DeltaHub Capital

SMART CONTRACT AUDIT

ZOKYO.

Mar 23, 2021 | v. 1.0

PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.

TECHNICAL SUMMARY

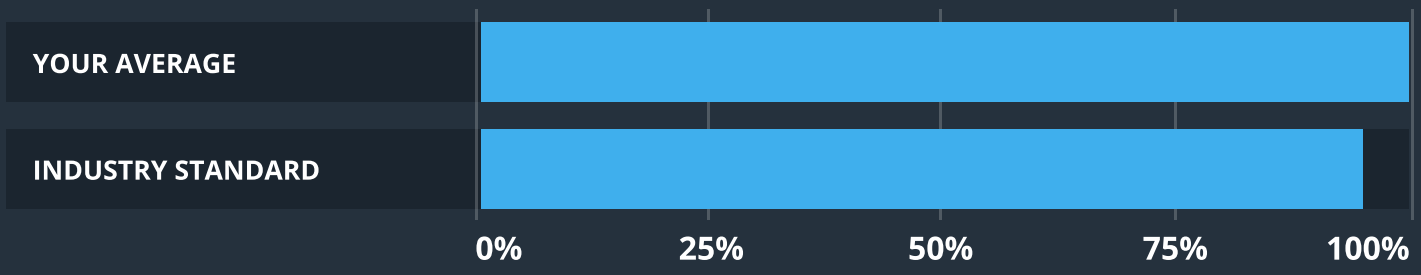
This document outlines the overall security of the DeltaHub smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document the DeltaHub smart contract codebase for quality, security, and correctness.

Contract Status

There were no critical issues found during the audit.

Testable Code



Testable code is 100%, which is above the industry standard of 95%.

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.

TABLE OF CONTENTS

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the DeltaHub repo commit – 632e0238db2e4f0c9d209cf099dd9e2c594b5c24.
Requirements: DeltaHub Pitch Deck.

Throughout the review process, care was taken to ensure that the token contract:

- Implements and adheres to existing Token standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of gas, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of DeltaHub smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1	Due diligence in assessing the overall code quality of the codebase.	3	Testing contract logic against common and uncommon attack vectors.
2	Cross-comparison with other, similar smart contracts by industry leaders.	4	Thorough, manual review of the codebase, line-by-line.

SUMMARY

There were no critical issues found during the **manual audit** and **automated testing** analysis. The smart contracts are well-composed, operate as expected, and are production-ready. It follows best practices in efficient use of gas, without unnecessary waste all the methods are safe from known types of attacks.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the ability of the contract to compile or operate in a significant way.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

Order of Functions

INFORMATIONAL | UNRESOLVED

The functions in DeltaHubStaking contract are not grouped according to their visibility and order.

Functions should be grouped according to their visibility and ordered in the following way:

- constructor;
- fallback function (if exists);
- external;
- public;
- internal;
- private.

Ordering helps readers identify which functions they can call and to find the constructor and fallback definitions easier.

Recommendation:
Consider changing functions order.

Action Steps:
Change functions order according to solidity documentations: Order of Functions.

Calculations error up to 100 000 Wei

INFORMATIONAL | UNRESOLVED

In the calculation of the reward for the users, there is a calculation error of up to 100 000 WEI.

Notice:
Templates of calculation reward

The next table shows how the reward will be calculated in the developed smart contract. Eg. only one user stake.

set rewards		day	reward	calc
70		1	10	70 / 7
		2	10	70 / 7
		3	10	70 / 7
	30	4	10	(40 + 30) / 7
		5	10	(40 + 30) / 7
		6	10	(40 + 30) / 7
		7	10	(40 + 30) / 7
		8	10	(40 + 30) / 7
		9	10	(40 + 30) / 7
		10	10	(40 + 30) / 7

Total reward: 100
If a user unstake on day 7, he will get **70 tokens**.

The below table shows how the reward can be calculated in another way.

set rewards		day	reward	calc
70		1	10	70 / 7
		2	10	70 / 7
		3	10	70 / 7
	30	4	10 + 4.28	70 / 7 + 30 / 7
		5	10 + 4.28	70 / 7 + 30 / 7
		6	10 + 4.28	70 / 7 + 30 / 7
		7	10 + 4.28	70 / 7 + 30 / 7
		8	4.28	30 / 7
		9	4.28	30 / 7
		10	4.28	30 / 7

Total reward: 100
If a user unstake on day 7, he will get **87.12 tokens**.

Make sure that you choose the correct template for calculating rewards.

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

As part of our work assisting DeltaHub in verifying the correctness of their contract code, our team was responsible for writing integration tests using the Truffle testing framework.

