Cryption Smart Contracts Initial Audit Report

Overview

Scope of Audit

The scope of this audit was to analyse and document the **Cryption** smart contracts codebase for quality, security, and correctness.

Check Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked maths
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contracts, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods, and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational Issues

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of security issues per severity.

TYPE	HIGH	MEDIUM	LOW	INFORMATIONAL
Open	1	3	4	0
Acknowledged	0	1	0	0
Closed	0	0	0	0

Introduction

During the period of **Feb 11, 2022 to March 11, 2022** - QuillAudits Team performed a security audit for **Cryption** smart contracts.

The code for the audit was taken from following the official link**:**

Codebase: Freeze commit - 5a1c15d21f97f2c9dac1188e07d15450616e3fcf

Issues Found – Code Review / Manual Testing

High Severity Issues

- **[H1] updatePoolOwnerFeePercentage doesn’t follow ownable access rights**

  The OwnerFeePercentage update should be controlled only by the owner, not any other address. If any other address is able to set the fee then the whole staking mechanism will behave wrongly and a malicious address can exploit the FeeApplicable calculation.
  We always recommend making all the state updating functions to be called by onlyOwner.

  Status: Open

Medium Severity Issues

- **[M1] withdrawAcquiredFees should be OnlyFeeBeneficiary**

  withdrawAcquiredFees() is an onlyOwner function, although the feeBeneficiary is the one receiving the fees. This could result in the owner not calling the function, resulting in denial of fees transfer to the feeBeneficiary for a long time.

  We recommend making the function to onlyFeeBeneficiary instead of onlyOwner.

  Status: Open

- **[M2] Referral manager Not used in contracts .**

  ReferralManager was used in the contract. But the implementation of referralManager.sol in the contracts was not found(only its interface was there.)

The team acknowledged that they have just added a support for referral manager for future as they wanted an ability to start / stop referral functionality

Status: Acknowledge

- **[M3] No rescueFund method is implemented in all the contracts**
  There should be a rescueFund method to rescue the tokens funds which are mistakenly sent and are not part of the contracts.

  The behaviour of the rescueFund should only be operated by the owner and nobody else.

  Recommendation for rescue funds method
  ```
  function rescueFunds(IERC20 _token, address _recipient) external onlyOwner
  {};
  ```

  Status: Open

- **[M4] No method to take out Ether from Factory contracts**

  We recommend making a function to pull the deposited Ether from the contracts such that it doesn't get locked forever.

  Status: Open

Low Severity Issues

- **[L1] In updateEndBlock the updateEndBlockNumber should be checked**

  In updateEndBlock the while updating the updateEndBlockNumber the current block.number should always be less than the new updateEndBlockNumber.

  We recommend incorporating the same check while updating.

  Status: Open

- **[L2] Use SafeTransfer instead of normal Transfer**

  As the tokens used in purchase/vesting can be USDT and we should use safeTransfer instead of normal transfer while transferring those tokens [Reference]

  Status: **Open**

- **[L3] Used locked pragma version:**

  The pragma versions used in the contract are not locked. Consider using the latest versions among 0.8.11 for deploying the contracts and libraries as it does not compile for any other version and can be confusing for a developer. Solidity source files indicate the versions of the compiler they can be compiled with.

  ```
  pragma solidity ^0.8.0; // bad: compiles between 0.8.0 and 0.8.11
  pragma solidity 0.8.0; // good : compiles w 0.8.0 only but not the latest version
  pragma solidity 0.8.11; // best: compiles w 0.8.11
  ```

  Status: **Open**

- **[L4] Gas optimizations:**
  1. As by default EVM assigns 0 to uint256 data type. Explicitly reassigning the same value at line no 54 in stakingPool is a waste of gas. We recommend removing the assignment.
  2. In the loop at line 439 of Staking Pool the variable rewardPool.length is recalculated every time in the loop. We recommend storing the variable in a uint256 data type and using the same variable in the loop.
  3. In the crowsale and Liquidity locker the `initialized` is reassigned to its default value. We recommend to remove the assigning of false to `initialized.`

