Assure DeFi®

THE VERIFICATION **GOLD STANDARD**

Security Assessment

ICB Network

Date: 18/04/2024

Audit Status: PASS

Audit Edition: Advanced

ASSURE DEFI®

THE VERIFICATION **GOLD STANDARD**

Risk Analysis

Vulnerability summary

Classification	Description
🔴 High	High-level vulnerabilities can result in the loss of assets or manipulation of data.
🟠 Medium	Medium-level vulnerabilities can be challenging to exploit, but they still have a considerable impact on smart contract execution, such as allowing public access to critical functions.
🟡 Low	Low-level vulnerabilities are primarily associated with outdated or unused code snippets that generally do not significantly impact execution, sometimes they can be ignored.
🟢 Informational	Informational vulnerabilities, code style violations, and informational statements do not affect smart contract execution and can typically be disregarded.

Executive Summary

According to the Assure assessment, the Customer's smart contract is **Secured.**

Insecure	Poorly Secured	Secured	Well Secured

Scope

Target Code And Revision

For this audit, we performed research, investigation, and review of the ICB Network contracts followed by issue reporting, along with mitigation and remediation instructions outlined in this report.

Target Code And Revision

Project	Assure
Language	Solidity
Codebase	ICBVesting.sol - [ShA256] *30ded4e9e4cff24a01dcd16d4782d83d825c23bfc2a85228cf6f23bcc9635a2b*
Audit Methodology	Static, Manual

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.

Category	Item
Code review & Functional Review	● Compiler warnings. ● Race conditions and Reentrancy. Cross-function race conditions. ● Possible delays in data delivery. ● Oracle calls. ● Front running. ● Timestamp dependence. ● Integer Overflow and Underflow. ● DoS with Revert. ● DoS with block gas limit. ● Methods execution permissions. ● Economy model. ● Private user data leaks. ● Malicious Event log. ● Scoping and Declarations. ● Uninitialized storage pointers. ● Arithmetic accuracy. ● Design Logic. ● Cross-function race conditions. ● Safe Zeppelin module. ● Fallback function security. ● Overpowered functions / Owner privileges

AUDIT OVERVIEW

◆◆◆ HIGH

No high severity issues were found.

◆◆◆ MEDIUM

No medium severity issues were found.

◆◆◆ LOW

1. Preventing Gas Limit Exceedance in lockICB()

Contract: ICBVesting

Function: lockICB()

Issue: The lockICB() function risks running out of gas if the array size for _users or _amounts is not capped, potentially leading to incomplete transactions.

Mitigation: Implement a require() statement to enforce a maximum length for the _users and _amounts arrays, preventing the function from exceeding gas limits.

2. Ensuring Valid Addresses in lockICB()

Contract: ICBVesting

Function: lockICB()

Issue: There's a risk of bad contract behavior if zero addresses are included in the _users array within the lockICB() function.

Mitigation: Introduce a require() check within the function to ensure that no zero addresses are allowed in the _users array, thereby maintaining the integrity of the contract operations.

INFORMATIONAL

1. Correcting Time Unit Misinterpretation in `lockICB()`

Contract: ICBVesting

Function: lockICB()

Issue: Incorrect application of time units where `lockDurations` and `vestingPeriods` could be erroneously calculated if multiplied by 1 minute instead of 1 day, leading to wrong lock timestamps.

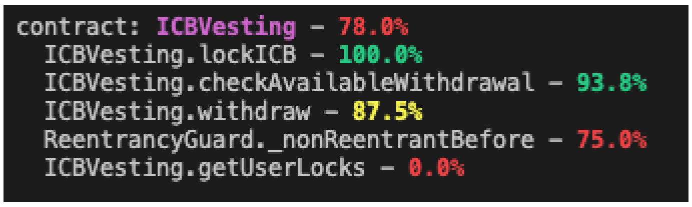
Mitigation: Modify the function to multiply `lockDurations` and `vestingPeriods` by 1 day instead of 1 minute to accurately represent the intended periods.

