is recommended to change the unstake amount in the `_unstake` event to the correct parameter.

Resolution:

The client has correctly changed the parameters.

BID-4 Use `!= 0` instead of `> 0` for Unsigned Integer Comparison

Severity: Minor

Status: Fixed

Code Location:

Bido.sol#157,166

Descriptions:

When dealing with unsigned integer types, comparisons with `!= 0` are cheaper than with `> 0`.

```
require(_amount > 0, "UNSTAKE_ZERO");
```

Suggestion:

It is recommended to use `!= 0` instead of `> 0` for unsigned integer comparison.

Resolution:

The client has optimized the code.

SBT-1 Uncalled Initialization Function

Severity: Minor

Status: Fixed

Code Location:

StBTCPermit.sol#193-200

Descriptions:

The `_initializeEIP712StBTC` function is defined in the `StBTCPermit` contract. This function is used to initialize the address of the `eip712StBTC` contract. However, this function is `internal` and there is no superior function to call it, which will cause `_initializeEIP712StBTC` to be unavailable.

Suggestion:

It is recommended to confirm whether this situation conflicts with the design concept.

Resolution:

The client has deleted the `StBTCPermit` contract.

WBT-1 Immutable Parameters

Severity: Minor

Status: Fixed

Code Location:

WstBTC.sol#27

Descriptions:

The `stBTC` parameter is defined in the contract. This parameter will only be initialized in the `constructor` and will not be changed subsequently.

Suggestion:

It is recommended to change this parameter to an `immutable` type.

Resolution:

The client has changed the variable to immutable.

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.