  Status: **Open**

Informational Issues

None

Goerli Testnet Test Contract

FeeBeneficiary Address - 0xF164A4DE04D55f268AdB795434BcE932Ea8Db731
FeeManager - 0xdba93782b50284488c00Ea3773A13f1999792fCD
LinerVesting - 0x6d4F50a671F3004B3455F7883787e6876c19A5D3
Vesting Factory - 0x5c2958F4bA2e02c913cc4DB8F8d9aeb8196E8410

MOCKERC20 - 0xe7743A888F1a6ea7C53Aea5ffd5747Ae67fE4a0E

LP TOKEN - 0xA828c9A09F5fd8a32e5F9AfE3aF24DCA871D111A
REWARD TOKEN1 - 0x23791EABAA14f4bD30B25b2f2bc72dEB7C31aFf5
REWARD TOKEN2 - 0xf402eb169099143Ae80b7d19ad83880DFEe11261
Staking Pool - 0x38c87d2c2f86F18a1dd56Cbc85cD4314D4182961
Staking Factory - 0x1F1588A6b97067827520e1A07C112E0B7d5913b9

Liquidity Locker - 0xE4811C5Dc107945E73311584Cba23f406b05CD1F
Liquidity Locker Factory - 0x09ECd5D2e10D2CA4dD957b23dE55aFadEF74147b

Crowdsale - 0x8d9632D7D0DeBe8465F28417dC560658eC33B2b8
purchaseToken1 - 0x01e297525377fa68cbcb938a441cbbb84a3baf35
purchaseToken2 - 0x2c4806f5616F389211B030677c97a778732F78bC
tokenInCrowdsale - 0xa1d8b1e2d31a70bfb108be36a8e49d3b748b0f34
LaunchPadFactory - 0xE8160863e7d9e29234aebc47BD450DeBbec8aE77

Functional Tests

PASSED

**********************Vesting**********************

addImplementation

https://goerli.etherscan.io/tx/0x3dc4aaac81d6d34ffb33ca8b7b6eedb05fff2a5fd13d2ebf66ae5e2bf82f4d08

launchVesting with 10,0,_encodedData
https://goerli.etherscan.io/tx/0x53fb6824fdba914baa469dadd6b2a5f5553618b10c19ef64f127f37fcc0c40cb
Vesting contract launched via minimal proxy
https://goerli.etherscan.io/address/0x941e1883f9714686a98c32fd1c68efffe5fe1526#code
set Fee manager and enable FeeManager
https://goerli.etherscan.io/tx/0x9a333a2b7dd06c75e047475c5c34264abe84c169e8d69ee54825ce64c09c9083
Launch vesting with Incorrect ID 1
https://goerli.etherscan.io/tx/0x43ca1b2f89c6271e195f30815bfd33fbc553f112de5f481a28e00d3c5794394b
Invalid fee sent
https://goerli.etherscan.io/tx/0x34e694a4794599993f333538d6bd0788e9dd3c4b27ffa5512284c559655326b6
Update fee Info
https://goerli.etherscan.io/tx/0x01fd552a0b65c76550204f35e896d87b2805c963927f3fbd04dc8550ba5c381b
No approval for vesting factory
https://goerli.etherscan.io/tx/0x7d6ca0f9b4d727108c50b21ffbe6e6165e53e01a7a68015454f05ca64f68c07f
Approve 100 token amount for vesting factory from user
https://goerli.etherscan.io/tx/0x40ed04153cf2f3e93fa47d4436790e6451e43345a5cdef94f205c838a4b737e6
Launch new Vesting contract with 10, 0, _encodedData
https://goerli.etherscan.io/tx/0x696b760c9733097f669e2b6c13e6b161784239abf788336d0e40b31477a42c41
Vesting contract lanched with 0 index Liner vesting considering feeManger
https://goerli.etherscan.io/address/0x60a1b0c6203be187275dc3fc816d40aa26655881
approve 100000 mock tokens to vesting contract
https://goerli.etherscan.io/tx/0x4efb8a02781db8313224e5e64e8b9aaef7f4138882e58b76ddcbdea6e4786b67
create vesting schedule
https://goerli.etherscan.io/tx/0x89c351675c477578ee5c1224cf3e9efc4afa0a2fb474cc9feda0f8d6538329d7
Draw down before cliff duration
https://goerli.etherscan.io/tx/0x0c7696ee276cc36a4b4aee317dea0dbaa72fdc33754be41fe799e717b7ca9c37

