HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Argon
Date: June 22th, 2021

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Smart Contract Code Review and Security Analysis Report for argon-staking-contracts
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Token, Defi, Staking, Pool
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/Argon-Foundation/argon-staking-contracts/tree/d26031dcce6509119fe6bf60413a153f693559b1
Deployed contract	
Timeline	15 JUNE 2021 - 22 JUNE 2021
Changelog	22 JUNE 2021 - INITIAL AUDIT

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by Argon (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer's smart contract and its code review conducted between 15 Jun 2021 - 22 Jun 2021.

Scope

The scope of the project is smart contracts in the repository:

Repository https://github.com/Argon-Foundation/argon-staking-contracts/tree/d26031dcce6509119fe6bf60413a153f693559b1
File:
ArgonStakeMaster.sol
ArgonStakingPool.sol

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency ▪ Data Consistency
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Executive Summary

According to the assessment, the Customer's smart contracts are secured.

Insecure	Poor secured	Secured	Well-secured

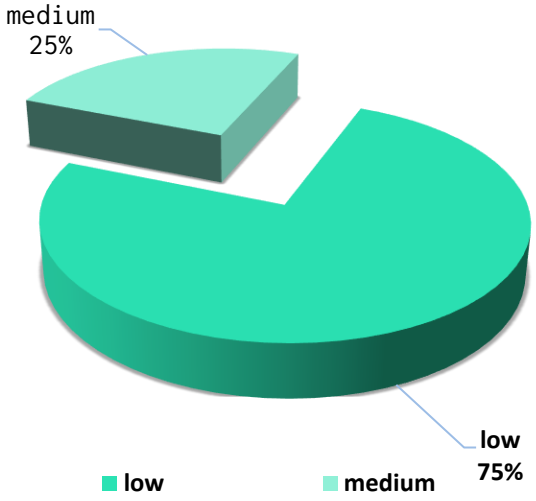
You are here

Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. A general overview is presented in AS-IS section, and all found issues can be found in the Audit overview section.

Notice: There is no needed QA test exists. We strongly recommend development team to add unit and integrations tests of existed smart contracts.

Security engineers found 3 low and 1 Medium issues during the audit.

Graph 1. The distribution of vulnerabilities after the first review.



Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations, and info statements can't affect smart contract execution and can be ignored.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

AS-IS overview

ArgonStakeMaster.sol

Description

The contract to create a new pools and store list of existed pools.

Imports

ArgonStakeMaster has following imports:

- ArgonStakingPool.sol;

Inheritance

ArgonStakeMaster has following inherited contracts:

- Ownable

Usages

ArgonStakeMaster contract has no usages.

Structs

ArgonStakeMaster contract has no data structures.

Enums

ArgonStakeMaster contract has no enums.

Events

ArgonStakeMaster contract has following events:

- PoolCreated

Modifiers

ArgonStakeMaster has no modifiers.

Fields

ArgonStakeMaster contract has following fields and constants:

- address[] public pools;

Functions
ArgonStakeMaster has following public functions:

- createStakingPool
- getAllPools

ArgonStakingPool.sol

Description

The contract to manage and use concreate stake pool.

Imports

ArgonStakingPool has following imports:

- IERC20
- SafeERC20
- Address
- SafeMath
- ReentrancyGuard
- Ownable
- Context

Inheritance

ArgonStakingPool has following inherit contracts:

- Ownable
- ReentrancyGuard

Usages

ArgonStakingPool contract has following usages:

- using SafeMath for uint256;
- using SafeERC20 for IERC20;

Structs

ArgonStakingPool contract has following data structures:

- UserInfo
- UserPoolInfo

Enums

ArgonStakingPool contract has no enums.

Events

ArgonStakingPool contract has following events:

- FinishBlockUpdated
- PoolReplenished
- TokensStaked
- StakeWithdrawn
- WithdrawAllPools
- WithdrawPoolRemainder

Modifiers

ArgonStakingPool has no modifiers.

Fields

ArgonStakingPool contract has following fields and constants:

- IERC20 public stakingToken;
- IERC20 public rewardToken;
- uint256 public startBlock;
- uint256 public lastRewardBlock;
- uint256 public finishBlock;
- uint256 public allStakedAmount;
- uint256 public allPaidReward;
- uint256 public allRewardDebt;
- uint256 public poolTokenAmount;
- uint256 public rewardPerBlock;
- uint256 public accTokensPerShare; // Accumulated tokens per share
- uint256 public participants; //Count of participants
- bool public isPenalty;
- uint256 public penaltyRate;
- uint256 public penaltyBlockLength;
- address public penaltyAddress;

- mapping(address => UserInfo) public userInfo;
- mapping(uint256 => UserPoolInfo) public userPoolInfo;
- mapping(uint256 => bool) public availablePools;

Functions
ArgonStakingPool has following public functions:

- withdrawPoolRemainder
- withdrawAllPools
- withdrawStake
- stakeTokens
- updatePool

ArgonStakingPool has following view functions:

- getUserPoolInfo
- getUserInfo
- pendingReward

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Audit overview

Critical

No critical issues were found.

High

No high issues were found.

Medium

1. All contracts should use the latest compiler version.

Low

1. There is a redundant call of transferOwnership() in createStakingPool function of ArgonStakeMaster contract.
2. Pools data should be updated before calling transferPendingReward and before user checks in the withdrwal function. Call updatePools() function at the start of withdrawing function.
3. We strongly recommend you to change userPoolInfo data structure to mapping(uint256 => mapping(uint256 => UserPoolInfo)). It will make cade more clear and transparent, also it will help to avoid low level ABI functions and raw keccak256.

Lowest / Code style / Best Practice

No lowest issues were found.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, high-level description of functionality was presented in As-Is overview section of the report.

Notice: There is no needed QA test exists. We strongly recommend development team to add unit and integrations tests of existed smart contracts.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security engineers found 3 low and 1 Medium issues during the audit.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on blockchain platform. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.