Tests were based on the functionality of the code, as well as a review of the DeltaHub contract requirements (for Audit) for details about issuance amounts and how the system handles these.

Tests are written by DeltaHub and Zokyo's Security teams

Execution Report

```
yarn run v1.22.5
$ truffle test

Compiling your contracts...
===========================
> Compiling @openzeppelin/contracts/access/Ownable.sol
> Compiling @openzeppelin/contracts/math/Math.sol
> Compiling @openzeppelin/contracts/math/SafeMath.sol
> Compiling @openzeppelin/contracts/token/ERC20/ERC20.sol
> Compiling @openzeppelin/contracts/token/ERC20/IERC20.sol
> Compiling @openzeppelin/contracts/token/ERC20/SafeERC20.sol
> Compiling @openzeppelin\contracts\GSN\Context.sol
> Compiling @openzeppelin\contracts\math\SafeMath.sol
> Compiling @openzeppelin\contracts\token\ERC20\IERC20.sol
> Compiling @openzeppelin\contracts\utils\Address.sol
> Compiling .\contracts\DeltaHubStaking.sol
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\mock\DeltaHubMock.sol
> Compilation warnings encountered:

    @openzeppelin/contracts/access/Ownable.sol:26:5: Warning: Visibility for constructor is ignored. If you
want the contract to be non-deployable, making it "abstract" is sufficient.
    constructor () internal {
```

```
    ^ (Relevant source part starts here and spans across multiple lines).
,@openzeppelin/contracts/token/ERC20/ERC20.sol:55:5: Warning: Visibility for constructor is ignored. If
you want the contract to be non-deployable, making it "abstract" is sufficient.
    constructor (string memory name_, string memory symbol_) public {
    ^ (Relevant source part starts here and spans across multiple lines).
,/D/Projects/Audit/Deltahub/deltahub-contracts/contracts/DeltaHubStaking.sol:54:5: Warning: Visibility
for constructor is ignored. If you want the contract to be non-deployable, making it "abstract" is
sufficient.
    constructor(IERC20 _dhc) public {
    ^ (Relevant source part starts here and spans across multiple lines).
,/D/Projects/Audit/Deltahub/deltahub-contracts/contracts/Migrations.sol:13:5: Warning: Visibility for
constructor is ignored. If you want the contract to be non-deployable, making it "abstract" is sufficient.
    constructor() public {
    ^ (Relevant source part starts here and spans across multiple lines).
,/D/Projects/Audit/Deltahub/deltahub-contracts/contracts/mock/DeltaHubMock.sol:8:5: Warning: Visibility
for constructor is ignored. If you want the contract to be non-deployable, making it "abstract" is
sufficient.
    constructor() public ERC20("DelataHub Capital", "DHC") {
    ^ (Relevant source part starts here and spans across multiple lines).

> Artifacts written to C:\Users\roman\AppData\Local\Temp\test--17152-5hKygGAaf0LM
> Compiled successfully using:
   - solc: 0.7.5+commit.eb77ed08.Emscripten.clang


  Contract: DeltaHubStaking
    ✓ has a default owner (117ms)
    ✓ lets a new owner change the message (296ms)
    ✓ prevents non-owners from transferring (146ms)
    ✓ should not allow enter if not enough approve (2537ms)
    ✓ should not allow withraw more than what you have (1154ms)
    ✓ should work with more than one participant (2121ms)
    ✓ Two stakers with the same stakes wait 1 w (2743ms)
    1) Two stakers with the different (1:3) stakes wait 1 w


    Events emitted during test:
    ---------------------------

    IERC20.Transfer(
```

```
  from: <indexed> 0x0000000000000000000000000000000000000000 (type: address),
  to: <indexed> 0xB349FB172D6D5f693b0aA1C6eEc4c61cFd6846f4 (type: address),
  value: 100000000000000000000000000000000 (type: uint256)
)

IERC20.Transfer(
  from: <indexed> 0x0000000000000000000000000000000000000000 (type: address),
  to: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address),
  value: 100000000000000000000000000000000 (type: uint256)
)

Ownable.OwnershipTransferred(
  previousOwner: <indexed> 0x0000000000000000000000000000000000000000 (type: address),
  newOwner: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address)
)

IERC20.Transfer(
  from: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address),
  to: <indexed> 0xd45A464a2412A2f83498d13635698a041b9dBe9b (type: address),
  value: 100000000000000000000000 (type: uint256)
)

IERC20.Transfer(
