Assure DeFi®

THE VERIFICATION **GOLD STANDARD**

Security Assessment

Gradient

Date: 10/06/2025

Audit Status: PASS

Audit Edition: Advanced+

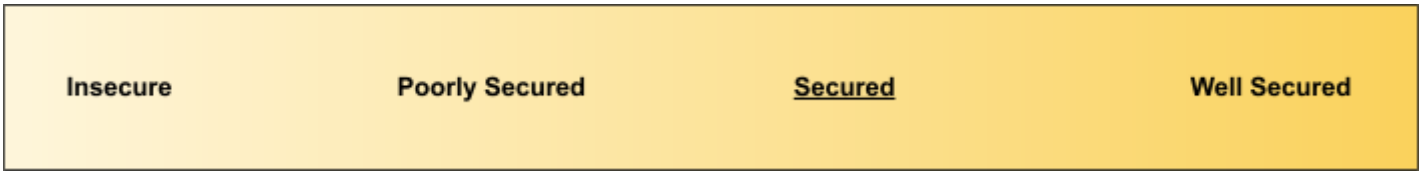
Risk Analysis

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Low	Low-level vulnerabilities are primarily associated with outdated or unused code snippets that generally do not significantly impact execution, sometimes they can be ignored.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.**



Scope

Target Code And Revision

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

Target Code And Revision

Project	Assure
Language	Solidity
Codebase	https://github.com/GradientDevelopment/Gradient/blob/main/contracts/Orderbook.sol V2 - Orderbook.sol [sha256]: 56031a645dae478a0eee61261c74f65357347bb61729dfa2f74510073822934a V3 - Orderbook.sol (1) [sha256] - Final version: 13ce7ac0305d8696f1413a627262b264b13db5d31162f120eebbc691534369fc
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

1. Expired Buy Orders Lock Funds [Fixed ✅]

Function: cleanupExpiredOrder

Issue: Expired buy orders never refund locked ETH, permanently locking users funds.

Recommendation: In cleanupExpiredOrder, detect buy orders and refund remainingAmount * price / 1e18 + feeRefund; decrement totalFeesCollected accordingly.

Fix: cleanupExpiredOrder now handles buy orders, refunds principal+fee, and decrements totalFeesCollected.

2. Fee Accounting Discrepancy on Cancellation [Fixed ✅]

Function: cancelOrder

Issue: totalFeesCollected is not decremented when refunding user fees, enabling owner to over-withdraw

Recommendation: Subtract feeRefund from totalFeesCollected when issuing user refunds.

Fix: cancelOrder caps and subtracts feeRefund from totalFeesCollected before refunding, preventing double-dipping.

3. cleanupExpiredOrder Locks Principal When Fees Are Drained [Fixed ✅]

Function: cleanupExpiredOrder

Issue: Broken buy-order refunds when fees have been withdrawn: you compute refundAmount = principal + buyerFee but only decrement totalFeesCollected if you have fees. If the owner has already drained all fees, totalFeesCollected is zero and you still require address(this).balance >= refundAmount but the contract only holds the principal ETH for the buy, so the require reverts and locks the user's principal forever.

Recommendation: Mirror your cancelOrder logic: compute actualFeeRefund = min(buyerFee, totalFeesCollected), decrement totalFeesCollected by that, and reduce refundAmount by buyerFee - actualFeeRefund so you always refund at most what you hold.

Fix: Correct fee-cap logic now ensures only principal is ever stuck, even if fees have been withdrawn.

MEDIUM

1. Incorrect Fee Logic on Price-Savings Refund [Fixed ✅]

Function: _fulfillMatchedOrders

Issue: Double-charging/refunding of fees on price-savings leads to unintended payouts

Recommendation: Rework “savings” refund logic: refund only the net difference without reapplying fee or remove savedFee.

Fix: In _fulfillMatchedOrders, you now refund only the raw savedAmount, the extra savedFee logic has been removed.

