

Assure DeFi[®]

THE VERIFICATION **GOLD STANDARD**



Security Assessment

MetaHorseStaking



Date: 25/02/2024

Audit Status: FAILED

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 Low	Low-level vulnerabilities are primarily associated with outdated or unused code snippets that generally do not significantly impact execution, sometimes they can be ignored.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

<u>Insecure</u>	Poorly Secured	Secured	Well Secured
------------------------	-----------------------	----------------	---------------------

Scope

Target Code And Revision

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

Target Code And Revision

Project	Assure
Language	Solidity
Codebase	Testnet contract link: Metahorsestaking.sol - https://goerli.etherscan.io/address/0xD60Ae1f5fc279cb9e4619Bd517EF32C305FB9c08#code
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	<ul style="list-style-type: none">• 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



1. Owner can withdraw all the staked balances

Contract: Metahorsestaking

Functions: withdraw()

Issue: The admin of the contract can withdraw all the staked contract balances from all the users.

Fix: Don't allow withdrawal of the total balance, instead only the balance provided by the admin to use as a rewards for the users.

2. Division by zero

Contract: Metahorsestaking

Functions: airdrop()

Issue: When the `_total` value retrieved from the `getAccumulatedPoint` function equals 0, a division by zero error occurs, leading to incorrect function behavior.

Fix: Implement a check to ensure the `_total` amount is greater than 0 before proceeding with the token transfer.

3. Misdirected token transfers

Contract: Metahorsestaking

Functions: airdrop()

Issue: The `airDrop()` function currently sends the calculated token amounts to the contract's administrator (admin) instead of the intended recipients, the stakers. This misdirection of funds is a critical flaw, leading to the potential loss of rewards that should be distributed to users participating in staking.

Fix: The transfer of accumulated points should be directed towards the intended stakers. This can be achieved by replacing `msg.sender` with `stakers[i]` within the transfer call inside the `airDrop()` function. This ensures that the rewards are distributed to the correct recipients based on their accumulated points.

4. Misdirected stake deposits

Contract: Metahorsetaking

Functions: airdrop()

Issue: The staking implementation has a critical flaw where the msg.value, representing the staked amount by users, is sent directly to a treasuryAddress rather than being kept within the staking contract. This approach leads to discrepancies in the contract's balance, affecting the functionality of unstake and emergencyUnstake operations. These functions rely on the contract's balance to validate and process withdrawals, and by diverting funds away from the contract, it can result in a state where users are unable to withdraw their staked amounts despite the contract logic suggesting otherwise.

Fix: Adjust the .call method to target address(this), ensuring that the staked amount is sent to the staking contract's address itself. It's advised to specify the treasury address as the recipient in the _to argument.



1. Ensuring positive stake amounts in 'stake' function

Contract: Metahorsetaking

Functions: airdrop()

Issue: The current implementation of the stake() function lacks a crucial validation check to ensure that the msg.value—the amount of Ether being staked—is greater than 0. This oversight allows for the initiation of stakes with a value of 0, potentially leading to undesirable contract behavior.

Fix: To mitigate this issue and enhance the contract's robustness, it is essential to introduce a require() statement at the beginning of the stake() function. This condition will validate that msg.value is greater than 0, thereby preventing the initiation of zero-value stakes.



1. Zero address Validation Missing

Contract: Metahorse staking

Functions:

setTreasuryAddress:

TreasuryAddress can be 0 address.

TransferAdmin:

Newadmin can be 0 address.

Withdraw():

_to can be 0 address.

Issue: During the audit we detected that multiple variables can be set to 0 address leading to potentially lost funds or unintended behaviors.

Fix: Implement checks to validate that the address is not the zero address (0x0) in the affected functions/contracts.



No Informational severity issues were found.

