



CrowdSwap

Staking

SMART CONTRACT AUDIT

09.06.2022

Made in Germany by Chainsulting.de



Table of contents

| | |
|-----------------------------------------------------------|----|
| 1. Disclaimer | 4 |
| 2. About the Project and Company | 5 |
| 2.1 Project Overview..... | 6 |
| 3. Vulnerability & Risk Level..... | 8 |
| 4. Auditing Strategy and Techniques Applied..... | 9 |
| 4.1 Methodology..... | 9 |
| 5. Metrics | 10 |
| 5.1 Tested Contract Files..... | 10 |
| 5.2 Used Code from other Frameworks/Smart Contracts | 11 |
| 5.3 CallGraph..... | 12 |
| 5.4 Inheritance Graph..... | 13 |
| 5.5 Source Lines & Risk..... | 14 |
| 5.6 Capabilities | 15 |
| 5.7 Source Unites in Scope..... | 16 |
| 6. Scope of Work..... | 17 |
| 6.1 Findings Overview..... | 18 |
| 6.2 Manual and Automated Vulnerability Test..... | 19 |
| CRITICAL ISSUES | 19 |
| HIGH ISSUES | 19 |
| 6.2.1 Initialize Not Protected | 19 |
| MEDIUM ISSUES | 19 |
| 6.2.2 Mathematical Units | 20 |



| | |
|-------------------------------------|----|
| LOW ISSUES | 20 |
| 6.2.3 Missing Events | 21 |
| 6.2.4 Ownership Control | 21 |
| INFORMATIONAL ISSUES | 22 |
| 6.2.5 Missing Test Cases | 22 |
| 6.2.6 Unexplicit Uint Types | 23 |
| 6.2.7 Floating Compiler Version | 23 |
| 6.2.8 Missing NatSpec Documentation | 24 |
| 6.3 SWC Attacks | 25 |
| 7. Test Deployment | 29 |
| 8. Verify Claims | 34 |
| 9. Executive Summary | 35 |
| 10. Deployed Smart Contract | 35 |
| 11. About the Auditor | 36 |

1. Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Smart Chains GmbH. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

| Major Versions / Date | Description |
|-----------------------|-------------------------------------------------------|
| 0.1 (11.03.2022) | Layout |
| 0.4 (13.03.2022) | Automated Security Testing Manual Security Testing |
| 0.5 (14.03.2022) | Verify Claims and Test Deployment |
| 0.6 (15.03.2022) | Testing SWC Checks |
| 0.9 (16.03.2022) | Summary and Recommendation |
| 1.0 (17.03.2022) | Final document |
| 1.1 (09.06.2022) | Re-check |
| 1.2 (09.06.2022) | Deployed contracts |

2. About the Project and Company

Smart Chains GmbH
Weidenstr. 20
53562 St. Katharinen
Germany



Website: <https://crowdswap.org/>

Twitter: https://twitter.com/CrowdSwap_App

Discord: <https://discord.com/invite/ugZa8Q5nmb>

LinkedIn: <https://www.linkedin.com/company/crowdswap-crypto/>

Telegram: <https://t.me/crowdswap>

Blog: <https://crowdswap.org/blog1/>

2.1 Project Overview

Crowdstake is part of the CrowdSwap ecosystem and allows users to stake crowd tokens in return for a share of profit from fees and decision making in voting. Stakers receive rewards for providing liquidity and holding the tokens.

Search

Search finds the best prices for your token pair across the DeFi Space including all ancillary costs. Network boundaries are no longer a barrier anymore. If the price is better on another network, CrowdSwap will guide you through cross-chain swaps with ease. The best price routing algorithm includes aggregation of liquidity sources on any search. If there is a better routing option available you will find it in the result ranking.

Swap

The standard known token swap features several swap possibilities that will optimize your token trades. You can choose between direct or bulk swap or choose from a variety of predefined strategies to e.g. increase your holdings of a token. Taking the information from the search results and the transparency wallet network crossing is also transparent for you and done by the swap itself. For better understanding see the whitepaper and have a look at the cross-chain liquidity protocol (CCLP)

Cross-chain liquidity protocol (CCLP)

CrowdSwap provides different kinds of token transfers between chains, all of which are transparent for the user itself. From a technical point of view, we use our own cross-chain solution that is based on cross-chain liquidity pools to exchange the tokens naturally. No need for pegged tokens, burning and minting anymore. CCLP will give the basis for awesome swapping experiences that can lead to significant gains and fee reduction for the ethereum network. The optimized liquidity pools of CCLP will make traders and liquidity providers excited to get their hands on. The CCLPs reduces fees for the trader while increasing yield for the liquidity provider at the same time.

Wallet Manager - one Wallet for all

Wallets are great and necessary tools when it comes to DeFi and swaps. Given the rising of DeFi to a multi chain universe, wallets have to add features for multi chain transparency. Our wallet manager simplifies the complexity of having different wallets for each network. Leaving the security at the wallet side, the wallet manager gathers important information for search and best price routing based on your tokens across all connected wallets. You don't have to choose a network before executing your best price search. Search will find the best price for your desired token, providing you with even better opportunities regardless of the network.



Best Price Routing (BPR)

Best Price Routing is the algorithm that is responsible for executing the swap over integrated networks and DEXes. Furthermore it provides the best prices for search, too. Based on on-chain data analysis, BPR calculates all ancillary costs for the swaps and returns an overall price per token. We analyze the smart contract and investigate on-chain data intensively to provide the best routings. We found out that routing paths were chosen by liquidity pool healthiness over best prices for the user. Sometimes DEX aggregators push liquidity pools over other factors such as price. With CrowdSwap this will not be the case anymore! The BPR works as an aggregator by nature. If there is a better route for the given token pair, this will be included into the search result.

