Bitsmiley Smart Contract

Audit Report

Mon May 06 2024

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

ScaleBit

Bitsmiley Smart Contract Audit Report

1 Executive Summary

1.1 Project Information

Description	BitSmiley is a protocol based on the Bitcoin blockchain under the Fintegra framework. It consists of three main components: a decentralized overcollateralized stablecoin protocol, a native trustless lending protocol, and a derivatives protocol.
Type	DeFi
Auditors	ScaleBit
Timeline	Thu Mar 14 2024 - Thu Mar 14 2024
Languages	Solidity
Platform	BTC
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/bitSmiley-protocol/evm-contracts
Commits	e734b38fbd171b7ad9204baade8d2091bc995380 85004bb8b68be935fe815b3ecf4dded992819d82 6bcd8e1cae553d5bf2af8d4ad76dd48e4b9702f8

1.2 Files in Scope

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

ID	File	SHA-1 Hash
ITO	contracts/IToken.sol	5873e78f3b811af948bc37ffd37bb2463d767f03
BUSD	contracts/BitUSD.sol	e2ccc92eaf47191b056123df2ea4c36d2c127164
SERC7EW	contracts/StakingERC721EndWithdraw.sol	093e80b85abdcda96dc10e71bd53bc86a55cb454
SFE	contracts/StabilityFee.sol	96b8859066eaa36acae48457316172cf9a8f0ac4
BSNF	contracts/BitSmileyNoFee.sol	56e658628f798f44b13a2043be91741e8699d2fe
BUSDL2	contracts/BitUSDL2.sol	250b1c512559988e8f098f1b9e1fb64c4d2a545a
ORA	contracts/Oracle.sol	8603ebe6bbf2eb988f9f8eca1cf47dfb1b8d97a8
TUT	contracts/TransferUtil.sol	e0242edf28b1b5752dfeca19696bd8683f1dc339
SUT	contracts/Oracle/SubmimissionUtil.sol	f27c6a23cf0051eb9796c7aa425cab2aaba4886b
STR	contracts/Oracle/Structs.sol	f05f789b75cf7bb226a062d413b422fcb29255ba
OFE	contracts/Oracle/OracleFeed.sol	6e99eadf7b69e79a99f851dd95bd59bc82d86564

IVM	contracts/IVaultManager.sol	85dcd7b99fbeb7b4705743c877d1cf4faf6e2c48
VOP	contracts/TestUtils/VaultOperator.sol	3911a9a3e11f3c3d2bc1a7ba63ee83247bfdbc8e
BUSDL2T	contracts/TestUtils/BitUSDL2Test.sol	c5aaf1b63ffe24e00a6d89cd386c5a5e52798e84
WBTC	contracts/TestUtils/WBTC.sol	2da424ba1e77e1a75f5cada19b0edf02ebae31b1
GJO	contracts/TestUtils/GemJoin.sol	bf0dba57161071a4a7f065473a49c91c7a819733
NFT	contracts/TestUtils/NFT.sol	9dc24c94b6577239b2ce2d661fe81cbd7140b5d8
VMT	contracts/TestUtils/VaultManagerTest.sol	2fe775ed2ea0eeb3338fa9b04451371da16d046d
VMA	contracts/VaultManager.sol	f6c8780e88042170db920a24308a99715dacb6fd
IOR	contracts/IOracle.sol	4ad9a39ad4e6027bf1e5631b12fda98608dcc78c
VAU	contracts/Vault.sol	3d9390fe113dcbfd97c136b458d9346d311ab2ef

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	6	0	6
Informational	4	0	4
Minor	0	0	0
Medium	1	0	1
Major	1	0	1
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 Bitsmiley to identify any potential issues and vulnerabilities in the source code of the Bitsmiley 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
BSF-1	Failed to Update Interest Modification Time	Medium	Acknowledged
BSN-1	Unused Constant	Informational	Acknowledged
BUS-1	Centralization Risk	Major	Acknowledged
SER-1	Withdraw Function Can Only Be Called Once	Informational	Acknowledged
SER-2	The Unreasonable Reward Calculation Mechanism	Informational	Acknowledged
VMA-1	Insufficient Parameter Validation	Informational	Acknowledged