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.*

MetaHorseStaking contract tests:

```
contract: MetaHorseStaking - 75.7%
  MetaHorseStaking.startStake - 100.0%
  MetaHorseStaking.emergencyUnstake - 87.5%
  MetaHorseStaking.withdraw - 87.5%
  MetaHorseStaking.unstake - 83.3%
  MetaHorseStaking.getAvailableStakeBalance - 75.0%
  MetaHorseStaking.getStakeBalance - 75.0%
  MetaHorseStaking.stake - 75.0%
  ReentrancyGuard._nonReentrantBefore - 75.0%
  MetaHorseStaking.getAccumulatedPoint - 69.0%
  MetaHorseStaking.airDrop - 58.3%
```

```
tests/test_meta.py::test_airdrop RUNNING
Transaction sent: 0x50122fc586f57bcf6ef0fab7c96fd33f85b1e2da1c1b7538ee6c0803b8bfed12
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 0
ERC20Mock.constructor confirmed Block: 1 Gas used: 619873 (5.17%)
ERC20Mock deployed at: 0x3194cBDC3dbcd3E11a07892e7bA5c3394048Cc87

Transaction sent: 0x87542443e26feccd54ef902a7b0979555db117121422811e081884e1a9b889f
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 1
MetaHorseStaking.constructor confirmed Block: 2 Gas used: 1424477 (11.87%)
MetaHorseStaking deployed at: 0x602C71e4D4C47a042Ee7f46E0aee17F94A3bA0B6

Transaction sent: 0xda3e56e2f375d8c2ea7d4edd3290ae723667a027e7059710b3f2f318dc65c25b
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 2
MetaHorseStaking.setTreasuryAddress confirmed Block: 3 Gas used: 30922 (0.26%)

Transaction sent: 0x3f7ea48e2d6fb98a736ec2264751866f2750907dadbe630e132d1d2e94fc274b
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 0
MetaHorseStaking.airDrop confirmed (onlyAdmin: You are not Admin.) Block: 4 Gas used: 28051 (0.23%)

Transaction sent: 0x71185ac71bcd98e0a12eb0c872483034891e9e82fdd508d48e715c28fc80ad0a
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 1
MetaHorseStaking.startStake confirmed (onlyAdmin: You are not Admin.) Block: 5 Gas used: 28075 (0.23%)

Transaction sent: 0x49ca6188654a8815db9b7b3958d99fb3714f70d1688afe546301c09fb3ac6853
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 3
MetaHorseStaking.startStake confirmed Block: 6 Gas used: 65474 (0.55%)

Transaction sent: 0xda790c62a9a2c63d576aea0243d3ee23756dd6ddc2559f19cd80fa6c0bf1edb7
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 4
MetaHorseStaking.startStake confirmed (startStake: Staking already started!) Block: 7 Gas used: 28898 (0.24%)

Transaction sent: 0x67d2972781fb821b21e9933bde20b10d200f4a29a7e86851419ca1320165d2cf
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 5
MetaHorseStaking.airDrop confirmed (Now staking is ongoing) Block: 8 Gas used: 28939 (0.24%)

Transaction sent: 0x61acc73ec7014f9805b6eac6354d13c7e90a668890df3696f2c722e704a86ce0
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 2
MetaHorseStaking.stake confirmed Block: 9 Gas used: 239704 (2.00%)

Transaction sent: 0x30592a53f60c2ec9f27dc56d985b2bf785261625de80106a76ea0f83a5cd3d8d
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 0
MetaHorseStaking.stake confirmed Block: 10 Gas used: 209722 (1.75%)

Transaction sent: 0xf2d3c203130043cd290bb256b15d66f2c3dd7cf95a3a734cc5f773c86ca6e45a
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 6
ERC20Mock.mint confirmed Block: 11 Gas used: 65637 (0.55%)

Transaction sent: 0xda2a1ef0dc8118f11c550a5e20bf3f006e7c92118186b3707b5c43686c45c502
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 7
MetaHorseStaking.airDrop confirmed Block: 12 Gas used: 98326 (0.82%)

tests/test_meta.py::test_airdrop PASSED
```



```

tests/test_meta.py::test_stake RUNNING
Transaction sent: 0x737c73ebf4aa18d55318c514adcbb4bff4f6bf3e5ef88d81ed9275b51bcbccaf
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 8
ERC20Mock.constructor confirmed Block: 13 Gas used: 619873 (5.17%)
ERC20Mock deployed at: 0x420b1099B9eF5baba6D92029594eF45E19A04A4A

Transaction sent: 0x165f44e3e8767f78f2191999fa5c5824cac733643368f181f8c94447c707f842
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 9
MetaHorseStacking.constructor confirmed Block: 14 Gas used: 1424477 (11.87%)
MetaHorseStacking deployed at: 0xa3853dDCd2E3fC28e8E130288F2a8D8d5EE37472

Transaction sent: 0x0400c14fc8789062f98a44bc12a16bb7cd340aef2beef2c665531f934eaf7b
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 10
MetaHorseStacking.setTreasuryAddress confirmed Block: 15 Gas used: 30922 (0.26%)

Transaction sent: 0xae40c3ffd359100679a0c253da40847fad97701431781069052c7400d3731cb8
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 11
MetaHorseStacking.stake confirmed (moreThanZero: Invalid stacking mode.) Block: 16 Gas used: 27494 (0.23%)

Transaction sent: 0x91a8e261f901df182396f4a05a0b44a7554f42c599ee7e72c8ba07fcbc047f4b
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 12
MetaHorseStacking.stake confirmed (Stake isn't allowed this time.) Block: 17 Gas used: 28289 (0.24%)

Transaction sent: 0xc68541af2a2862f71b0de3132c9dbff1fb3acal6dbb60bb928a8e6e0d214286e
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 13
MetaHorseStacking.startStake confirmed Block: 18 Gas used: 65474 (0.55%)

Transaction sent: 0x32ba3f2a4ca5a89e262feffeeefca208c2c89b0b0d7a0212bfd131360a37a79e
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 3
MetaHorseStacking.stake confirmed Block: 19 Gas used: 239663 (2.00%)

Transaction sent: 0x9c1250309075f8926ccfa543542c9a6daf5d4a9d45de6f9cbff653bbab8834a
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 4
MetaHorseStacking.emergencyUnstake confirmed (Sorry, wait for the server to deposit.) Block: 20 Gas used: 42215 (0.35%)

Transaction sent: 0x2769394977043c81627e1f26c854c87222a435c7f686b6e0a3cdaac0707fd8de
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 5
MetaHorseStacking.emergencyUnstake confirmed Block: 22 Gas used: 50065 (0.42%)

Transaction sent: 0xd87d329f986a0547a976824a9df4ca30be190bc20e01b6aa970111048b0331e1
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 6
MetaHorseStacking.stake confirmed (You have unstaked within 1 day. Try agin after 1 day.) Block: 23 Gas used: 29284 (0.24%)

Transaction sent: 0x4861a1efbd3d3e72268a71596787d7590ee4851c1b6506a75d25796af24c2e5f
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 7
MetaHorseStacking.stake confirmed Block: 24 Gas used: 189638 (1.58%)

tests/test_meta.py::test_stake PASSED