  from: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address),
  to: <indexed> 0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0 (type: address),
  value: 100000000000000000000 (type: uint256)
)

IERC20.Transfer(
  from: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address),
  to: <indexed> 0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b (type: address),
  value: 100000000000000000000 (type: uint256)
)

IERC20.Transfer(
  from: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address),
  to: <indexed> 0xE11BA2b4D45Eaed5996Cd0823791E0C93114882d (type: address),
  value: 100000000000000000000 (type: uint256)
```

```
  )

  IERC20.Transfer(
    from: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address),
    to: <indexed> 0x95cED938F7991cd0dFcb48F0a06a40FA1aF46EBC (type: address),
    value: 100000000000000000000 (type: uint256)
  )

  IERC20.Approval(
    owner: <indexed> 0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0 (type: address),
    spender: <indexed> 0xd45A464a2412A2f83498d13635698a041b9dBe9b (type: address),
    value: 57896044618658097711785492504343953926634992332820282019728792003956564819968
(type: uint256)
  )

  IERC20.Approval(
    owner: <indexed> 0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b (type: address),
    spender: <indexed> 0xd45A464a2412A2f83498d13635698a041b9dBe9b (type: address),
    value: 57896044618658097711785492504343953926634992332820282019728792003956564819968
(type: uint256)
  )

  IERC20.Approval(
    owner: <indexed> 0xE11BA2b4D45Eaed5996Cd0823791E0C93114882d (type: address),
    spender: <indexed> 0xd45A464a2412A2f83498d13635698a041b9dBe9b (type: address),
    value: 57896044618658097711785492504343953926634992332820282019728792003956564819968
(type: uint256)
  )

  IERC20.Approval(
    owner: <indexed> 0xd03ea8624C8C5987235048901fB614fDcA89b117 (type: address),
    spender: <indexed> 0xd45A464a2412A2f83498d13635698a041b9dBe9b (type: address),
    value: 57896044618658097711785492504343953926634992332820282019728792003956564819968
(type: uint256)
  )

  IERC20.Transfer(
    from: <indexed> 0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0 (type: address),
```

```
    to: <indexed> 0xd45A464a2412A2f83498d13635698a041b9dBe9b (type: address),
    value: 1000000000000000000 (type: uint256)
  )

  IERC20.Approval(
    owner: <indexed> 0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0 (type: address),
    spender: <indexed> 0xd45A464a2412A2f83498d13635698a041b9dBe9b (type: address),
    value: 57896044618658097711785492504343953926634992332820282019727792003956564819968
(type: uint256)
  )

  IERC20.Transfer(
    from: <indexed> 0x0000000000000000000000000000000000000000 (type: address),
    to: <indexed> 0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0 (type: address),
    value: 1000000000000000000 (type: uint256)
  )

  DeltaHubStaking.Staked(
    user: <indexed> 0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0 (type: address),
    amount: 1000000000000000000 (type: uint256)
  )

  IERC20.Transfer(
    from: <indexed> 0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b (type: address),
    to: <indexed> 0xd45A464a2412A2f83498d13635698a041b9dBe9b (type: address),
    value: 3000000000000000000 (type: uint256)
  )

  IERC20.Approval(
    owner: <indexed> 0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b (type: address),
    spender: <indexed> 0xd45A464a2412A2f83498d13635698a041b9dBe9b (type: address),
    value: 57896044618658097711785492504343953926634992332820282019725792003956564819968
(type: uint256)
  )

  IERC20.Transfer(
    from: <indexed> 0x0000000000000000000000000000000000000000 (type: address),
    to: <indexed> 0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b (type: address),
```

```
  value: 3000000000000000000 (type: uint256)
)

DeltaHubStaking.Staked(
  user: <indexed> 0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b (type: address),
  amount: 3000000000000000000 (type: uint256)
)

---------------------------

✓ Two stakers with the different (1:3) stakes wait 2 weeks (4657ms)
2) Three stakers with the different (1:3:5) stakes wait 3 weeks

Events emitted during test:
---------------------------

IERC20.Transfer(
  from: <indexed> 0x0000000000000000000000000000000000000000 (type: address),