3 Participant Process

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

Admin

- The `Admin` can set `liquidationFees` through `setLiquidationFee()`.
- The `Admin` can set liquidation beneficiary and set address of the contract StabilityFee through `setAddress()`.
- The `Admin` can set caller address that decides which contract it can be interact with through `setCaller()`.
- The `Admin` can set fee beneficiary through `setFeeBeneficiary()`.
- The `Admin` can set fee rate for collateral through `setFeeRate()`.
- The `Admin` can pause the contract through `pause()`.
- The `Admin` can unpause the contract through `unpause()`.

User

- The `User` can open a vault and mint amounts of bitUSD through `openVault()`.
- The `User` can borrow bitUSD based on his or her vault value through `mint()`.
- The `User` can repay the debt for owened vault through `repay()`.
- The `User` can repay all debt for owened vault through `repayAll()`.

Liquidator

- The `Liquidator` can liquidate any unsafe vault through `liquidate()`.

4 Findings

BSF-1 Failed to Update Interest Modification Time

Severity: Medium

Status: Acknowledged

Code Location:

contracts/StabilityFee/BaseStabilityFee.sol#150

Descriptions:

When updating the interest modification time information, the situation where the interest is 0 is not considered, which may cause the calculation of interest to exceed the actual borrowing situation. Consider a situation where bob pays off his debt in January and borrows again one month later. The calculated interest payable at this time is 0 because the debt scale is 0. This will skip the time update and wait for another month. After Bob pays off the debt, the interest calculated at this time is the time when the debt was first paid off, which results in an extra month of interest payment.

Suggestion:

It is recommended to make the time updated correctly.

Resolution:

The client replied that they are aware of this issue and will fix it in the future.

BSN-1 Unused Constant

Severity: Informational

Status: Acknowledged

Code Location:

contracts/BitSmileyNoFee.sol#16

Descriptions:

There are unused `error()` in the contract.

Suggestion:

It is recommended to remove unused `error()` if there's no further design.

BUS-1 Centralization Risk

Severity: Major

Status: Acknowledged

Code Location:

contracts/BitUSD.sol;

contracts/Oracle.sol

Descriptions:

Centralization risk was identified in the smart contract.

- The Admin can mint/burn any amount of tokens to/from any address.
- The Admin can set the oracle price.
- The Admin can pause/unpause the contract.

Suggestion:

It is recommended to take measures to mitigate this issue.

SER-1 Withdraw Function Can Only Be Called Once

Severity: Informational

Status: Acknowledged

Code Location:

contracts/StakingERC721EndWithdraw.sol#86

Descriptions:

We found that the `withdraw()` function can only be called once by an address, which may cause the user's pledge to be unable to be retrieved.

Suggestion:

It is recommended to make sure this matches your design to prevent any property damage.

SER-2 The Unreasonable Reward Calculation Mechanism

Severity: Informational

Status: Acknowledged

Code Location:

contracts/StakingERC721EndWithdraw.sol#165

Descriptions:

According to the contract code, we believe that the current reward mechanism may also need to pay attention to the issuance restrictions of NFT addresses and the value of NFT.

Suggestion:

It is recommended to pay attention to the quantity limit of NFT issuance to prevent malicious tools from minting a large number of NFT and taking away the rewards. Consider the value of NFT as one of the factors that determine the number of rewards.

VMA-1 Insufficient Parameter Validation

Severity: Informational

Status: Acknowledged

Code Location:

contracts/VaultManager.sol#143

Descriptions:

The variable `caller` was checked, but the parameter `_caller` was not checked for being 0.

Suggestion:

It is recommended to add a zero address check for the parameter `_caller`.

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.