# Smart contract security audit

# 3air

v.1.0

# Table of Contents

# 1.0 Introduction

## 1.1 Project engagement

During April of 2022, 3air 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. 3air 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 3air 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 3air 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:

3air.sol - 8aa19ea90a3e828c70914d5eb539f20ec18bd37f72359996e6c8de267aead6bf

Vesting.sol - 22c1c6d8c326ca618c6347586b14b42234dc9493458b4eea22880427bda1fd19

## 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.

| № | 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. | MEDIUM ISSUES (only owner, no external access) |
| 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 | MEDIUM ISSUES |

# 3.0 Security Issues

## 3.1 High severity issues [1]

**1. Wrong token economics parametrization**

The vesting comment of Airdrops_1 should be 2,5% per month as described in the comment but instead is applied '0'.

```
            //2,5% per month starting 3 months after TGE

            return (91 days, 1217 days, 0);

        } else if (vestingType == VestingType.AIRDROPS_1) {
```

Recommendation:

Check if the logic is correctly applied.

## 3.2 Medium severity issues [4]

**1. Centralized function can lead to a disruption of trade**:

The disableSecurityProxy (file a 3air.sol) function if it's activated will disable all the transfers triggering an exception ('Transfer not allowed').

```
37      function _beforeTokenTransfer(address from, address to, uint256 amount)
38      internal
39      override(ERC20)
40      {
41
42          if(securityProxyAddress != address(0)) {
43              require(ISecurityProxy(securityProxyAddress).validateTransfer(from, to, amount), "Transfer not allowed");
44          }
45
46          super._beforeTokenTransfer(from, to, amount);
47      }
```

**Recommendation:**

Add a max time limit for the function, for example, if will be used as antisniper tool limit it to x blocks.

**2. High cost of gas in the function getVestingDataByType**

The function 'getVestingDataByType' that is called every time that 'addVestingTerm' is executed (Adding a vesting for an address) will use high cost of gas.

Recommendation:

We recommend having an array where you add the address/vesting details and amount in order to reduce gas cost.

**3. 0 Address is not checked for owner**

During the constructor owner address is not checked to not be equal to 0 address.

**Recommendation:**

We recommend to check the owner address inside the constructor with a similar procedure to the next one:

```
require(ownerAddress != address(0), "Owner can't be address 0");
```

**4. Centralization:**

3air.sol

setSecurityProxy(): Owner can change the security proxy address and enable it.

disableSecurityProxy(): Owner can disable the security proxy.

Vesting.sol

addVestingTerm(): Owner can add a new vesting type.

addVestingTerms(): Owner can add multiple vesting types.

Recommendation:

-Add a time lock on privileged operations.

-Use a multisig wallet to prevent SPOF on the Private Key.

-Introduce DAO mechanism for owner functions (will add transparency and user involvement).

*Notes: All team wallets are used with multisig wallets.*

## 3.3 Low severity issues [1]

**1. Function claim tokens don't verify at start balances/tokens**

At the function for claiming tokens there is no requirement to check if the user has additional releasable tokens in the vesting process.

```
function claimTokens(uint256 termId) public {



        require(termId < totalVestingTerms[msg.sender], "Term does not
exist");
```

Recommendation:

We recommend adding requirements at the start to verify if the user has enough privileges to do the claiming process or it's finished.

# 4.0 Summary of the audit

The contract follows code practices that can imply a high cost, there are also validation structures that could be validated through requires/arrays.

Currently there are stable platforms that allow vesting to be carried out, such as unicrypt, which is an alternative for this.

We recommend reviewing the points mentioned in the report and applying the necessary corrections.