On-chain data analysis

On-chain data analysis is a big advantage to optimize trading. Not only can we reduce the collateral costs of swaps, we can furthermore use it to provide automated features like trading strategies and better user experience. On-chain data is already used in the BPR to find the best price based on token price, transaction and transfer costs and fees. Aggregated on-chain data is the key to take action before everybody else does. We will provide additional premium services based on insight of on-chain data.



3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

| Level | Value | Vulnerability | Risk (Required Action) |
|---------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------|
| Critical | 9 – 10 | A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken. | Immediate action to reduce risk level. |
| High | 7 – 8.9 | A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way. | Implementation of corrective actions as soon as possible. |
| Medium | 4 – 6.9 | A vulnerability that could affect the desired outcome of executing the contract in a specific scenario. | Implementation of corrective actions in a certain period. |
| Low | 2 – 3.9 | A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective. | Implementation of certain corrective actions or accepting the risk. |
| Informational | 0 – 1.9 | A vulnerability that have informational character but is not effecting any of the code. | An observation that does not determine a level of risk |

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
 - i. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the smart contract.
 - ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 - iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Chainsulting describe.
2. Testing and automated analysis that includes the following:
 - i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
 - ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.



5. Metrics

The metrics section should give the reader an overview on the size, quality, flows and capabilities of the codebase, without the knowledge to understand the actual code.

5.1 Tested Contract Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

Source: <https://github.com/CrowdSwap/CrowdStaking>

| File | Fingerprint (MD5) |
|------------------------|----------------------------------|
| DateTimeUtil.sol | c52853256545a50de60c6b00cdc29e5d |
| OwnableUpgradeable.sol | 7658a9067bab832fcb12f275b6c839ea |
| StakingRewards.sol | e95232893774ccb74681eb716a26897c |

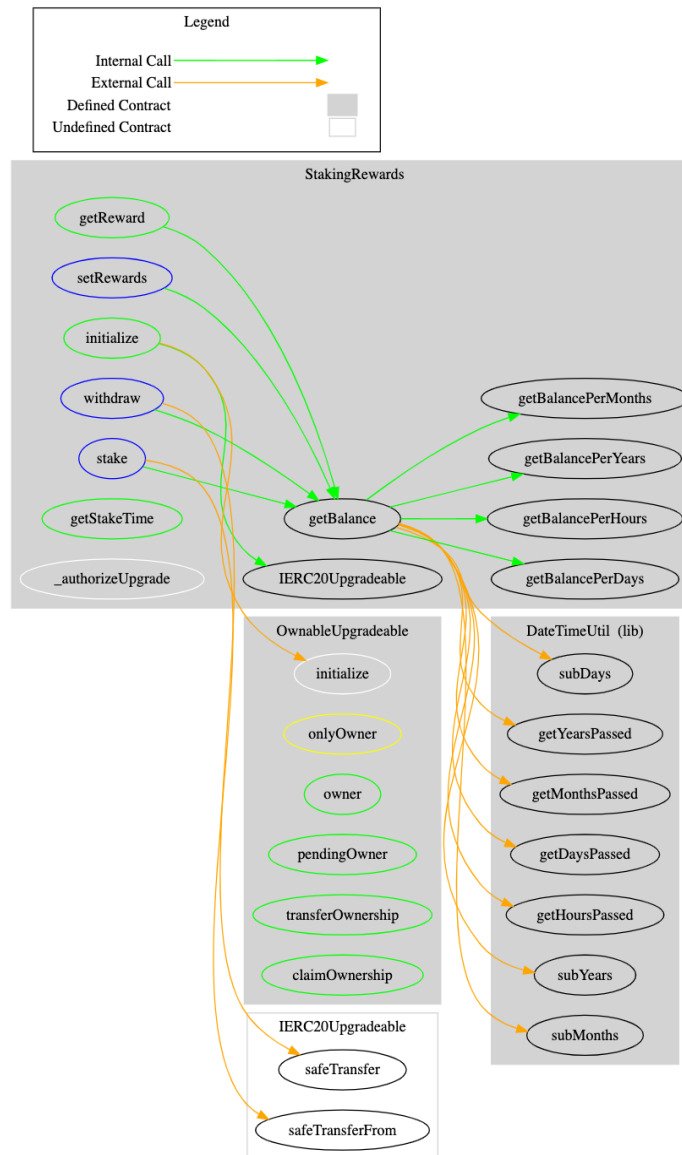
Updated (09.06.2022)

| File | Fingerprint (MD5) |
|------------------------|----------------------------------|
| DateTimeUtil.sol | 3ac08348d3f89b2a4a86ba18c47580c7 |
| OwnableUpgradeable.sol | 4e64e809e64c5b3a98889a57b780017a |
| StakingRewards.sol | 8bf58d0067cc0693ba0b4c15efdf215f |