  to: <indexed> 0x81920CAF1F99Bb0f7D72Fdfba840cff21d63CcC5 (type: address),
  value: 100000000000000000000000000000000 (type: uint256)
)

IERC20.Transfer(
  from: <indexed> 0x0000000000000000000000000000000000000000 (type: address),
  to: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address),
  value: 100000000000000000000000000000000 (type: uint256)
)

Ownable.OwnershipTransferred(
  previousOwner: <indexed> 0x0000000000000000000000000000000000000000 (type: address),
  newOwner: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address)
)

IERC20.Transfer(
  from: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address),
  to: <indexed> 0xd99748782d7643b00C36a4Bb296C4A099dF98Ff3 (type: address),
  value: 100000000000000000000000 (type: uint256)
)
```

```
  IERC20.Transfer(
    from: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address),
    to: <indexed> 0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0 (type: address),
    value: 100000000000000000000 (type: uint256)
  )

  IERC20.Transfer(
    from: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address),
    to: <indexed> 0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b (type: address),
    value: 100000000000000000000 (type: uint256)
  )

  IERC20.Transfer(
    from: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address),
    to: <indexed> 0xE11BA2b4D45Eaed5996Cd0823791E0C93114882d (type: address),
    value: 100000000000000000000 (type: uint256)
  )

  IERC20.Transfer(
    from: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address),
    to: <indexed> 0xd03ea8624C8C5987235048901fB614fDcA89b117 (type: address),
    value: 100000000000000000000 (type: uint256)
  )

  IERC20.Transfer(
    from: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address),
    to: <indexed> 0x95cED938F7991cd0dFcb48F0a06a40FA1aF46EBC (type: address),
    value: 100000000000000000000 (type: uint256)
  )

  IERC20.Approval(
    owner: <indexed> 0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0 (type: address),
    value: 57896044618658097711785492504343953926634992332820282019728792003956564819968
(type: uint256)
  )

  IERC20.Approval(
    owner: <indexed> 0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b (type: address),
```

```
    spender: <indexed> 0xd99748782d7643b00C36a4Bb296C4A099dF98Ff3 (type: address),
    value: 57896044618658097711785492504343953926634992332820282019728792003956564819968
(type: uint256)
  )

  IERC20.Approval(
    owner: <indexed> 0xE11BA2b4D45Eaed5996Cd0823791E0C93114882d (type: address),
    spender: <indexed> 0xd99748782d7643b00C36a4Bb296C4A099dF98Ff3 (type: address),
    value: 57896044618658097711785492504343953926634992332820282019728792003956564819968
(type: uint256)
  )

  IERC20.Approval(
    owner: <indexed> 0xd03ea8624C8C5987235048901fB614fDcA89b117 (type: address),
    spender: <indexed> 0xd99748782d7643b00C36a4Bb296C4A099dF98Ff3 (type: address),
    value: 57896044618658097711785492504343953926634992332820282019728792003956564819968
(type: uint256)
  )

  DeltaHubStaking.RewardAdded(
    reward: 7200000000000000000000 (type: uint256)
  )

  IERC20.Transfer(
    from: <indexed> 0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0 (type: address),
    to: <indexed> 0xd99748782d7643b00C36a4Bb296C4A099dF98Ff3 (type: address),
    value: 1000000000000000000 (type: uint256)
  )

  IERC20.Approval(
    owner: <indexed> 0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0 (type: address),
    spender: <indexed> 0xd99748782d7643b00C36a4Bb296C4A099dF98Ff3 (type: address),
    value: 57896044618658097711785492504343953926634992332820282019727792003956564819968
(type: uint256)
  )

  IERC20.Transfer(
    from: <indexed> 0x0000000000000000000000000000000000000000 (type: address),
```

```
    to: <indexed> 0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0 (type: address),
    value: 1000000000000000000 (type: uint256)
  )

  DeltaHubStaking.Staked(
    user: <indexed> 0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0 (type: address),
    amount: 1000000000000000000 (type: uint256)
  )

  IERC20.Transfer(
    from: <indexed> 0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b (type: address),
    to: <indexed> 0xd99748782d7643b00C36a4Bb296C4A099dF98Ff3 (type: address),