Launch new vesting contract with smaller cliff duration
https://goerli.etherscan.io/tx/0x032c84ff301a109592064ae68d75c9ab5e9226726932da5b0cbb2739f9736c85
approve 100000 mock tokens to vesting contract
https://goerli.etherscan.io/tx/0xcb343225b0df1d2a5b4d79c34212fe4edfc020139d4c116df47f843bffe02d26
create vesting schedules
https://goerli.etherscan.io/tx/0xddc90ff0eab81dfe624dd0eeab5835873632d737edf1d117f7455d7a06abb54f

PASSED
****************_**Staking**_****************
addImplementation
https://goerli.etherscan.io/tx/0x2a3ac959c56e7d2b99f767f1ce1a7243d8d50347a32a780cb23f5cf0941f77cb
approve 10^21 rewardToken to staking factory
https://goerli.etherscan.io/tx/0x5e4303c6b81960f8a7433f2edfdceaf3e3d34fd844f0085f7e5808f6c8c7d09a
Launch Staking with no feeManager
https://goerli.etherscan.io/tx/0x3c90abced3f9b13db1f950ad9b124d9883733d7295a21b53057368ead2189645
staking contract launched via minimal proxy
https://goerli.etherscan.io/address/0x9065F8100D81254Ce61ac6e2196DC0C254b1B7F9#code
add feeManager
https://goerli.etherscan.io/tx/0x79421ee77da1909683e61ea4e78c2014a77095fe2ea86a8574a810dd7f0c17d0
approve 10^22 from user to staking factory
https://goerli.etherscan.io/tx/0x6056554bf81746e186c0b4f451efb58a06876eba7bb0e8eb47f167b9daec3152
update fee info
https://goerli.etherscan.io/tx/0x3ec8750163c8c03c35e0ed05111f0cbd0890ce1cf32e9904f8657dbbd498fb04
Launch staking contract with feeManger enable and 0 Staking pool
https://goerli.etherscan.io/tx/0x247f1685cb79e61aeb16506180b74379413c8fc6e28ada1f9cecba493856cd08
Staking contract with minimal proxy
https://goerli.etherscan.io/address/0xeb901ddd8cc2bd73eafed7e9d053cc2070b5f143#code
updateOwnerFunctionality

https://goerli.etherscan.io/tx/0x70c87d02ed526ed1751e188cfd24639ed889cbafed683a4e96433ed3fdf2904e
add RewardToken2 with not a pool owner
https://goerli.etherscan.io/tx/0x20d7abc81c6b9b15be19b429791cfc46b501e2e3e90f40ab812e786b07c4a108
approval to pool1 for 100000 LP tokens from user1
https://goerli.etherscan.io/tx/0x781d0b10c809d137e066b31ea735468c7bc893613e0bf8f3541878aef811b2e9
deposit 100 in pool1
https://goerli.etherscan.io/tx/0x760a20428a921a7257133bab3d416a60bdde37a1e43a60b977128623b76a73a0
withdraw 50 in pool1
https://goerli.etherscan.io/tx/0x413f64529ea1dc57f331bc256c08d90f34d825322b1e034cf0366b8b78ac0a9e
deposit 10000 in pool1
https://goerli.etherscan.io/tx/0x42a635971ae739cc9481ef8ed47c024d2fc325723d428488551d562264cea1fe
withdraw 10000 in pool1
https://goerli.etherscan.io/tx/0x017230c81282d6d3e73da25583ddcefaba89dce8203a3c25489c3600bcf02b02
transfer reward token to poolOwner
https://goerli.etherscan.io/tx/0xecc7ba1dc2414d897a19f6d086dd0eb3b12bb8d5d0ce92b682120b327f26e317
updateBlockReward to 100
https://goerli.etherscan.io/tx/0x0ad5a04ea77a2a0887f0accaa4d76bd9d22531d046def2425a7fa9c60326b0f6