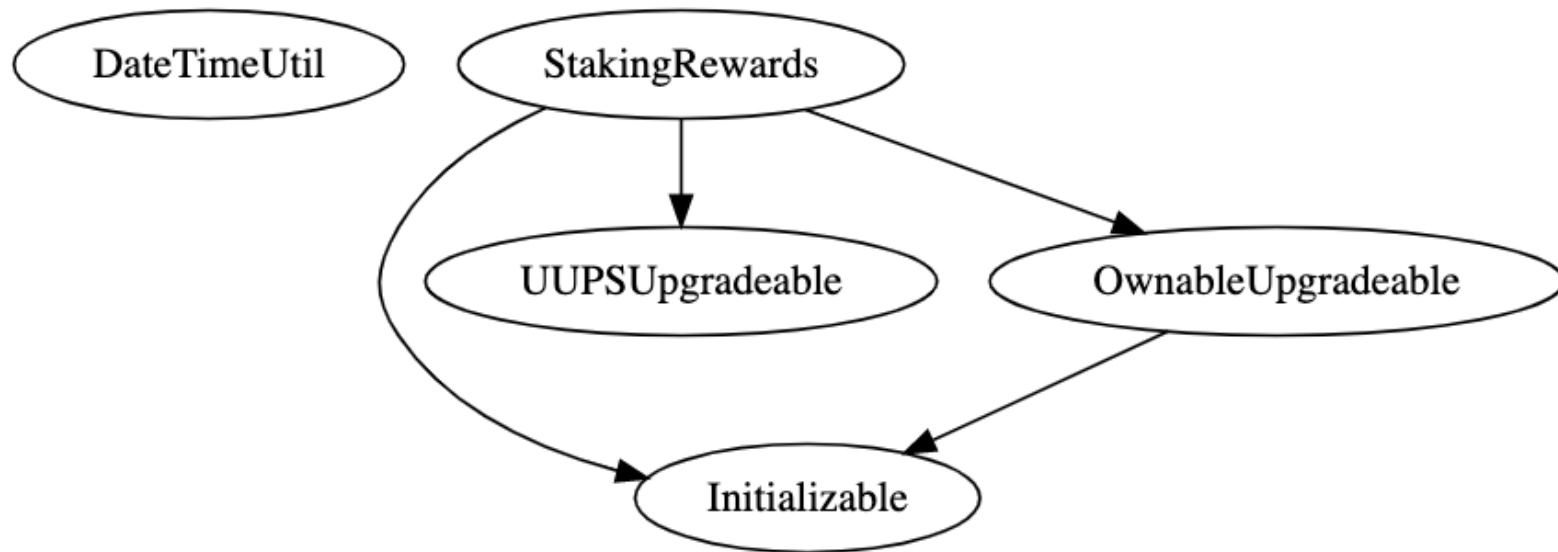
5.2 Used Code from other Frameworks/Smart Contracts (direct imports)

| Dependency / Import Path | Source |
|--------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| @openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol | https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/release-v4.5/contracts/proxy/utils/Initializable.sol |
| @openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol | https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/release-v4.5/contracts/proxy/utils/UUPSUpgradeable.sol |
| @openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol | https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/release-v4.5/contracts/token/ERC20/IERC20Upgradeable.sol |
| @openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol | https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/release-v4.5/contracts/token/ERC20/utils/SafeERC20Upgradeable.sol |

5.3 CallGraph

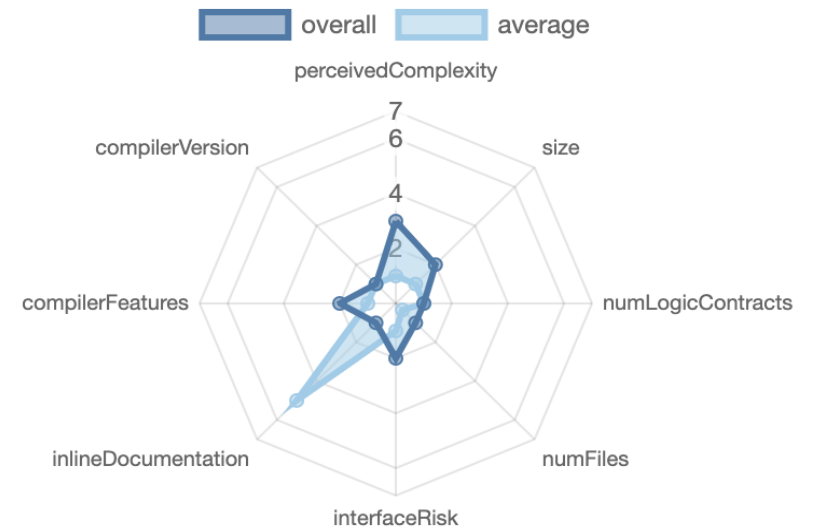
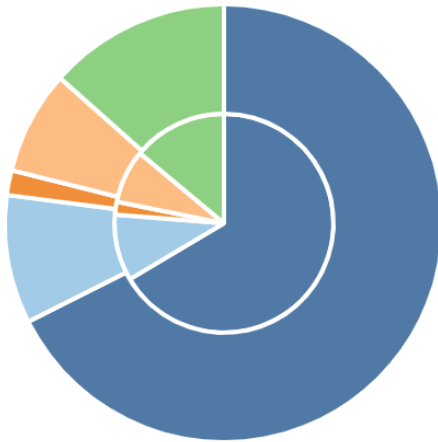


5.4 Inheritance Graph














5.5 Source Lines & Risk

source comment single block mixed
empty todo blockEmpty





5.6 Capabilities

|  Solidity Versions observed |  Experimental Features |  Can Receive Funds |  Uses Assembly |  Has Destroyable Contracts |
|--------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------|
| 0.8.10 | | | | |

|  Transfers ETH |  Low-Level Calls |  DelegateCall |  Uses Hash Functions |  ECREcover |  New/Create/Create2 |
|-------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|
| | | | | | |


Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.





|  Public |  Payable |
|------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| 11 | 0 |

| External | Internal | Private | Pure | View |
|----------|----------|---------|------|------|
| 3 | 17 | 4 | 11 | 5 |

StateVariables

| Total |  Public |
|-------|--------------------------------------------------------------------------------------------|
| 13 | 7 |

5.7 Source Unites in Scope

| Type | File | Logic Contracts | Interfaces | Lines | nLines | nSLOC | Comment Lines | Complex. Score | Capabilities |
|-----------------------------------------------------------------------------------|----------------------------------|-----------------|------------|------------|------------|------------|---------------|----------------|--------------|
|  | contracts/DateTimeUtil.sol | 1 | _____ | 41 | 41 | 35 | 1 | 12 | _____ |
|  | contracts/StakingRewards.sol | 1 | _____ | 189 | 178 | 143 | 4 | 106 | _____ |
|  | contracts/OwnableUpgradeable.sol | 1 | _____ | 66 | 66 | 32 | 26 | 19 | _____ |
|  | Totals | 3 | _____ | 296 | 285 | 210 | 31 | 137 | _____ |

Legend: []

- **Lines:** total lines of the source unit
- **nLines:** normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC:** normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines:** lines containing single or block comments
- **Complexity Score:** a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

6. Scope of Work

The CrowdSwap Team provided us with the files that needed to be tested. The scope of the audit is the staking contract.

Following contracts with the direct imports has been tested:

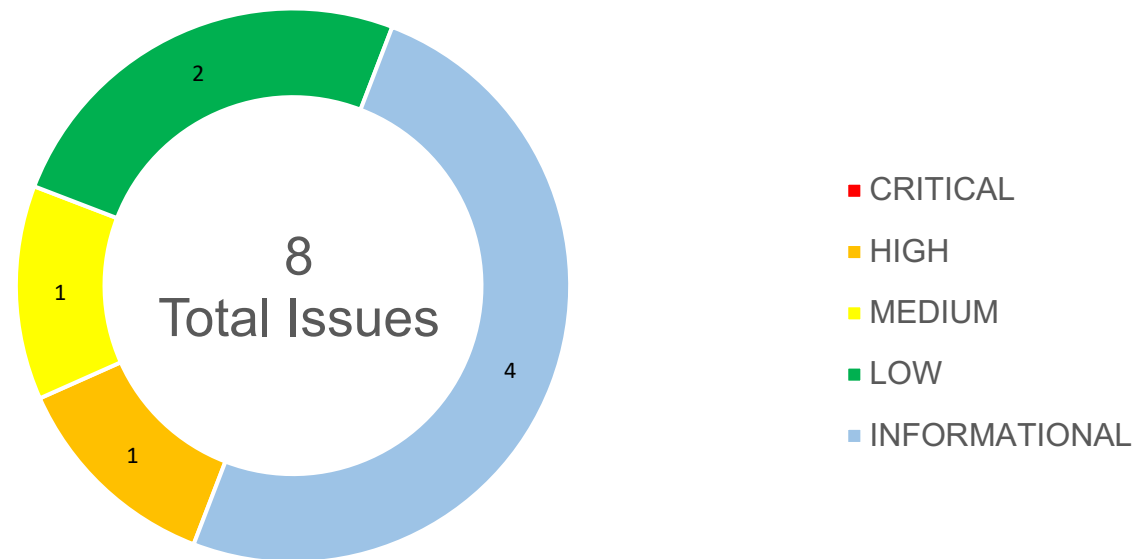
- StakingRewards.sol

The team put forward the following assumptions regarding the security, usage of the contracts:

- Staking rewards are unfair as rewards are rounded down e.g reward 1 hour given, if stake again for 50 minutes, rewards are lost
- The user can withdraw the stake at any time and receiving the correct reward
- Owner/Deployer is not able to withdraw staked token from user
- Owner/Deployer cannot pause the staking
- Only the Owner is able to set rewards
- Rewards send to the contract cannot be withdrawn by the Owner
- The smart contract is coded according to the newest standards and in a secure way.

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.

6.1 Findings Overview



| No | Title | Severity | Status |
|-------|-------------------------------|---------------|--------|
| 6.2.1 | Initialize Not Protected | HIGH | FIXED |
| 6.2.2 | Mathematical Units | MEDIUM | FIXED |
| 6.2.3 | Missing Events | LOW | FIXED |
| 6.2.4 | Ownership Control | LOW | FIXED |
| 6.2.5 | Missing Test Cases | INFORMATIONAL | FIXED |
| 6.2.6 | Unexplicit Uint Types | INFORMATIONAL | FIXED |
| 6.2.7 | Floating Compiler Version | INFORMATIONAL | FIXED |
| 6.2.8 | Missing NatSpec Documentation | INFORMATIONAL | FIXED |



6.2 Manual and Automated Vulnerability Test

CRITICAL ISSUES

During the audit, Chainsulting's experts found **0 Critical issues** in the code of the smart contract.

HIGH ISSUES

During the audit, Chainsulting's experts found **1 High issue** in the code of the smart contract.