    value: 3000000000000000000 (type: uint256)
  )

  IERC20.Approval(
    owner: <indexed> 0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b (type: address),
    spender: <indexed> 0xd99748782d7643b00C36a4Bb296C4A099dF98Ff3 (type: address),
    value: 57896044618658097711785492504343953926634992332820282019725792003956564819968
(type: uint256)
  )

  IERC20.Transfer(
    from: <indexed> 0x0000000000000000000000000000000000000000 (type: address),
    to: <indexed> 0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b (type: address),
    value: 3000000000000000000 (type: uint256)
  )

  DeltaHubStaking.Staked(
    user: <indexed> 0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b (type: address),
    amount: 3000000000000000000 (type: uint256)
  )

  IERC20.Transfer(
    from: <indexed> 0xE11BA2b4D45Eaed5996Cd0823791E0C93114882d (type: address),
    to: <indexed> 0xd99748782d7643b00C36a4Bb296C4A099dF98Ff3 (type: address),
    value: 5000000000000000000 (type: uint256)
  )
```

```
  IERC20.Approval(
    owner: <indexed> 0xE11BA2b4D45Eaed5996Cd0823791E0C93114882d (type: address),
    spender: <indexed> 0xd99748782d7643b00C36a4Bb296C4A099dF98Ff3 (type: address),
    value: 57896044618658097711785492504343953926634992332820282019723792003956564819968
(type: uint256)
  )

  IERC20.Transfer(
    from: <indexed> 0x0000000000000000000000000000000000000000 (type: address),
    to: <indexed> 0xE11BA2b4D45Eaed5996Cd0823791E0C93114882d (type: address),
    value: 5000000000000000000 (type: uint256)
  )

  DeltaHubStaking.Staked(
    user: <indexed> 0xE11BA2b4D45Eaed5996Cd0823791E0C93114882d (type: address),
    amount: 5000000000000000000 (type: uint256)
  )

  ----------------------------

  ✓ One staker on 2 durations with gap (2146ms)
  3) Notify Reward Amount from mocked distribution to 10,000

  Events emitted during test:
  ----------------------------

  IERC20.Transfer(
    from: <indexed> 0x0000000000000000000000000000000000000000 (type: address),
    to: <indexed> 0xACF05e2a9b757e70C2900432c9b4E08Fdc54dD53 (type: address),
    value: 100000000000000000000000000000000 (type: uint256)
  )

  IERC20.Transfer(
    from: <indexed> 0x0000000000000000000000000000000000000000 (type: address),
    to: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address),
    value: 100000000000000000000000000000000 (type: uint256)
  )
```

```
Ownable.OwnershipTransferred(
  previousOwner: <indexed> 0x0000000000000000000000000000000000000000 (type: address),
  newOwner: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address)
)

IERC20.Transfer(
  from: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address),
  to: <indexed> 0x85cd1F9550CdF662944bEc3E58021f8dbE6fE878 (type: address),
  value: 100000000000000000000000 (type: uint256)
)

IERC20.Transfer(
  from: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address),
  to: <indexed> 0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0 (type: address),
  value: 100000000000000000000 (type: uint256)
)

IERC20.Transfer(
  from: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address),
  to: <indexed> 0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b (type: address),
  value: 100000000000000000000 (type: uint256)
)

IERC20.Transfer(
  from: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address),
  to: <indexed> 0xE11BA2b4D45Eaed5996Cd0823791E0C93114882d (type: address),
  value: 100000000000000000000 (type: uint256)
)

IERC20.Transfer(
  from: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address),
  to: <indexed> 0xd03ea8624C8C5987235048901fB614fDcA89b117 (type: address),
  value: 100000000000000000000 (type: uint256)
)

IERC20.Transfer(
  from: <indexed> 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (type: address),
  to: <indexed> 0x95cED938F7991cd0dFcb48F0a06a40FA1aF46EBC (type: address),
```

```
    value: 1000000000000000000 (type: uint256)
  )

  IERC20.Approval(
    owner: <indexed> 0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0 (type: address),
    spender: <indexed> 0x85cd1F9550CdF662944bEc3E58021f8dbE6fE878 (type: address),
    value: 57896044618658097711785492504343953926634992332820282019728792003956564819968
(type: uint256)
  )

  IERC20.Approval(
    owner: <indexed> 0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b (type: address),