```

```

tests/test_meta.py::test_unstake RUNNING
Transaction sent: 0x0c5a0a8dc2d96cc8b9475e4332ab03ab8b7216cd1f6fd4281db3a43b4764c38f
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 15
ERC20Mock.constructor confirmed Block: 25 Gas used: 619873 (5.17%)
ERC20Mock deployed at: 0x30375B532345801c88c2AD12541b09E9Aa53A93d

Transaction sent: 0xefdbaf34729d18d5ca109e2d4e75b733db774e8b8e2d05e8d51859fbf86fd713
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 16
MetaHorseStacking.constructor confirmed Block: 26 Gas used: 1424477 (11.87%)
MetaHorseStacking deployed at: 0x26f153358B1C6a4C0B660eDd694a0555A9F1cce3

Transaction sent: 0xcf2df775fe552cbb81474c9d3772bb17f6162a61ceb86b9d31ccd9c501528b58
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 17
MetaHorseStacking.setTreasuryAddress confirmed Block: 27 Gas used: 30922 (0.26%)

Transaction sent: 0x2336d8acb960d0164ffb74e1c1c525c82f80edcfbd0b1333f4d3e68f74589948
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 18
MetaHorseStacking.startStake confirmed Block: 28 Gas used: 65474 (0.55%)

Transaction sent: 0x893549154a6a8eaca31e4b5d2422c3c621337245104c92d9dfecbfc68bb3d212
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 8
MetaHorseStacking.stake confirmed Block: 29 Gas used: 239663 (2.00%)

Transaction sent: 0x0d1f392b93106b270d627ec4b7e2826d327f220dd8aa36a9cca6693eff07e131
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 9
MetaHorseStacking.unstake confirmed (Sorry, wait for the server to deposit.) Block: 30 Gas used: 40110 (0.33%)

Transaction sent: 0xd7fc567292ec51f600a66c45994ed2a37ab50d658f7e1fa22bc1abbd9c9b13d08
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 10
MetaHorseStacking.unstake confirmed Block: 32 Gas used: 85127 (0.71%)

tests/test_meta.py::test_unstake PASSED