6.2.1 Initialize Not Protected

Severity: HIGH

Status: FIXED

Code: NA

File(s) affected: StakingRewards.sol

Update: The StakingRewards contract must only be deployed as a Proxy (We are using Openzeppelin upgrades plugin).

| Attack / Description | Code Snippet | Result/Recommendation |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|
| StakingRewards.sol is an upgradeable contract that uses an initializer. The initialize function is an unprotected external function. Anyone can call it before the owner or caller with right intentions; and pass in parameters like stakingToken and rewards. | StakingRewards.sol (Line 45) <pre>function initialize(address _stakingToken, uint _rewardPerYear, uint _rewardPerMonth, uint _rewardPerDay, uint _rewardPerHour) public initializer {</pre> | It is recommended to protect the function using access control like onlyOwner modifier. |

MEDIUM ISSUES

During the audit, Chainsulting's experts found **1 Medium issue** in the code of the smart contract



6.2.2 Mathematical Units

Severity: MEDIUM

Status: FIXED

Code: NA

File(s) affected: DateTimeUtil.sol

Update: <https://github.com/CrowdSwap/CrowdStaking/commit/9d9ea98d6a6de87dbbb7f6ee58237c5097f9c424>

| Attack / Description | Code Snippet | Result/Recommendation |
|--------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| The current implementation has a calculation for time units. | <pre>DateTimeUtil.sol (line 6 - line 9) uint constant SECONDS_PER_YEAR = 365 * 24 * 60 * 60; uint constant SECONDS_PER_MONTH = 30 * 24 * 60 * 60; uint constant SECONDS_PER_DAY = 24 * 60 * 60; uint constant SECONDS_PER_HOUR = 60 * 60;</pre> | <p>It is highly recommended to use solidity integrated time units to increase code readability and avoid calculation error.</p> <pre>uint constant SECONDS_PER_HOUR = 60 * 1 seconds;</pre> <p>https://docs.soliditylang.org/en/v0.8.10/units-and-global-variables.html#time-units</p> |

LOW ISSUES

During the audit, Chainsulting's experts found **2 Low issues** in the code of the smart contract



6.2.3 Missing Events

Severity: LOW

Status: FIXED

Code: NA

File(s) affected: StakingRewards.sol

Update: <https://github.com/CrowdSwap/CrowdStaking/commit/9d9ea98d6a6de87dbbb7f6ee58237c5097f9c424>

| Attack / Description | Code Snippet | Result/Recommendation |
|------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Critical functions which are important to track or alert on change, doesn't have events. | <pre>StakingRewards.sol (line 95) function setRewards()... StakingRewards.sol (line 32) function initialize() ...</pre> | <p>It is recommended to emit events for these critical functions, to allow off-chain tracking, monitoring, logging and alerting.</p> <pre>emit RewardsSet(param1, param2, ...) etc</pre> |

6.2.4 Ownership Control

Severity: LOW

Status: FIXED

Code: CWE-282

File(s) affected: StakingRewards.sol

Update: The stakingReward contract's owner is a Multisig wallet. The two-step process for transferring ownership is already implemented by using transferOwnership and claimOwnership functions in OwnableUpgradeable contract.

| Attack / Description | Code Snippet | Result/Recommendation |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| StakingRewards.sol contract is Ownable. While it allows setting of restricted aspects and only owner aspects like upgrade. It centralizes power in one address. Owner can | Inherits OwnableUpgradeable to have an owner | <p>It is recommended to use Multisig for ownership access such as Gnosis Safe.</p> <p>It is recommended to use a two-step process when transferring ownership, to ensure the new owner can confirm, has access and control to the new Owner</p> |



| | | |
|-------------------------------------------------------------------|--|-----------------------------------------------------------|
| transfer ownership to invalid address or can access can get lost. | | address. That avoids loss of ownership over the contract. |
|-------------------------------------------------------------------|--|-----------------------------------------------------------|

INFORMATIONAL ISSUES

During the audit, Chainsulting's experts found **4 informational issue** in the code of the smart contract

6.2.5 Missing Test Cases

Severity: INFORMATIONAL

Status: FIXED

Code: NA

File(s) affected: ALL

Update: Test cases are pushed to the github repo. We don't think any audit process is necessary for test cases, just added them as information.

| Attack / Description | Code Snippet | Result/Recommendation |
|-----------------------------------------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| The current implementation has no unit-tests. | NA | It is highly recommended to write test cases for every function and control the behaviour at any time of contract executions. It is necessary to check the desired functionality of executed code. Tests are also a good proof that the written code is working in the intended way. |

6.2.6 Unexplicit Uint Types

Severity: INFORMATIONAL

Status: FIXED

Code: NA

File(s) affected: DatTimeUtil.sol and StakingRewards.sol

Update: <https://github.com/CrowdSwap/CrowdStaking/commit/9d9ea98d6a6de87dbbb7f6ee58237c5097f9c424>

| Attack / Description | Code Snippet | Result/Recommendation |
|--------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| In the current implementation several unsigned integer variables have an unexplicit type (uint). | <code>DatTimeUtil.sol (line 6 - line 9)</code> <code>StakingRewards.sol</code> (line 16 - line 27) (line 32 - line 37) line 46, line 66, lin (162-183) etc | It is recommended to use explicit variable declarations such as uint8 or uint256. It improves code readability and can help to make sure variables have an intended size. |