2. Unbounded OrderQueue Growth (Gas DoS) [Acknowledge ✅]

Function: getActiveOrders / orderQueues (global)

Issue: Unbounded queue growth, stale IDs never pruned → gas-DoS on lookups

Recommendation: On fill/cancel/expire, remove or mark-compact order IDs from orderQueues[queueKey].

3. Unsafe ERC-20 Transfers [Fixed ✅]

Function: createOrder, cancelOrder, cleanupExpiredOrder

Issue: Unsafe ERC-20 handling: direct transfer/transferFrom may break on non-standard tokens

Recommendation: Use OpenZeppelin SafeERC20 wrapper to handle missing bool returns and reverts uniformly.

Fix: All token transfers now use SafeERC20 safeTransfer/safeTransferFrom.

1. Low-Level .call Without Gas/Stipend Safeguards [Fixed ✅]

Function: withdrawFees, refunds

Issue: Low-level .call{value:.} with no gas stipend control; albeit state-first, absence of nonReentrant on withdrawFees

Recommendation: Wrap all external ETH transfers with OpenZeppelin’s Address.sendValue and add nonReentrant to withdrawFees.

Fix: Using onlyOwner.

2. Ownable Constructor Misuse [Acknowledge ✅]

Function: withdrawFees, refunds

Issue: Incorrect invocation Ownable(msg.sender) may fail compilation or misconfigured ownership

Recommendation: Remove parameter; call Ownable() default constructor so that OZ own msg.sender is correctly set.

INFORMATIONAL

---

1. Timestamp Dependency [Acknowledge ✅]

Function: isOrderExpired

Issue: Uses block.timestamp, vulnerable to miner timestamp manipulation (15s)

Recommendation: Document this limitation or consider block-number-based expiry for very time-sensitive use-cases.

2. Decimal limit [Acknowledge ✅]

Function: createOrder()

Issue: If the token price is expected to have 18 decimals, smaller amounts may not function correctly.

Recommendation: Remove the division by 18 to ensure the price per token can accommodate smaller ETH amounts.