```

```

tests/test_meta.py::test_emergency_unstake RUNNING
Transaction sent: 0x34278338a9981ceda62702b90ec3180843cd8ef54daf811fecb0090c7f744451
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 20
ERC20Mock.constructor confirmed Block: 33 Gas used: 619873 (5.17%)
ERC20Mock deployed at: 0xdCF93F11ef216cEC9C07fd31dD801c9b2b39Afb4

Transaction sent: 0x0ba6dd4b45621ce4c8db44961e6a2e53e2bb2de44074d27d6ba1d030a5b0419d
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 21
MetaHorseStacking.constructor confirmed Block: 34 Gas used: 1424477 (11.87%)
MetaHorseStacking deployed at: 0x8cb61491F1859f53438918F1A5aFCA542Af9D397

Transaction sent: 0x792721620b97cec186b238a2adb3c7dfb4a65fb52e8679ba638a1cee97007d35
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 22
MetaHorseStacking.setTreasuryAddress confirmed Block: 35 Gas used: 30922 (0.26%)

Transaction sent: 0x601d17d4665e6d351420efb1e9714372d1012d384f1947f816af6529346e4f73
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 23
MetaHorseStacking.startStake confirmed Block: 36 Gas used: 65474 (0.55%)

Transaction sent: 0x4144d053946b52d7d83b19d534bb40875b74103bfa29ac28e3ebad5d655b90de
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 11
MetaHorseStacking.stake confirmed Block: 37 Gas used: 239663 (2.00%)

Transaction sent: 0x0325ff05db586bfbea432286e4b02a9c3af559e2420bbe5cdac478eb03c81cb1
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 12
MetaHorseStacking.emergencyUnstake confirmed (Sorry, wait for the server to deposit. ) Block: 38 Gas used: 42215 (0.35%)

Transaction sent: 0xebba405e45006b457e3609acd83cba50183906f498114d69e047cab7f6ec9bb
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 13
MetaHorseStacking.emergencyUnstake confirmed Block: 40 Gas used: 50065 (0.42%)

tests/test_meta.py::test_emergency_unstake PASSED

```

```

tests/test_meta.py::test_withdraw RUNNING
Transaction sent: 0x9ed43035e7e133e842a5707cc3f1f960376054a944bcc71a2dccee66adafde18
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 25
ERC20Mock.constructor confirmed Block: 41 Gas used: 619873 (5.17%)
ERC20Mock deployed at: 0x654f70d8442EA18904FA1AD79114f7250F7E9336

Transaction sent: 0x55b92ee57a323baf0222d29cbccaeaf4540139fab57e1fbb44218b8023e5a10b
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 26
MetaHorseStacking.constructor confirmed Block: 42 Gas used: 1424477 (11.87%)
MetaHorseStacking deployed at: 0xAdeD61D42dE86f9058386D1D0d739d20C7eAfc43

Transaction sent: 0xa90e1488643652079f8d4e09911a6a1f12b538ddaad67e785e0764e25304b4c6
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 27
MetaHorseStacking.setTreasuryAddress confirmed Block: 43 Gas used: 30922 (0.26%)

Transaction sent: 0x1a11b5fcc55b5c4f5a06dc37436747dc8356439279ee6b8bdb5307973f5dfe3b
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 14
MetaHorseStacking.withdraw confirmed (onlyAdmin: You are not Admin.) Block: 44 Gas used: 28886 (0.24%)

Transaction sent: 0x7fdce2375eeeb83ad8609ada18944155c9bee72310f7020af74617bd7d4321e3
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 28
MetaHorseStacking.withdraw confirmed (_withdraw: The withdraw value must be smaller than totalStakedAmount) Block: 45 Gas used: 28941 (0.24%)

Transaction sent: 0x50fcb598ad80944f9a32f3e6eb411fb184deb4598c7d5c129ad1e60111ae494f
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 30
MetaHorseStacking.withdraw confirmed Block: 47 Gas used: 32944 (0.27%)

tests/test_meta.py::test_withdraw PASSED