6.2.7 Floating Compiler Version

Severity: INFORMATIONAL

Status: FIXED

Code: SWC-103

File(s) affected: ALL

Update: <https://github.com/CrowdSwap/CrowdStaking/commit/9d9ea98d6a6de87dbbb7f6ee58237c5097f9c424>

| Attack / Description | Code Snippet | Result/Recommendation |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| The current pragma solidity directive is floating. It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is | <code>pragma solidity ^0.8.10</code> | It is recommended to follow the latter example, as future compiler versions may handle certain language constructions in a way the developer did not foresee. i.e. Pragma solidity 0.8.10 See SWC-103: https://swcregistry.io/docs/SWC-103 |



| | | |
|------------------------------------------------------------------------------|--|--|
| especially important if you rely on bytecode-level verification of the code. | | |
|------------------------------------------------------------------------------|--|--|

6.2.8 Missing NatSpec Documentation

Severity: INFORMATIONAL

Status: FIXED

Code: NA

File(s) affected: ALL

Update: <https://github.com/CrowdSwap/CrowdStaking/commit/9d9ea98d6a6de87dbbb7f6ee58237c5097f9c424>

| Attack / Description | Code Snippet | Result/Recommendation |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Solidity contracts can use a special form of comments to provide rich documentation for function, return variables, and more. This special form is named Ethereum Natural Language Specification Format(NatSpec). | N/A | It is recommended to include natspec documentation and follow the doxygen style including @author, @title, @notice, @dev, @param, @return and make it easier to review and understand your smart contract. |

6.3 SWC Attacks

| ID | Title | Relationships | Test Result |
|-------------------------|---------------------------------------------------|----------------------------------------------------------------------------------------|-------------|
| SWC-131 | Presence of unused variables | CWE-1164: Irrelevant Code | ✓ |
| SWC-130 | Right-To-Left-Override control character (U+202E) | CWE-451: User Interface (UI) Misrepresentation of Critical Information | ✓ |
| SWC-129 | Typographical Error | CWE-480: Use of Incorrect Operator | ✓ |
| SWC-128 | DoS With Block Gas Limit | CWE-400: Uncontrolled Resource Consumption | ✓ |
| SWC-127 | Arbitrary Jump with Function Type Variable | CWE-695: Use of Low-Level Functionality | ✓ |
| SWC-125 | Incorrect Inheritance Order | CWE-696: Incorrect Behavior Order | ✓ |
| SWC-124 | Write to Arbitrary Storage Location | CWE-123: Write-what-where Condition | ✓ |
| SWC-123 | Requirement Violation | CWE-573: Improper Following of Specification by Caller | ✓ |

| ID | Title | Relationships | Test Result |
|-------------------------|-----------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|-------------|
| SWC-122 | Lack of Proper Signature Verification | CWE-345: Insufficient Verification of Data Authenticity | ✓ |
| SWC-121 | Missing Protection against Signature Replay Attacks | CWE-347: Improper Verification of Cryptographic Signature | ✓ |
| SWC-120 | Weak Sources of Randomness from Chain Attributes | CWE-330: Use of Insufficiently Random Values | ✓ |
| SWC-119 | Shadowing State Variables | CWE-710: Improper Adherence to Coding Standards | ✓ |
| SWC-118 | Incorrect Constructor Name | CWE-665: Improper Initialization | ✓ |
| SWC-117 | Signature Malleability | CWE-347: Improper Verification of Cryptographic Signature | ✓ |
| SWC-116 | Timestamp Dependence | CWE-829: Inclusion of Functionality from Untrusted Control Sphere | ✓ |
| SWC-115 | Authorization through tx.origin | CWE-477: Use of Obsolete Function | ✓ |
| SWC-114 | Transaction Order Dependence | CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition') | ✓ |

| ID | Title | Relationships | Test Result |
|-------------------------|--------------------------------------|-----------------------------------------------------------------------------------|-------------|
| SWC-113 | DoS with Failed Call | CWE-703: Improper Check or Handling of Exceptional Conditions | ✓ |
| SWC-112 | Delegatecall to Untrusted Callee | CWE-829: Inclusion of Functionality from Untrusted Control Sphere | ✓ |
| SWC-111 | Use of Deprecated Solidity Functions | CWE-477: Use of Obsolete Function | ✓ |
| SWC-110 | Assert Violation | CWE-670: Always-Incorrect Control Flow Implementation | ✓ |
| SWC-109 | Uninitialized Storage Pointer | CWE-824: Access of Uninitialized Pointer | ✓ |
| SWC-108 | State Variable Default Visibility | CWE-710: Improper Adherence to Coding Standards | ✓ |
| SWC-107 | Reentrancy | CWE-841: Improper Enforcement of Behavioral Workflow | ✓ |
| SWC-106 | Unprotected SELFDESTRUCT Instruction | CWE-284: Improper Access Control | ✓ |
| SWC-105 | Unprotected Ether Withdrawal | CWE-284: Improper Access Control | ✓ |
| SWC-104 | Unchecked Call Return Value | CWE-252: Unchecked Return Value | ✓ |

| ID | Title | Relationships | Test Result |
|-------------------------|--------------------------------|------------------------------------------------------------------------------|-------------|
| SWC-103 | Floating Pragma | CWE-664: Improper Control of a Resource Through its Lifetime | ✓ |
| SWC-102 | Outdated Compiler Version | CWE-937: Using Components with Known Vulnerabilities | ✓ |
| SWC-101 | Integer Overflow and Underflow | CWE-682: Incorrect Calculation | ✓ |
| SWC-100 | Function Default Visibility | CWE-710: Improper Adherence to Coding Standards | ✓ |