FAIL
anybody can update poolOwnerFeePercentage by calling updatePoolOwnerFeePercentage
https://goerli.etherscan.io/tx/0xd297a893065c0cc29e00658acd11c246e419323d8ff44b3f841fa869993a5a50
updateEndBlock is less than current block
https://goerli.etherscan.io/tx/0xd5e629e2ea98c98e87e4c0f4a62e5d91bfb2719b9f034ec2fd19f935db9c9595

PASSED
****************_**LaunchPad**_****************

addImplementation

https://goerli.etherscan.io/tx/0x937721adf4a12f5ddbc67906f9fc67b121062291ffa95ad5cb172d36ed690640

approveFactory with 1000 tokens

https://goerli.etherscan.io/tx/0xb94af1e04723a7386eb76c2533aa4e169603bb75d4d3213c56ed312468bb4485

launch CrowdSale with two purchaseToken

https://goerli.etherscan.io/tx/0x2319891abe9afd494b52e7e450434a35d63966518f85e7af6f2a261d609b064a

Crowdsale Contract -0x73858AcC6324d72D3D53A54ba61363d6C4534484

Purchase token before the crowdSale starts

https://goerli.etherscan.io/tx/0x77ffce9affc88d6584c478410e22616eb5d70d9606ff1543e0fae9c212917ebf

updated rate for purchase token1 and token2

https://goerli.etherscan.io/tx/0xdd05011c4f471c5b81c299d359db1eb8ef212c5ad0d89b4989edb3890b3eeb77

https://goerli.etherscan.io/tx/0x2f040188a0af4d7e17b55b870693e8e7b76d1371ae471a4f7b3c64a43ff59d42

approve 20 tokens to CrowdSale for both purchaseToken from user

https://goerli.etherscan.io/tx/0x78ffc413c9085c165a1acc4c0c1eca522129685eb8ab36f46ba9694415bed391

https://goerli.etherscan.io/tx/0xc53dd32998a5660d79ec147008f3a5d4c606e24da8a44ad26d3dd3ff76370a40

Purchase 20 tokens each from CrowdSale

https://goerli.etherscan.io/tx/0x68b2348d319fdd57a74e4e663120e9db132f59b7af2ee248dfa8a0f44272af75

https://goerli.etherscan.io/tx/0x6b70476db666ec7284a557fc431d8fdad0f023b4c65ec3fe6602d0d84453810e

DrawDown not allowed

https://goerli.etherscan.io/tx/0xbcf522fe69161cd6d5bd9aa582ae571490e9378f74c8c8e07a555de7d7198ab1

End CrowdSale

https://goerli.etherscan.io/tx/0x545140b425288e3cede77cc9a24c52a73b760715c8cefb0a956b0d831abbf77d

CrowdSale Can't be reinitialized

https://goerli.etherscan.io/tx/0xdb4d37fcf63a522f5fc0136e4957ef7d4ff2076ee30c59f0aac429fc7a31c97d

Automated Tests

Slither:

Snippets provided on Google drive.

Results:

One major issue and several medium and low severity Issues are found. All the issues have been categorised above according to their level of severity. We recommend fixing all the Issues and provide the fixed version.
Apart from fixing the issue the Cryption team should also focus on testing of the contract as current test scenarios are very bad and coverage is not good. Incorporating more test cases scenarios always makes the contract robust.

Closing Summary

Several issues of High, Medium and Low severity issues have been reported.
We recommend fixing the issues before proceeding to the public mainnet release.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or endorsement of the **Cryption platform**. This audit does not provide a security or correctness guarantee of the audited smart contracts. 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. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the **Cryption** Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.