    spender: <indexed> 0x85cd1F9550CdF662944bEc3E58021f8dbE6fE878 (type: address),
    value: 57896044618658097711785492504343953926634992332820282019728792003956564819968
(type: uint256)
  )

  IERC20.Approval(
    owner: <indexed> 0xE11BA2b4D45Eaed5996Cd0823791E0C93114882d (type: address),
    spender: <indexed> 0x85cd1F9550CdF662944bEc3E58021f8dbE6fE878 (type: address),
    value: 57896044618658097711785492504343953926634992332820282019728792003956564819968
(type: uint256)
  )

  IERC20.Approval(
    owner: <indexed> 0xd03ea8624C8C5987235048901fB614fDcA89b117 (type: address),
    spender: <indexed> 0x85cd1F9550CdF662944bEc3E58021f8dbE6fE878 (type: address),
    value: 57896044618658097711785492504343953926634992332820282019728792003956564819968
(type: uint256)
  )

  DeltaHubStaking.RewardAdded(
    reward: 10000000000000000000 (type: uint256)
  )

  IERC20.Transfer(
    from: <indexed> 0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0 (type: address),
    to: <indexed> 0x85cd1F9550CdF662944bEc3E58021f8dbE6fE878 (type: address),
```

```
    value: 1000000000000000000 (type: uint256)
  )

  IERC20.Approval(
    owner: <indexed> 0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0 (type: address),
    spender: <indexed> 0x85cd1F9550CdF662944bEc3E58021f8dbE6fE878 (type: address),
    value: 57896044618658097711785492504343953926634992332820282019727792003956564819968
(type: uint256)
  )

  IERC20.Transfer(
    from: <indexed> 0x0000000000000000000000000000000000000000 (type: address),
    to: <indexed> 0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0 (type: address),
    value: 1000000000000000000 (type: uint256)
  )

  DeltaHubStaking.Staked(
    user: <indexed> 0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0 (type: address),
    amount: 1000000000000000000 (type: uint256)
  )

  IERC20.Transfer(
    from: <indexed> 0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b (type: address),
    to: <indexed> 0x85cd1F9550CdF662944bEc3E58021f8dbE6fE878 (type: address),
    value: 3000000000000000000 (type: uint256)
  )

  IERC20.Approval(
    owner: <indexed> 0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b (type: address),
    spender: <indexed> 0x85cd1F9550CdF662944bEc3E58021f8dbE6fE878 (type: address),
    value: 57896044618658097711785492504343953926634992332820282019725792003956564819968
(type: uint256)
  )

  IERC20.Transfer(
    from: <indexed> 0x0000000000000000000000000000000000000000 (type: address),
    to: <indexed> 0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b (type: address),
    value: 3000000000000000000 (type: uint256)
```

```
  )

  DeltaHubStaking.Staked(
    user: <indexed> 0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b (type: address),
    amount: 3000000000000000000 (type: uint256)
  )


  ---------------------------


9 passing (1m)
3 failing

1) Contract: DeltaHubStaking
     Two stakers with the different (1:3) stakes wait 1 w:

    AssertionError: expected '119047619047619047' to be less than or equal to '0'
    + expected - actual

    -119047619047619047
    +0

    at Context.<anonymous> (test\DeltaHubStaking.test.ts:192:38)
    at runMicrotasks (<anonymous>)
    at processTicksAndRejections (node:internal/process/task_queues:93:5)

2) Contract: DeltaHubStaking
     Three stakers with the different (1:3:5) stakes wait 3 weeks:

    AssertionError: expected '18000089285714285620685' to be less than or equal to
'180000000000000000000'
    + expected - actual

    -18000089285714285620685
    +180000000000000000000

    at Context.<anonymous> (test\DeltaHubStaking.test.ts:285:38)
```

```
    at runMicrotasks (<anonymous>)
    at processTicksAndRejections (node:internal/process/task_queues:93:5)

  3) Contract: DeltaHubStaking
       Notify Reward Amount from mocked distribution to 10,000:

     AssertionError: expected '16534391534391534' to be less than or equal to '0'
     + expected - actual

     -16534391534391534
     +0

     at Context.<anonymous> (test\DeltaHubStaking.test.ts:389:38)
     at runMicrotasks (<anonymous>)
     at processTicksAndRejections (node:internal/process/task_queues:93:5)



error Command failed with exit code 3.
info Visit https://yarnpkg.com/en/docs/cli/run for documentation about this command.
```