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.*

```
tests/test_orderbook.py::test_create_order RUNNING
Transaction sent: 0xe1ff63ff59524a7abed77049f2267cd7b5068698d62e4d65c6253184363940a0
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  ERC20Mock.constructor confirmed   Block: 1   Gas used: 523822 (4.37%)
  ERC20Mock deployed at: 0x3194cBDC3dbcd3E11a07892e7bA5c3394048Cc87

Transaction sent: 0xd50dd7eb989b4b9c0ed63a0c62b52530ea223e7248ec9c81887ff7b0c084d662
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 1
  GradientOrderbook.constructor confirmed   Block: 2   Gas used: 2201820 (18.35%)
  GradientOrderbook deployed at: 0x602C71e4DAC47a042Ee7f46E0aee17F94A3bA0B6

Transaction sent: 0x632fd8cdbc2ca674a0b17ecc5a782f496f5ae0d9f1f2ebbee7aaca6a716fb902
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 2
  ERC20Mock.mint confirmed   Block: 3   Gas used: 65821 (0.55%)

Transaction sent: 0x4825644c1325c53aeb237dd63ec1e3d211ba051a8556923632af155cce96736d
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  ERC20Mock.approve confirmed   Block: 4   Gas used: 44283 (0.37%)

Transaction sent: 0xc53440ed8c693f42fb7d7eef69432b6b3bafdbe828682afd72321e8375954a23
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 3
  ERC20Mock.mint confirmed   Block: 5   Gas used: 50809 (0.42%)

Transaction sent: 0xe9ea835997dc3c06e70b09f6c8813d54b7e76fa8d48946e4add57caa54d93be7
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  ERC20Mock.approve confirmed   Block: 6   Gas used: 44283 (0.37%)

Transaction sent: 0xed960cafe4980a64ad52ff5b7c760d31496aa7d45baac9924a86225dc15ac69b
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 1
  GradientOrderbook.createOrder confirmed (Invalid token)   Block: 7   Gas used: 28345 (0.24%)

Transaction sent: 0xbd71f061f10053898d4b96029e24dd698e7ba418549099bc1c96c4749d0aec59
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 2
  GradientOrderbook.createOrder confirmed (Amount must be greater than 0)   Block: 8   Gas used: 28590 (0.24%)

Transaction sent: 0x246f8a683ae16182e882797b1dfd2e985461593dd898d2bc0f61cb7b34926653
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 3
  GradientOrderbook.createOrder confirmed (Invalid price range)   Block: 9   Gas used: 28571 (0.24%)

Transaction sent: 0x2d250ddbe7612ca7a484720a020319741c10b397e7d9b5820d7f19961c0b7ed9
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 4
  GradientOrderbook.createOrder confirmed (TTL must be greater than 0)   Block: 10   Gas used: 28612 (0.24%)

Transaction sent: 0xcb02cb90d001bbfa8f753fa25fcc1d2f5b2180eb3cfb81f7cc228b4601f61e3e
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 5
  GradientOrderbook.createOrder confirmed (Insufficient ETH sent)   Block: 11   Gas used: 29866 (0.25%)

Transaction sent: 0xfa4414734a55fba51b5715a4943b856baf9273f9a29490f1919d9873e0cb4d03
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 6
  GradientOrderbook.createOrder confirmed   Block: 12   Gas used: 207761 (1.73%)

Transaction sent: 0x9859c43e990d8b2c9aa56b23d783361f1d0692767e29a02df9def57c06157cf1
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 1
  GradientOrderbook.createOrder confirmed   Block: 13   Gas used: 239440 (2.00%)

tests/test_orderbook.py::test_create_order PASSED
```

```
tests/test_orderbook.py::test_cancel_order RUNNING
Transaction sent: 0x65d4d295d008530607748c25c3f2a4873c11c995b32f6b0bb3edee3876502ca4
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 4
  ERC20Mock.constructor confirmed   Block: 14   Gas used: 523822 (4.37%)
  ERC20Mock deployed at: 0xe0aA552A10d7EC8760Fc6c246D391E698a82dDf9

Transaction sent: 0xb96e65d9462e1406b42c7d46cc265d026d97948344ac794ef2460bcd115b40f8
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 5
  GradientOrderbook.constructor confirmed   Block: 15   Gas used: 2201820 (18.35%)