7. Test Deployment

Account 1: 0x03aDBe9615A7C66DAaff8e2e10f197E1B663071f (Contracts Owner)
Account 2: 0xD32154Aca268223100e9e8F850ed9AE41B9F9a86 (User1)
Account 3: 0x6d5A684ff9aa0E240C3D3D5a998429658c613753 (User2)
Account 4: 0xFc0BBFE1e6391Ad733B0016d41E50cFa2725717E (User3)

_rewardPerYear = 0.1 ether;
_rewardPerMonth = 0.03 ether;
_rewardPerDay = 0.001 ether;
_rewardPerhour = 0.00005 ether;

No compounding conversions taken into account, as figures are not representative reality but for testing code calculations purposes only

7.1 Deployments

DummyStakeToken: 0xf52E3a7741a79f12aE8b00BdeF7185b1974B0DaD

Tx: <https://mumbai.polygonscan.com/tx/0xdbb19051782a589d7312f7defbda1d53b932c3e140ed62c11627b38dc1b9e025>

Contract: <https://mumbai.polygonscan.com/address/0xf52E3a7741a79f12aE8b00BdeF7185b1974B0DaD>

StakingRewardsContract: 0x9D21d18E1d84b84dFC50f80D52bdc35f16A9dAb7

Tx: <https://mumbai.polygonscan.com/tx/0x982fd7866c8fce8743c1230998b73d8ac634456fca3f266027bc05b86a65feb3>

Contract: <https://mumbai.polygonscan.com/address/0x9d21d18e1d84b84dfc50f80d52bdc35f16a9dab7>

Initialize Tx: <https://mumbai.polygonscan.com/tx/0x431be7372ad1a5124fdb80fca9324f90b950a626b559533d2bba194f75d76b5a>

End Deposit time: Wed Apr 21 2021 16:45:00 GMT+0200 (Central European Summer Time)



7.2 Staking

User 1 (Stake 10 token)

Approve DummyStakeToken Tx:

<https://mumbai.polygonscan.com/tx/0xa7a369f3039666899060b9590ca8892578a8800e7a4e005a3d30e4b01caf0e2c>

Stake Tx: (Mar-15-2022 09:18:44 AM +UTC)

<https://mumbai.polygonscan.com/tx/0x348db3d233f546c08ae7a91d14f38f05c6b9765247e9efc00f19444e871bae63>

User 2 (Stake 10 token)

Approve DummyStakeToken Tx:

<https://mumbai.polygonscan.com/tx/0x2acfc4540ac00a5a5d42a023432c4a6affded46a71f6f8edcaeaf9ac35d2d126>

Stake Tx: (Mar-15-2022 09:21:55 AM +UTC)

<https://mumbai.polygonscan.com/tx/0xd19ceb1b881264846412df8d4624e86c6414af7a566f42758585ab8519299a36>

7.3 Withdraw

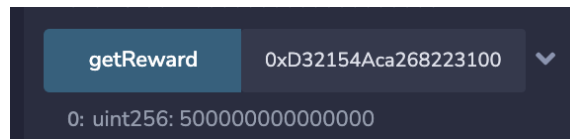
1. **User can withdraw stake at any time receiving reward correctly** (Mar-15-2022 09:40:12 AM +UTC)

Withdraw small 10 tokens in wei format (User 2)

Tx: <https://mumbai.polygonscan.com/tx/0x9522c8f5218eee12d9bd8d6875f0fec738e9f22931d06a79e7ae44d1c36426d0>

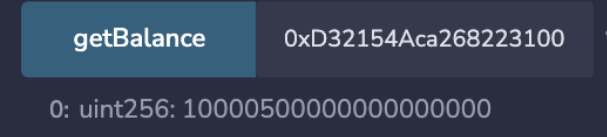
Balance Tokens : 9.999999999999999 in ether format

As expected no rewards added as time is 09:21:55 AM +UTC - 09:40:12 which is less than an hour



getRewards(User1) after an hour of staking 0.0005 = 5000000000000000 which correctly corresponds to 0.0005 per hour since the user staked from (Mar-15-2022 09:18:44 AM +UTC) to (Mar-15-2022 10:19:44 AM +UTC) which is just over an hour.

Reward = 0.00005 tokens / hour * 1 hour * 10 tokens = 0.0005



getBalance(User1) after an hour of staking = **10tokens + 0.0005 = 10.0005**
10000500000000000000

User1 withdraws all tokens(10000500000000000000) and stakes again
withdraw(user1) Tx:

<https://mumbai.polygonscan.com/tx/0xc112c08aeb98703c870f03e603e15facedd3f7b189f521242dfe15be573dfa20>

[illegible]

getBalance(user1) : reverts as correctly removed from stakers (StakingRewards.sol line 74 - line 89)

User1 stakes 10 tokens again for almost 2 hours about 1hour 58 minutes and withdraws but still gets the same reward as staking for an hour, additional minutes e.g if less than 1 hour do not count. This may impact the fairness of the system.

Approve DummyStakeToken Tx:

<https://mumbai.polygonscan.com/tx/0x2c998946cb396da50fd55d60869fafe087d4096bdef8215ae3daa1c9e3afa785>

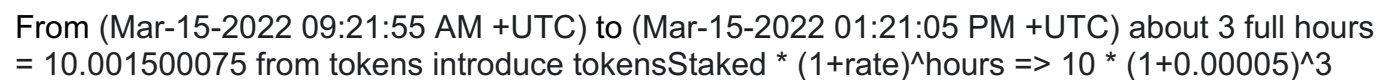
Stake Tx: (Mar-15-2022 11:05:49 AM +UTC)

<https://mumbai.polygonscan.com/tx/0x67a65d2a5df28487dad85619e9df2b6d7f7d7e5ebba2a12ac5c849321ed8a23e>



Amount withdrawn is $(10.0005) = 10000500000000000000$ after 1 hour and 58 minutes giving similar rewards as for 1 hour.

