

Smart contract security audit

Firebirds

v.1.0



No part of this publication, in whole or in part, may be reproduced, copied, transferred or any other right reserved to its copyright a CTDSec, including photocopying and all other copying, any transfer or transmission using any network or other means of communication, in any form or by any means such as any information storage, transmission or retrieval system, without prior written permission.

Table of Contents

1.0 Introduction	3
1.1 Project engagement	3
1.2 Disclaimer	3
2.0 Coverage	3
2.1 Target Code and Revision	3
2.2 Attacks made to the contract	4
3.0 Security Issues	6
3.1 High severity issues [0]	6
3.2 Medium severity issues [1]	6
3.3 Low severity issues [2]	6
4.0 Summary of the audit	7

1.0 Introduction

1.1 Project engagement

During April of 2022, Firebirds engaged CTDSec to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. Firebirds provided CTDSec with access to their code repository and whitepaper.

1.2 Disclaimer

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the network's fast-paced and rapidly changing environment, we at CTDSec recommend that Firebirds team put in place a bug bounty program to encourage further and active analysis of the smart contract.

2.0 Coverage

2.1 Target Code and Revision

For this audit, we performed research, investigation, and review of the Firebirds contract followed by issue reporting, along with mitigation and remediation instructions outlined in this report. The following code files are considered in-scope for the review:

Source:

Firebirds.sol [SHA256] - f73241be6c42368792f7a14cbd79203b627dc0bc649792286b703cd5bab9622d

2.2 Attacks made to the contract

In order to check for the security of the contract, we tested several attacks in order to make sure that the contract is secure and follows best practices.

No	Issue description.	Checking status
1	Compiler warnings.	PASSED
2	Race conditions and Reentrancy. Cross-function race conditions.	PASSED
3	Possible delays in data delivery.	PASSED
4	Oracle calls.	PASSED
5	Front running.	PASSED
6	Timestamp dependence.	PASSED
7	Integer Overflow and Underflow.	PASSED
8	DoS with Revert.	PASSED
9	DoS with block gas limit.	PASSED
10	Methods execution permissions.	PASSED
11	Economy model. If application logic is based on an incorrect economic model, the application would not function correctly and participants would incur financial losses. This type of issue is most often found in bonus rewards systems, Staking and Farming contracts, Vault and Vesting contracts, etc.	PASSED
12	The impact of the exchange rate on the logic.	PASSED
13	Private user data leaks.	PASSED
14	Malicious Event log.	PASSED
15	Scoping and Declarations.	PASSED
16	Uninitialized storage pointers.	PASSED

17	Arithmetic accuracy.	PASSED
18	Design Logic.	MEDIUM ISSUES
19	Cross-function race conditions.	PASSED
20	Safe Zeppelin module.	PASSED
21	Fallback function security.	PASSED
22	Overpowered functions / Owner privileges	PASSED

3.0 Security Issues

3.1 High severity issues [0]

No high severity issues found.

3.2 Medium severity issues [1]

1. Rounding errors

The variable roundedAmount is calculated by doing a division of the amount and total supply and then multiplication to total supply, doing the multiplication first rounding errors will be reduced.

```
300      uint256 roundedAmount = (_amount / totalSupply()) * totalSupply();
```

Recommendation:

Do multiplication before division.

3.3 Low severity issues [2]

1. Hardcoded string

The project contains a hardcoded url:

```
string public constant BASE_URI = "https://firebirdsnft.mypinata.cloud/ipfs/QmSDXbB8o7Jtiz8SHDpNTqNJThNN4rYzgs2WJpU5gDQJ1E/";
```

Recommendation:

We recommend initializing the string at the constructor instead of hardcoding it.

2. To do comments

Contract have 'To do' comments inside the code:

```
94  ∨ // TODO i suggest this to exclude contracts from mint, remove the line below to allow
```

Recommendation:

Confirm with the development team if the code behavior is the one that is wanted.

4.0 Summary of the audit

Contract is safe to be deployed, we recommend that the team make use of a multisignature wallet for the owner role and to review medium/low issues.