```

Annexes

Testing code:

```
from brownie import (
    reverts,
)

from scripts.helpful_scripts import (
    DAY_TIMESTAMP,
    get_account,
    evm_increase_time,
)

from scripts.deploy import (
    deploy_erc,
    deploy_meta,
)

def test_airdrop(only_local):
    # Arrange
    owner = get_account(0)
    other = get_account(1)
    treasury = get_account(2)
    extra = get_account(3)

    erc = deploy_erc(owner, "test", "test")
    meta = deploy_meta(owner, erc.address)
    meta.setTreasuryAddress(treasury)

    # Assert
    with reverts("onlyAdmin: You are not Admin."):
        meta.airDrop({"from": other})
    with reverts("onlyAdmin: You are not Admin."):
        meta.startStake({"from": other})
    meta.startStake({"from": owner})
    with reverts("startStake: Staking already started!"):
        meta.startStake({"from": owner})
    with reverts("Now staking is ongoing"):
        meta.airDrop({"from": owner})

    meta.stake(1, {"from": other, "value": 1e18})
    meta.stake(2, {"from": extra, "value": 1e18})
    erc.mint(meta.address, 1e18)
    evm_increase_time(DAY_TIMESTAMP * 100)
    meta.airDrop({"from": owner})

def test_stake(only_local):
```

```

# Arrange
owner = get_account(0)
other = get_account(1)
treasury = get_account(2)

erc = deploy_erc(owner, "test", "test")
meta = deploy_meta(owner, erc.address)
meta.setTreasuryAddress(treasury)

with reverts("moreThanZero: Invalid stacking mode."):
    meta.stake(4, {"from": owner})
with reverts("Stake isn't allowed this time."):
    meta.stake(0, {"from": owner})
meta.startStake({"from": owner})
meta.stake(0, {"from": other, "value": 1e18})
with reverts("Sorry, wait for the server to deposit. "):
    meta.emergencyUnstake({"from": other})
owner.transfer(meta.address, "10 ether")
assert meta.totalStaked() == 1e18
meta.emergencyUnstake({"from": other})
assert meta.totalStaked() == 0
with reverts("You have unstaked within 1 day. Try again after 1 day."):
    meta.stake(0, {"from": other, "value": 1e18})

evm_increase_time(DAY_TIMESTAMP * 2)
meta.stake(3, {"from": other, "value": 1e18})

def test_unstake(only_local):
    # Arrange
    owner = get_account(0)
    other = get_account(1)
    treasury = get_account(2)

    erc = deploy_erc(owner, "test", "test")
    meta = deploy_meta(owner, erc.address)
    meta.setTreasuryAddress(treasury)

    meta.startStake({"from": owner})
    meta.stake(0, {"from": other, "value": 1e18})
    evm_increase_time(DAY_TIMESTAMP * 16)
    with reverts("Sorry, wait for the server to deposit. "):
        meta.unstake({"from": other})
    owner.transfer(meta.address, "10 ether")
    assert meta.totalStaked() == 1e18
    meta.unstake({"from": other})
    assert meta.totalStaked() == 0

def test_emergency_unstake(only_local):
    # Arrange

```

```

owner = get_account(0)
other = get_account(1)
treasury = get_account(2)

erc = deploy_erc(owner, "test", "test")
meta = deploy_meta(owner, erc.address)
meta.setTreasuryAddress(treasury)

meta.startStake({"from": owner})
meta.stake(0, {"from": other, "value": 1e18})
evm_increase_time(DAY_TIMESTAMP * 1)
with reverts("Sorry, wait for the server to deposit. "):
    meta.emergencyUnstake({"from": other})
owner.transfer(meta.address, "10 ether")
assert meta.totalStaked() == 1e18
meta.emergencyUnstake({"from": other})
assert meta.totalStaked() == 0

def test_withdraw(only_local):
    # Arrange
    owner = get_account(0)
    other = get_account(1)
    treasury = get_account(2)





    erc = deploy_erc(owner, "test", "test")
    meta = deploy_meta(owner, erc.address)
    meta.setTreasuryAddress(treasury)

    with reverts("onlyAdmin: You are not Admin."):
        meta.withdraw(other, 1e18, {"from": other})
    with reverts("_withdraw: The withdraw value must be smaller than
totalStakedAmount"):
        meta.withdraw(other, 1e18, {"from": owner})
    owner.transfer(meta.address, "10 ether")
    meta.withdraw(other, 1e18, {"from": owner})

```


Technical Findings Summary

Findings

Vulnerability Level	Total	Pending	Not Apply	Acknowledged	Partially Fixed	Fixed
 High	4					
 Medium	1					
 Low	1					
 Informational	0					

Assessment Results

Score Results

Review	Score
Audit Score	5/100
Assure KYC	Completed: https://assuredefi.com/projects/metahorse-unity/
Audit Score	0/100

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 FAILED

Following our comprehensive security audit of the project MetaHorseStaking, we inform you that the project has not met the required security standards. The audit has uncovered three high+medium vulnerabilities that imply a total risk in the staked funds, leaving all of the users funds at risk.

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.