Testing coverage

During the testing phase, custom use cases were written to cover all the logic of contracts. **Check “Annexes” to see the testing code.*

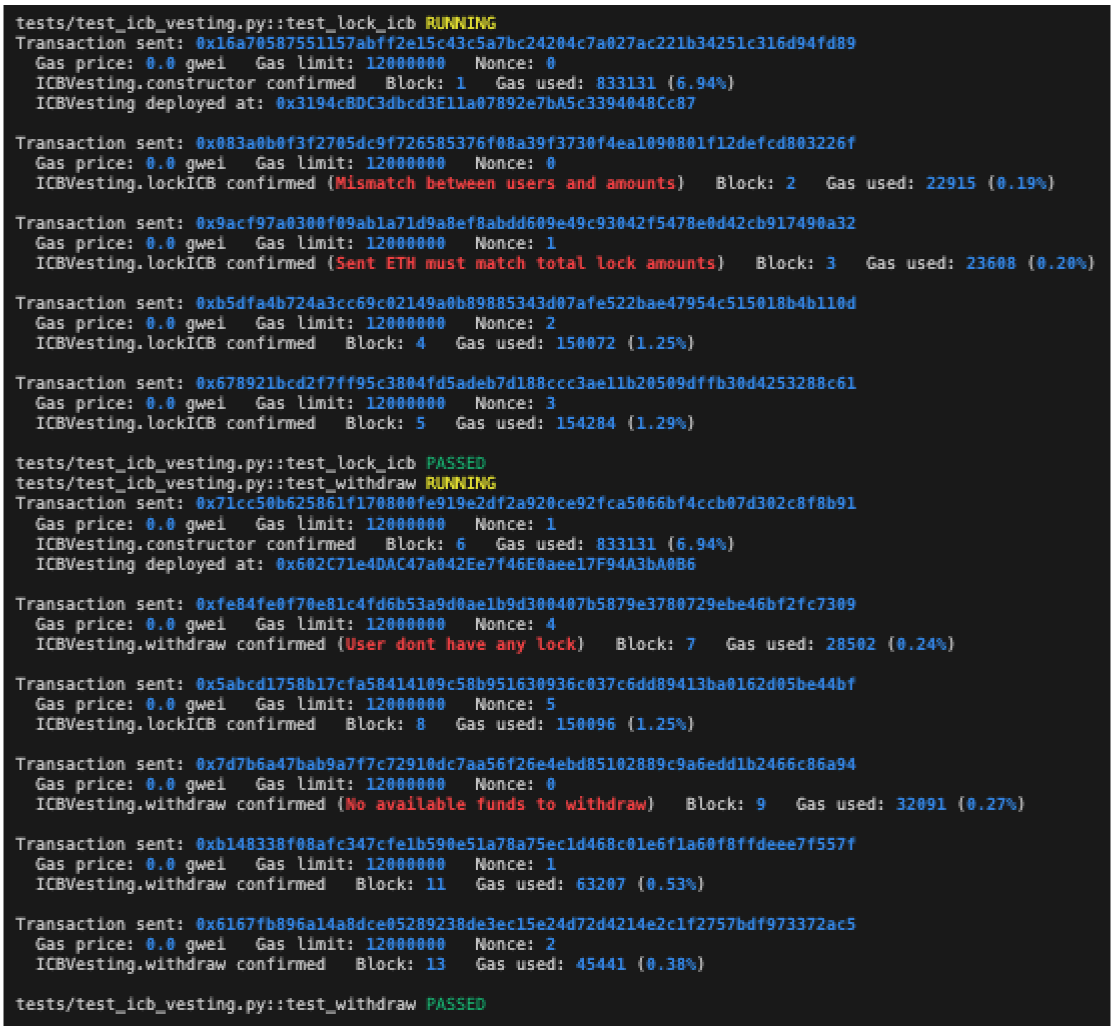
ICB Network vesting test:

Coverages:



```
contract: ICBVesting - 78.0%
  ICBVesting.lockICB - 100.0%
  ICBVesting.checkAvailableWithdrawal - 93.8%
  ICBVesting.withdraw - 87.5%
  ReentrancyGuard._nonReentrantBefore - 75.0%
  ICBVesting.getUserLocks - 0.0%
```

Testing ICB Network Token:



```
tests/test_icb_vesting.py::test_lock_icb RUNNING
Transaction sent: 0x16a70587551157abff2e15c43c5a7bc24204c7a027ac221b34251c316d94fd89
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  ICBVesting.constructor confirmed   Block: 1   Gas used: 833131 (6.94%)
  ICBVesting deployed at: 0x3194cBDC3dbcd3E11a07892e7bA5c3394048Cc87

Transaction sent: 0x083a0b0f3f2705dc9f726585376f08a39f3730f4ea1090801f12defcd803226f
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  ICBVesting.lockICB confirmed (Mismatch between users and amounts)   Block: 2   Gas used: 22915 (0.19%)

Transaction sent: 0x9acf97a0300f09ab1a71d9a8ef8abdd609e49c93042f5478e0d42cb917490a32
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 1
  ICBVesting.lockICB confirmed (Sent ETH must match total lock amounts)   Block: 3   Gas used: 23608 (0.20%)

Transaction sent: 0xb5dfa4b724a3cc69c02149a0b89885343d07afe522bae47954c515018b4b110d
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 2
  ICBVesting.lockICB confirmed   Block: 4   Gas used: 150072 (1.25%)

Transaction sent: 0x678921bcd2f7ff95c3804fd5adeb7d188ccc3ae11b20509dffb30d4253288c61
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 3
  ICBVesting.lockICB confirmed   Block: 5   Gas used: 154284 (1.29%)

tests/test_icb_vesting.py::test_lock_icb PASSED
tests/test_icb_vesting.py::test_withdraw RUNNING
Transaction sent: 0x71cc50b625861f170800fe919e2df2a920ce92fca5066bf4ccb07d302c8f8b91
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 1
  ICBVesting.constructor confirmed   Block: 6   Gas used: 833131 (6.94%)
  ICBVesting deployed at: 0x602C71e4DAC47a042Ee7f46E0aee17F94A3bA0B6

Transaction sent: 0xfe84fe0f70e81c4fd6b53a9d0ae1b9d300407b5879e3780729ebe46bf2fc7309
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 4
  ICBVesting.withdraw confirmed (User dont have any lock)   Block: 7   Gas used: 28502 (0.24%)

Transaction sent: 0x5abcd1758b17cfa58414109c58b951630936c037c6dd89413ba0162d05be44bf
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 5
  ICBVesting.lockICB confirmed   Block: 8   Gas used: 150096 (1.25%)

Transaction sent: 0x7d7b6a47bab9a7f7c72910dc7aa56f26e4ebd85102889c9a6edd1b2466c86a94
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  ICBVesting.withdraw confirmed (No available funds to withdraw)   Block: 9   Gas used: 32091 (0.27%)

Transaction sent: 0xb148338f08afc347cfe1b590e51a78a75ec1d468c01e6f1a60f8ffdeee7f557f
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 1
  ICBVesting.withdraw confirmed   Block: 11   Gas used: 63207 (0.53%)

Transaction sent: 0x6167fb896a14a8dce05289238de3ec15e24d72d4214e2c1f2757bdf973372ac5
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 2
  ICBVesting.withdraw confirmed   Block: 13   Gas used: 45441 (0.38%)

tests/test_icb_vesting.py::test_withdraw PASSED
```

Annexes

Testing code:

`icb_vesting.py:`

```python
from brownie import (
    reverts
)

from brownie.network.contract import Contract

from scripts.helpful_scripts import (
    ZERO_ADDRESS,
    DAY_TIMESTAMP,
    get_account,
    increase_timestamp
)

from scripts.deploy import (
    deploy_vesting
)

def test_lock_icb(only_local):
    # Arrange
    owner = get_account(0)
    other = get_account(1)
    extra = get_account(2)
    # Deploy contracts
    vesting = deploy_vesting(owner)


    with reverts("Mismatch between users and amounts"):
        vesting.lockICB(
            [],
            [1],
            1, 5, {"from": other})
    with reverts("Sent ETH must match total lock amounts"):
        vesting.lockICB(
            [extra],
            [1],
            1, 5, {"from": other})
    vesting.lockICB(
            [extra],
            [1],
            1440, 0, {"from": other, "value": 1e18})
    assert vesting.locks(0)[0] == extra
    assert vesting.locks(0)[1] == 1e18


    vesting.lockICB(
            [other],
            [2],
            1440, 0, {"from": other, "value": 2e18})
    assert vesting.locks(1)[0] == other
    assert vesting.locks(1)[1] == 2e18
```

```python
def test_withdraw(only_local):
    # Arrange
    owner = get_account(0)
    other = get_account(1)
    extra = get_account(2)

    # Deploy contracts
    vesting = deploy_vesting(owner)

    with reverts("User dont have any lock"):
        vesting.withdraw(0, {"from": other})
    vesting.lockICB(
            [extra],
            [1],
            1440, 1440, {"from": other, "value": 1e18})
    with reverts("No available funds to withdraw"):
        vesting.withdraw(0, {"from": extra})

    with reverts("No record for this user found"):
        vesting.getUserLocks(other, {"from": extra})

    increase_timestamp(int(DAY_TIMESTAMP * 1.5))
    vesting.withdraw(0, {"from": extra})
    assert vesting.locks(0)[0] == extra
    assert vesting.locks(0)[4] < 0.55e18

    increase_timestamp(int(DAY_TIMESTAMP * 1.5))
    vesting.withdraw(0, {"from": extra})
    assert vesting.locks(0)[0] == extra

    assert vesting.locks(0)[4] == 1e18
```

Technical Findings Summary

Findings

Vulnerability Level	Total	Pending	Not Apply	Acknowledged	Partially Fixed	Fixed
🔴 High	0					
🟠 Medium	0					
🟡 Low	2					
🟢 Informational	1					

Assessment Results

Score Results

Review	Score
Global Score	**90/100**
Assure KYC	Pending
Audit Score	85/100

The Following Score System Has been Added to this page to help understand the value of the audit, the maximum score is 100, however to attain that value the project must pass and provide all the data needed for the assessment. Our Passing Score has been changed to 84 Points for a higher standard, if a project does not attain 85% is an automatic failure. Read our notes and final assessment below. The Global Score is a combination of the evaluations obtained between having or not having KYC and the type of contract audited together with its manual audit.

Audit PASS

Following our comprehensive security audit of the token contract for ICB Network project, the audit has been successfully completed and passed. However, we recommend reviewing and addressing the low and informative vulnerabilities reported to ensure the robustness and security of the smart contract.

Disclaimer

Assure Defi has conducted an independent security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocating for the Project, and users relying on this report should not consider this as having any merit for financial advice in any shape, form, or nature. The contracts audited do not account for any economic developments that the Project in question may pursue, and the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude, and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are entirely free of exploits, bugs, vulnerabilities or deprecation of technologies.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any person reading this report should invest their funds without sufficient individual due diligence, regardless of the findings presented. Information is provided 'as is, and Assure Defi is under no covenant to audit completeness, accuracy, or solidity of the contracts. In no event will Assure Defi or its partners, employees, agents, or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions or actions with regards to the information provided in this audit report.

The assessment services provided by Assure Defi are subject to dependencies and are under continuing development. You agree that your access or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The services may access, and depend upon, multiple layers of third parties.