  GradientOrderbook deployed at: 0x6b4BDe1086912A6Cb24ce3dB43b3466e6c72AFd3

Transaction sent: 0xb7ed23510d5212a16769de8eea4fa4c863b67525b3bd5b50e330a275b595de44
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 6
  ERC20Mock.mint confirmed   Block: 16   Gas used: 65821 (0.55%)

Transaction sent: 0xb75ab4c8b403a86ccdfb990fc9cd9d1d87c816a6240a8b0473c503ad3212c9c0
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 7
  ERC20Mock.approve confirmed   Block: 17   Gas used: 44283 (0.37%)

Transaction sent: 0x2963ad1baa640822cab9b3aae96ba2dcc45de974c0fc0f1263ad16c2dbdae0b7
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 7
  ERC20Mock.mint confirmed   Block: 18   Gas used: 50809 (0.42%)

Transaction sent: 0xdd4c67522d37c4d803148ea97396e073dd4696070c12be0f0cf6241f97a436bc
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 2
  ERC20Mock.approve confirmed   Block: 19   Gas used: 44283 (0.37%)

Transaction sent: 0x1e071c209ef311e8381c57b21fca21fd23c75245f366a3d7e9028a0f71a52074
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 8
  GradientOrderbook.createOrder confirmed   Block: 20   Gas used: 207761 (1.73%)

Transaction sent: 0x2db2c48a64c2b3811fafe7c9746eb5de6d06c58119ecbfbd17c986229ab9fd23
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 3
  GradientOrderbook.createOrder confirmed   Block: 21   Gas used: 239440 (2.00%)

Transaction sent: 0x976600c45f9e7e18b3f178dd51be1bc627ca493752674b088171f8d3f9d304ea
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 9
  GradientOrderbook.cancelOrder confirmed (Order does not exist)   Block: 22   Gas used: 28348 (0.24%)

Transaction sent: 0x44abbd921151c24fa1042ae630ec1b39bb71c9068d84783265f7e06bf9bf57e6
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 8
  GradientOrderbook.cancelOrder confirmed (Not order owner)   Block: 23   Gas used: 29233 (0.24%)

Transaction sent: 0xd8d62b3992133a2538c27f4c13001a19e6c1c7b66d2343f5797ccdf7c1efd7a3
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 10
  GradientOrderbook.cancelOrder confirmed   Block: 24   Gas used: 63420 (0.53%)

Transaction sent: 0xe0e17b1ed348d3c628d44abb5011023013ede638a5436c4bdd22a391beafee3c
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 11
  GradientOrderbook.cancelOrder confirmed (Order not active)   Block: 25   Gas used: 30163 (0.25%)

Transaction sent: 0x3218cb443ee4aa9662f988a399ce4db9da2433bd271415411fdb25e988f35ed0
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 4
  GradientOrderbook.cancelOrder confirmed   Block: 26   Gas used: 55308 (0.46%)

Transaction sent: 0x3ea08da148ac9b0c41365ce5f6143f759e9658f482994308df46f46ce61c8eee
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 9
  GradientOrderbook.cleanupExpiredOrder confirmed (Order not active)   Block: 27   Gas used: 29323 (0.24%)

Transaction sent: 0xf703c191bfc141fc95f24fcc3d9b7fdefc26035850d67d38ccc28a9a4885fc9f
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 12
  GradientOrderbook.createOrder confirmed   Block: 28   Gas used: 201161 (1.68%)

Transaction sent: 0xc3d7f8c1ec0b74835c19328f729601103a2cf10668e2d50f13ae56b669c8fa06
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 13
  GradientOrderbook.cancelOrder confirmed (Order expired)   Block: 30   Gas used: 32019 (0.27%)

Transaction sent: 0xfafcde961977e00827948ee950726953fb6d43dff88f5329bb83fa146cb87b42
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 5
  GradientOrderbook.createOrder confirmed   Block: 31   Gas used: 224440 (1.87%)

Transaction sent: 0x9f8b4bef23b133a30418f76c6d82ddeacf57f01c562e85f87e6583cd1541ba60
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 10
  GradientOrderbook.cleanupExpiredOrder confirmed   Block: 33   Gas used: 54438 (0.45%)
```

```
tests/test_orderbook.py::test_fulfill_matched_orders RUNNING
Transaction sent: 0xcaae7574140b242b748996b62499a1322c5348c8b434123edc30c7d842c63ab1