Code Coverage

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts/	88.10	64.29	85.71	88.64	
DeltaHubStaking.sol	88.10	64.29	85.71	88.64	... 127, 142, 146
contracts/mock	100.00	100.00	100.00	100.00	
DeltaHubMock.sol	100.00	100.00	100.00	100.00	
All files	**88.64**	**64.29**	**86.67**	**89.13**	

```
> Istanbul reports written to ./coverage/ and ./coverage.json
> solidity-coverage cleaning up, shutting down ganache server
Error: ❌ 3 test(s) failed under coverage.
    at plugin
(D:\Projects\Audit\Deltahub\deltahub-contracts\node_modules\solidity-coverage\plugins\truffle.plugin.js:
121:27)
    at runMicrotasks (<anonymous>)
    at processTicksAndRejections (node:internal/process/task_queues:93:5)
Truffle v5.1.55 (core: 5.1.55)
Node v15.4.0
error Command failed with exit code 1.
info Visit https://yarnpkg.com/en/docs/cli/run for documentation about this command.
```

Conclusion

Code coverage according to Industry standards should not be less than 95%.

Tests Written by Zokyo

Execution Report

Unit tests

```
yarn run v1.22.5
$ truffle test
Compiling your contracts...
===========================
> Compiling @openzeppelin/contracts/access/Ownable.sol
> Compiling @openzeppelin/contracts/math/Math.sol
> Compiling @openzeppelin/contracts/math/SafeMath.sol
> Compiling @openzeppelin/contracts/token/ERC20/ERC20.sol
> Compiling @openzeppelin/contracts/token/ERC20/IERC20.sol
> Compiling @openzeppelin/contracts/token/ERC20/SafeERC20.sol
> Compiling @openzeppelin\contracts\GSN\Context.sol
> Compiling @openzeppelin\contracts\math\SafeMath.sol
> Compiling @openzeppelin\contracts\token\ERC20\IERC20.sol
> Compiling @openzeppelin\contracts\utils\Address.sol
> Compiling .\contracts\DeltaHubStaking.sol
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\mock\DeltaHubMock.sol
> Compilation warnings encountered:

    @openzeppelin/contracts/access/Ownable.sol:26:5: Warning: Visibility for constructor is ignored. If you
want the contract to be non-deployable,
making it "abstract" is sufficient.
    constructor () internal {
    ^ (Relevant source part starts here and spans across multiple lines).
,@openzeppelin/contracts/token/ERC20/ERC20.sol:55:5: Warning: Visibility for constructor is ignored. If
you want the contract to be non-deployable,
making it "abstract" is sufficient.
    constructor (string memory name_, string memory symbol_) public {
    ^ (Relevant source part starts here and spans across multiple lines).
,/D/Projects/Audit/Deltahub/deltahub-contracts/contracts/DeltaHubStaking.sol:54:5: Warning: Visibility
for constructor is ignored. If you want the contract to be non-deployable, making it "abstract" is
sufficient.
```

```
    constructor(IERC20 _dhc) public {
    ^ (Relevant source part starts here and spans across multiple lines).
,/D/Projects/Audit/Deltahub/deltahub-contracts/contracts/Migrations.sol:13:5: Warning: Visibility for
constructor is ignored. If you want the contract to be non-deployable, making it "abstract" is sufficient.
    constructor() public {
    ^ (Relevant source part starts here and spans across multiple lines).
,/D/Projects/Audit/Deltahub/deltahub-contracts/contracts/mock/DeltaHubMock.sol:8:5: Warning: Visibility
for constructor is ignored. If you want the
contract to be non-deployable, making it "abstract" is sufficient.
    constructor() public ERC20("DelataHub Capital", "DHC") {
    ^ (Relevant source part starts here and spans across multiple lines).

> Artifacts written to C:\Users\roman\AppData\Local\Temp\test--19388-PWJrnnOhtoQY
> Compiled successfully using:
   - solc: 0.7.5+commit.eb77ed08.Emscripten.clang


  Contract: DeltaHubStaking
    check basic init
      ✓ has a name (76ms)
      ✓ has a symbol (61ms)
      ✓ has 18 decimals (88ms)
      ✓ has total supply (248ms)
      constructor
        ✓ should set token address (149ms)
    Functions
      lastTimeRewardApplicable
        ✓ should return zero (285ms)
        ✓ should return ~current timestamp (652ms)
        ✓ should return periodFinish (1404ms)
      rewardPerToken
        ✓ should return zero if no one stake (230ms)
        ✓ should return reward per token (1758ms)
        ✓ should calculate reword per token correctly (4192ms)
        ✓ should calculate reword per token correctly with small amount of tokens (2144ms)
      earned
        should calculate reward correctly
          ✓ when no stakes (118ms)
```

```
      ✓ when no reward amount (1408ms)
      ✓ when stake small amount (3343ms)
      ✓ when stake big amount (1798ms)
      ✓ when stake different amount a few times (5816ms)
      ✓ when stake and unstake (2727ms)
  stake
    ✓ should revert if amount == 0 (496ms)
    ✓ should increase _totalStaked (1594ms)
    ✓ should add stake amount to _staked (2515ms)
    ✓ should transfer stake amount to contract (2897ms)
    ✓ should mint tokens (3330ms)
    ✓ should cath event Staked (340ms)
  unstake
    ✓ should revert if amount == 0 (600ms)
    ✓ should fail if unstake > staked (1635ms)
    ✓ should decrease _totalStaked (2021ms)
    ✓ should subtract unstake amount from _staked (3796ms)
    ✓ should transfer unstake amount to member (2171ms)
    ✓ should burn tokens (4015ms)
    ✓ should cath event Unstaked (929ms)
  exit
    ✓ should unstake all tokens (2020ms)
    ✓ should got reward (7032ms)
  getReward
    ✓ should pass if reward == 0 (560ms)
    ✓ should zeroed rewards (5837ms)
    ✓ should transfer reward (6249ms)
    ✓ should cath event RewardPaid (1657ms)
  notifyRewardAmount
    ✓ only Reward Distribution can set reward (349ms)
    ✓ should update rewardRate (2456ms)
    ✓ should update lastUpdateTime (1579ms)
    ✓ should update periodFinish (1870ms)
    ✓ should catch event RewardAdded (373ms)
  setRewardDistribution
    ✓ only owner can set Reward Distribution (368ms)
    ✓ should set Reward Distribution (968ms)
  totalStaked
```

```
    ✓ should return total staked amount (1889ms)
  totalStakedFor
    ✓ should return total staked for specific account (2407ms)

46 passing (5m)

Done in 335.96s.
```