Tx: <https://mumbai.polygonscan.com/tx/0x4926ea2e71b83e4241d3b384ea020a6924705c1a0b11b2b71c094f1c6581e6ba>

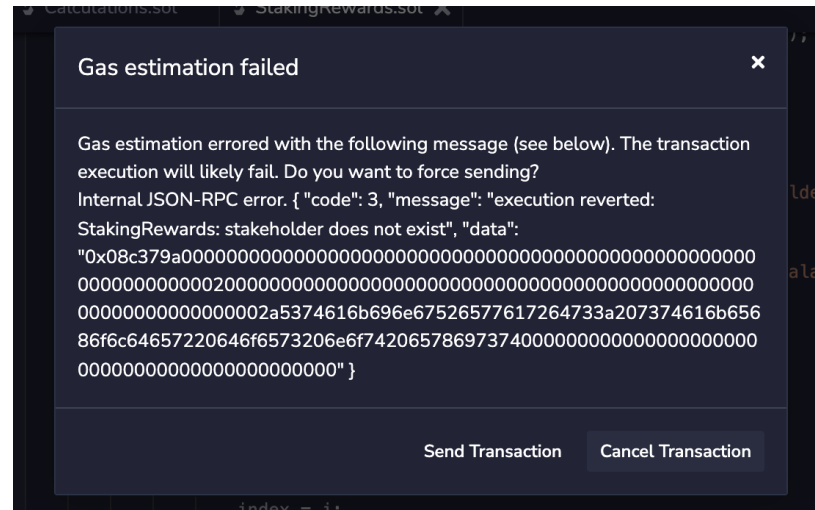


Owner tries to withdraw does not have any stake still calls withdraw()



3. Only the owner is able to set rewards

User 3 tries to setRewards => 'ce30' error message from onlyOwner() modifier (line 31 OwnableUpgradeable.sol)



8. Verify Claims

8.1 Staking rewards are unfair as rewards are rounded down per hour or time period used e.g reward 1 hour given even if stake for 1 hour 50 minutes, rewards are lost

Status: tested and verified ✗

It is recommended to consider solutions like adding rewards per second and using exponentiation to compound the rewards between deposit and withdrawal periods. e.g using $[\text{balance} + \text{balance}(1 + \text{rewardsPerSecond})^{\text{numberSeconds}}]$ also helping avoid the looping and checks if to iterate over years or days or hours.

8.2 The user can withdraw the stake at any time and receiving the correct reward

Status: tested and verified ✔

8.3 Owner/Deployer is not able to withdraw staked token from user

Status: tested and verified ✔

8.4 Owner/Deployer cannot pause the staking

Status: tested and verified ✗

Since StakingRewards.sol is an upgradeable contract owner can upgrade to a new Address that impacts existing functionality

8.5 Only the Owner is able to set rewards

Status: tested and verified ✔

8.6 Rewards send to the contract cannot be withdrawn by the Owner

Status: tested and verified ✔

8.7 The smart contract is coded according to the newest standards and in a secure way

Status: tested and verified ✔

9. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase. The final debriefs took place on the March 17, 2022. The overall code quality is good and not overloaded with unnecessary functions, these is greatly benefiting the security of the contract. It correctly implemented widely used and reviewed contracts from OpenZeppelin.

The main goal of the audit was to verify the claims regarding the security of the smart contract and the claims inside the scope of work. During the audit the following issues have been found: no critical, 1 high, 1 medium, 2 low and 4 informational.

Update (09.06.2022): All issues have been addressed and fixed

10. Deployed Smart Contract

VERIFIED

<https://polygonscan.com/address/0x3c868fe859ef46a133e032f22b443e6efd617449#code>

11. About the Auditor

Chainsulting is a professional software development firm based in Germany that provides comprehensive distributed ledger technology (DLT) solutions. Some of their services include blockchain development, smart contract audits and consulting.

Chainsulting conducts code audits on market-leading blockchains such as Hyperledger, Tezos, Ethereum, Binance Smart Chain, and Solana to mitigate risk and instil trust and transparency into the vibrant crypto community. They have also reviewed and secured the smart contracts of 1Inch, POA Network, Unicrypt, Amun, Furucombo among numerous other top DeFi projects.

Chainsulting currently secures [\\$100 billion](#) in user funds locked in multiple DeFi protocols. The team behind the leading audit firm relies on their robust technical know-how in the blockchain sector to deliver top-notch smart contract audit solutions tailored to the clients' evolving business needs.

The blockchain security provider brings the highest security standards to crypto and blockchain platforms, helping to foster growth and transparency within the fast-growing ecosystem.

Check our website for further information: <https://chainsulting.de>

How We Work



1 -----

PREPARATION

Supply our team with audit ready code and additional materials



2 -----

COMMUNICATION

We setup a real-time communication tool of your choice or communicate via e-mails.



3 -----

AUDIT

We conduct the audit, suggesting fixes to all vulnerabilities and help you to improve.



4 -----

FIXES

Your development team applies fixes while consulting with our auditors on their safety.



5 -----

REPORT

We check the applied fixes and deliver a full report on all steps done.