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 11
  ERC20Mock.constructor confirmed   Block: 34   Gas used: 523822 (4.37%)
  ERC20Mock deployed at: 0x7a3d735ee6873f17Dbdcab1d51B604928dc10d92

Transaction sent: 0x2afa7230790de6df64e30b7771ad026bc74656909a717871eb6a3c220e8acfc7
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 12
  GradientOrderbook.constructor confirmed   Block: 35   Gas used: 2201820 (18.35%)
  GradientOrderbook deployed at: 0x2c15A315610Bfa5248E4CbCbd693320e9D8E03Cc

Transaction sent: 0x015dc7f59c272cbdca9cddc3ab5dc525512cc70bf720c4df92c04676d8cb2b95
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 13
  ERC20Mock.mint confirmed   Block: 36   Gas used: 65821 (0.55%)

Transaction sent: 0x53b2b2499afaad26a491db8bafa25e9007f59680e0d3c9bd8b5274a456998613
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 14
  ERC20Mock.approve confirmed   Block: 37   Gas used: 44283 (0.37%)

Transaction sent: 0x988950f8eadab535d27a63ffd538aa4bf9423eb5abff191c01cbea1b0f034489
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 14
  ERC20Mock.mint confirmed   Block: 38   Gas used: 50809 (0.42%)

Transaction sent: 0xcc21710de993940a60054675e2f4c49fe328c779dfe91e946011964f094504b2
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 6
  ERC20Mock.approve confirmed   Block: 39   Gas used: 44283 (0.37%)

Transaction sent: 0x98c54e120307b1772c2dcbde2d5530dc9479bcf3625cb79058d1dd3c2a86a5e8
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 15
  GradientOrderbook.fulfillMatchedOrders confirmed (Caller is not whitelisted)   Block: 40   Gas used: 28664 (0.24%)

Transaction sent: 0x018c4e19737653bc65803fd067b6bba7ff48017abb9747dbe2c5578aaea9edf7
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 15
  GradientOrderbook.fulfillMatchedOrders confirmed (No order matches to fulfill)   Block: 41   Gas used: 28681 (0.24%)

Transaction sent: 0xd471c6ab4779717492f25581c101d684aa87d4e17d878c73e23e33b45165bd02
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 16
  GradientOrderbook.fulfillMatchedOrders confirmed (Order does not exist)   Block: 42   Gas used: 32279 (0.27%)

Transaction sent: 0x6f5ffb149e7dfbe7feddda0291a00a98bf1e92cfa26f3efb75c825cdc3840839
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 16
  GradientOrderbook.createOrder confirmed   Block: 43   Gas used: 207773 (1.73%)

Transaction sent: 0x5b429cedb138f749b4f4ee08c02dcc1fbee0cef30aef09a47f941664cdb9c145
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 7
  GradientOrderbook.createOrder confirmed   Block: 44   Gas used: 239440 (2.00%)

Transaction sent: 0x24d4f10c37839b71c4534a722778a70027549a1943f079bbbcf302a959c58b52
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 17
  GradientOrderbook.fulfillMatchedOrders confirmed (Invalid fill amount)   Block: 45   Gas used: 43542 (0.36%)

Transaction sent: 0x0d201597ae947f403e0123b7f990212c9ac966f15a50ef29da6c771001665567
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 18
  GradientOrderbook.fulfillMatchedOrders confirmed (Fill amount exceeds available)   Block: 46   Gas used: 43603 (0.36%)

Transaction sent: 0x2851a9d93501c306602625d1b7b12620fe6502c247525d0252fdda996e758269
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 19
  GradientOrderbook.fulfillMatchedOrders confirmed   Block: 47   Gas used: 157305 (1.31%)

Transaction sent: 0x77585e06abfba1e53f35725f15cf08e629000125081be243015c8f300ae2bd1f
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 8
  GradientOrderbook.cancelOrder confirmed   Block: 48   Gas used: 55308 (0.46%)

Transaction sent: 0x4c6ebedf089e456fa63a833a23f1bca1bc76bf5d6101321a1d49083deb347ded
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 20
  GradientOrderbook.fulfillMatchedOrders confirmed (Orders must be active)   Block: 49   Gas used: 30484 (0.25%)

tests/test_orderbook.py::test_fulfill_matched_orders PASSED
```