Integrations tests

```
Compiling your contracts...
===========================
> Compiling @openzeppelin/contracts/access/Ownable.sol
> Compiling @openzeppelin/contracts/math/Math.sol
> Compiling @openzeppelin/contracts/math/SafeMath.sol
> Compiling @openzeppelin/contracts/token/ERC20/ERC20.sol
> Compiling @openzeppelin/contracts/token/ERC20/IERC20.sol
> Compiling @openzeppelin/contracts/token/ERC20/SafeERC20.sol
> Compiling @openzeppelin\contracts\GSN\Context.sol
> Compiling @openzeppelin\contracts\math\SafeMath.sol
> Compiling @openzeppelin\contracts\token\ERC20\IERC20.sol
> Compiling @openzeppelin\contracts\utils\Address.sol
> Compiling .\contracts\DeltaHubStaking.sol
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\mock\DeltaHubMock.sol
> Compilation warnings encountered:

   @openzeppelin/contracts/access/Ownable.sol:26:5: Warning: Visibility for constructor is ignored. If you
want the contract to be non-deployable, making it "abstract" is sufficient.
   constructor () internal {
   ^ (Relevant source part starts here and spans across multiple lines).
,@openzeppelin/contracts/token/ERC20/ERC20.sol:55:5: Warning: Visibility for constructor is ignored. If
you want the contract to be non-deployable, making it "abstract" is sufficient.
   constructor (string memory name_, string memory symbol_) public {
   ^ (Relevant source part starts here and spans across multiple lines).
```

```
,/D/Projects/Audit/Deltahub/deltahub-contracts/contracts/DeltaHubStaking.sol:54:5: Warning: Visibility
for constructor is ignored. If you want the contract to be non-deployable, making it "abstract" is
sufficient.
    constructor(IERC20 _dhc) public {
    ^ (Relevant source part starts here and spans across multiple lines).
,/D/Projects/Audit/Deltahub/deltahub-contracts/contracts/Migrations.sol:13:5: Warning: Visibility for
constructor is ignored. If you want the contract to be non-deployable, making it "abstract" is sufficient.
    constructor() public {
    ^ (Relevant source part starts here and spans across multiple lines).
,/D/Projects/Audit/Deltahub/deltahub-contracts/contracts/mock/DeltaHubMock.sol:8:5: Warning: Visibility
for constructor is ignored. If you want the contract to be non-deployable, making it "abstract" is
sufficient.
    constructor() public ERC20("DelataHub Capital", "DHC") {
    ^ (Relevant source part starts here and spans across multiple lines).

> Artifacts written to C:\Users\roman\AppData\Local\Temp\test--26244-8GYMtlBsyDk8
> Compiled successfully using:
   - solc: 0.7.5+commit.eb77ed08.Emscripten.clang

Contract: DeltaHubStaking
    Integrations test case:
      ✓ set reward a few times (4032ms)
      ✓ check if it calc correctly reward per day (3340ms)
      ✓ check if it calc correctly reward per day when set diff reward (4757ms)
      ✓ 5 users stake same amount on diff period (14819ms)
      ✓ 5 users stake diff amount on diff period with diff reward (17828ms)
      ✓ when stake small and big amount (5111ms)
      ✓ when stake and unstake small and big amount (5464ms)

7 passing (1m)
```

Code Coverage

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts/	100.00	100.00	100.00	100.00	
DeltaHubStaking.sol	100.00	100.00	100.00	100.00	
contracts/mock	100.00	100.00	100.00	100.00	
DeltaHubMock.sol	100.00	100.00	100.00	100.00	
All files	**100.00**	**100.00**	**100.00**	**100.00**	

```
> Istanbul reports written to ./coverage/ and ./coverage.json
> solidity-coverage cleaning up, shutting down ganache server
Done in 335.88s.
```

We are grateful to have been given the opportunity to work with the DeltaHub team.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

Zokyo's Security Team recommends that the DeltaHub team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.