Annexes

Testing code:

```
from brownie import (
    reverts,
)


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


from scripts.deploy import (
    deploy_erc,
    deploy_orderbook
)


'''
    enum OrderType {
        Buy,
        Sell
    }
'''
```

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

    mock_token = deploy_erc(owner, "Mock", "MCK")
    orderbook = deploy_orderbook(owner)

    # mint some tokens
    mock_token.mint(other, 10e18)
    mock_token.approve(orderbook.address, 100e18, {"from": other})
    mock_token.mint(extra, 10e18)
    mock_token.approve(orderbook.address, 100e18, {"from": extra})

    priceXToken = 1e16

    with reverts("Invalid token"):
        orderbook.createOrder(
            0, ZERO_ADDRESS, 5,
            priceXToken, DAY_TIMESTAMP * 7,
            {"from": other})
    with reverts("Amount must be greater than 0"):
        orderbook.createOrder(
            0, mock_token.address, 0,
            priceXToken, DAY_TIMESTAMP * 7,
            {"from": other})
    with reverts("Invalid price range"):
        orderbook.createOrder(
```

```
            0, mock_token.address, 5,
            0, DAY_TIMESTAMP * 7,
            {"from": other})
    with reverts("TTL must be greater than 0"):
        orderbook.createOrder(
            0, mock_token.address, 5,
            priceXToken, 0,
            {"from": other})
    with reverts("Insufficient ETH sent"):
        orderbook.createOrder(
            0, mock_token.address, 5,
            priceXToken, DAY_TIMESTAMP * 7,
            {"from": other})


    tx = orderbook.createOrder(
            0, mock_token.address, 5,
            priceXToken, DAY_TIMESTAMP * 7,
            {"from": other, "value": priceXToken * 6})
    assert tx.events['OrderCreated'][0]['orderId'] == 0
    assert tx.events['OrderCreated'][0]['owner'] == other
    assert tx.events['OrderCreated'][0]['orderType'] == 0
    assert tx.events['OrderCreated'][0]['token'] == mock_token.address
    assert tx.events['OrderCreated'][0]['amount'] == 5
    assert tx.events['OrderCreated'][0]['price'] == priceXToken


    tx = orderbook.createOrder(
            1, mock_token.address, 10,
            priceXToken, DAY_TIMESTAMP * 7,
            {"from": extra})
```

```
    assert tx.events['OrderCreated'][0]['orderId'] == 1
    assert tx.events['OrderCreated'][0]['owner'] == extra
    assert tx.events['OrderCreated'][0]['orderType'] == 1
    assert tx.events['OrderCreated'][0]['token'] == mock_token.address
    assert tx.events['OrderCreated'][0]['amount'] == 10
    assert tx.events['OrderCreated'][0]['price'] == priceXToken

    # 0,05

def test_cancel_order(only_local):
    # Arrange
    owner = get_account(0)
    other = get_account(1)
    extra = get_account(2)

    mock_token = deploy_erc(owner, "Mock", "MCK")
    orderbook = deploy_orderbook(owner)

    # mint some tokens
    mock_token.mint(other, 10e18)
    mock_token.approve(orderbook.address, 100e18, {"from": other})
    mock_token.mint(extra, 10e18)
    mock_token.approve(orderbook.address, 100e18, {"from": extra})

    priceXToken = 1e16

    tx = orderbook.createOrder(
            0, mock_token.address, 5,
            priceXToken, DAY_TIMESTAMP * 7,
```

```
        {"from": other, "value": priceXToken * 6})
order_id_1 = tx.events['OrderCreated'][0]['orderId']


tx = orderbook.createOrder(
        1, mock_token.address, 10,
        priceXToken, DAY_TIMESTAMP * 7,
        {"from": extra})
order_id_2 = tx.events['OrderCreated'][0]['orderId']


with reverts("Order does not exist"):
    orderbook.cancelOrder(5, {"from": other})
with reverts("Not order owner"):
    orderbook.cancelOrder(order_id_1, {"from": owner})


tx = orderbook.cancelOrder(order_id_1, {"from": other})
assert tx.events['OrderCancelled'][0]['orderId'] == order_id_1


with reverts("Order not active"):
    orderbook.cancelOrder(order_id_1, {"from": other})


tx = orderbook.cancelOrder(order_id_2, {"from": extra})
assert tx.events['OrderCancelled'][0]['orderId'] == order_id_2


with reverts("Order not active"):
    orderbook.cleanupExpiredOrder(order_id_2)


tx = orderbook.createOrder(
        0, mock_token.address, 5,
        priceXToken, DAY_TIMESTAMP * 7,
```

```
            {"from": other, "value": priceXToken * 6})
    order_id = tx.events['OrderCreated'][0]['orderId']
    increase_timestamp(DAY_TIMESTAMP * 8)

    with reverts("Order expired"):
        orderbook.cancelOrder(order_id, {"from": other})

    tx = orderbook.createOrder(
            1, mock_token.address, 10,
            priceXToken, DAY_TIMESTAMP * 1,
            {"from": extra})
    order_id = tx.events['OrderCreated'][0]['orderId']
    increase_timestamp(DAY_TIMESTAMP * 5)

    tx = orderbook.cleanupExpiredOrder(order_id)
    assert tx.events['Transfer'][0]['from'] == orderbook.address
    assert tx.events['Transfer'][0]['to'] == extra
    assert tx.events['Transfer'][0]['value'] == 10

def test_fulfill_matched_orders(only_local):
    # Arrange
    owner = get_account(0)
    other = get_account(1)
    extra = get_account(2)

    mock_token = deploy_erc(owner, "Mock", "MCK")
    orderbook = deploy_orderbook(owner)

    # mint some tokens
```

```
mock_token.mint(other, 10e18)
mock_token.approve(orderbook.address, 100e18, {"from": other})
mock_token.mint(extra, 10e18)
mock_token.approve(orderbook.address, 100e18, {"from": extra})


priceXToken = 1e16


with reverts("Caller is not whitelisted"):
    orderbook.fulfillMatchedOrders(
        [], {"from": other})
with reverts("No order matches to fulfill"):
    orderbook.fulfillMatchedOrders(
        [], {"from": owner})
with reverts("Order does not exist"):
    orderbook.fulfillMatchedOrders(
        [[0,0,10]], {"from": owner})


tx = orderbook.createOrder(
        0, mock_token.address, 5,
        1.5e16, DAY_TIMESTAMP * 7,
        {"from": other, "value": priceXToken * 10})
order_id_1 = tx.events['OrderCreated'][0]['orderId']


tx = orderbook.createOrder(
        1, mock_token.address, 10,
        priceXToken, DAY_TIMESTAMP * 7,
        {"from": extra})
order_id_2 = tx.events['OrderCreated'][0]['orderId']
```

```
    with reverts("Invalid fill amount"):
        orderbook.fulfillMatchedOrders(
            [[order_id_1,order_id_2,0]], {"from": owner})
    with reverts("Fill amount exceeds available"):
        orderbook.fulfillMatchedOrders(
            [[order_id_1,order_id_2,15]], {"from": owner})


    tx = orderbook.fulfillMatchedOrders(
            [[order_id_1,order_id_2,5]], {"from": owner})
    assert tx.events['Transfer'][0]['from'] == orderbook.address
    assert tx.events['Transfer'][0]['to'] == other
    assert tx.events['Transfer'][0]['value'] == 5
    assert tx.events['OrderFulfilled'][0]['orderId'] == order_id_1
    assert tx.events['OrderFulfilled'][0]['amount'] == 5


    orderbook.cancelOrder(order_id_2, {"from": extra})


    with reverts("Orders must be active"):
        orderbook.fulfillMatchedOrders(
            [[order_id_1,order_id_2,5]], {"from": owner})
```

Technical Findings Summary

Findings

Vulnerability Level	Total	Pending	Not Apply	Acknowledged	Partially Fixed	Fixed
High	3					3
Medium	3			1		2
Low	2			1		1
Informational	2			2		

Assessment Results

Score Results

Review	Score
Global Score	**90/100**
Assure KYC	Not completed
Audit Score	90/100

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 the Gradient project, we inform you that the project has met the necessary security standards.

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 adGradient 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 adGradient, 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 serGradients 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 serGradients, 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 serGradients may access, and depend upon, multiple layers of third parties.