

HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Argon Foundation
Date: May 25th, 2021



This document may contain confidential information about IT systems and the intellectual property of the Customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities are fixed - upon a decision of the Customer.

Document

Name	Smart Contract Code Review and Security Analysis Report for Argon Foundation - Second Review
Approved by	Andrew Matiukhin CTO Hacken OU
Type	ERC20 Locker, Work Contracts
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Git repository	https://github.com/Argon-Foundation/platform-contracts/tree/8996d21499ede3f2eb36320ded9563f5467a0fa6
Timeline	14 MAY 2021 - 24 MAY 2021
Changelog	19 MAY 2021 - INITIAL AUDIT 24 MAY 2021 - SECOND REVIEW 25 MAY 2021 - THIRD REVIEW



Table of contents

Table of contents	3
Introduction	4
Scope	4
Executive Summary	6
Severity Definitions	8
Audit overview	9
Conclusion	11
Disclaimers	12

Introduction

Hacken OÜ (Consultant) was contracted by Argon Foundation (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 on May 24th, 2021.

Scope

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

<https://github.com/Argon-Foundation/platform-contracts/tree/8996d21499ede3f2eb36320ded9563f5467a0fa6>

ArgonFreelancers.sol
liquidityLock.sol

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

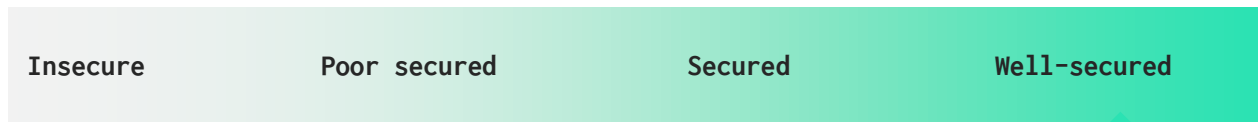
Category	Check Item
Code review	<ul style="list-style-type: none">▪ Reentrancy▪ Ownership Takeover▪ Timestamp Dependence▪ Gas Limit and Loops▪ DoS with (Unexpected) Throw▪ DoS with Block Gas Limit▪ Transaction-Ordering Dependence▪ Style guide violation▪ Costly Loop▪ ERC20 API violation▪ Unchecked external call▪ Unchecked math▪ Unsafe type inference▪ Implicit visibility level▪ Deployment Consistency▪ Repository Consistency▪ Data Consistency



Functional review	<ul style="list-style-type: none">▪ Business Logics Review▪ Functionality Checks▪ Access Control & Authorization▪ Escrow manipulation▪ Token Supply manipulation▪ Asset's integrity▪ User Balances manipulation▪ Kill-Switch Mechanism▪ Operation Trails & Event Generation
-------------------	---

Executive Summary

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



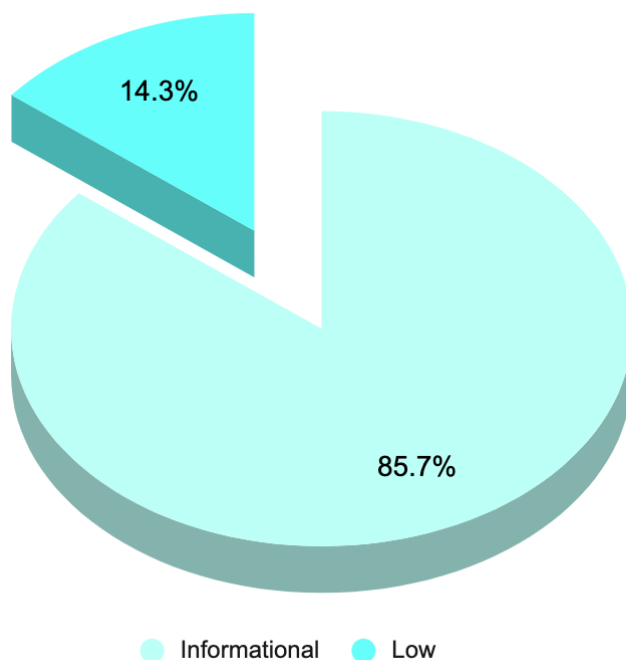
Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview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

Security engineers found 1 low and 6 informational issues during the first review.

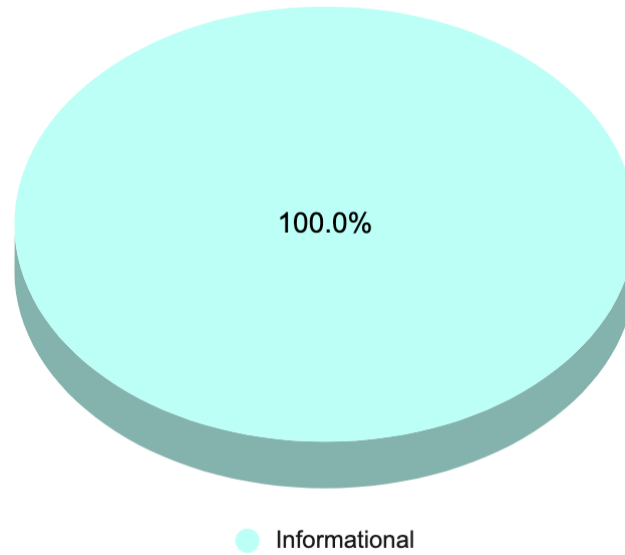
Security engineers found 1 informational issue during the second review.

Security engineers found **no issues** during the second review.

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



Graph 2. The distribution of vulnerabilities after the second review.



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.

Audit overview

■ ■ ■ ■ Critical

No Critical severity issues were found.

■ ■ ■ High

No High severity issues were found.

■ ■ Medium

No Medium severity issues were found.

■ Low

1. **Vulnerability:** Unused contract

Contract ApproveAndCallFallback defined in the ArgonFreelancers.sol is never used.

Fixed before second review

■ **Lowest / Code style / Best Practice**

1. **Vulnerability:** Incorrect version of solidity

Contract could not be built with solidity version 0.4.17 which, but should work starting the 0.4.24.

Recommendation: Please use recommended solidity versions to deploy

- 0.5.16 - 0.5.17
- 0.6.11 - 0.6.12
- 0.7.5 - 0.7.6

Fixed before second review

2. **Vulnerability:** Naming convention

Solidity defines a [naming convention](#) that should be followed.

Fixed before second review

3. **Vulnerability:** Boolean equality

Boolean constants can be used directly and do not need to be compared to **true** or **false**.

Fixed before second review

4. **Vulnerability:** Costly operations inside a loop

Decrementing the state variable in a loop incurs a lot of gas because of expensive **SSTOREs**, which might lead to an out-of-gas exception.

Recommendation: Please consider using a local variable to hold the loop computation result.

Fixed before third review

5. **Vulnerability:** Too many digits

Literals with many digits are difficult to read and review. Please consider using ether units and/or scientific notation and/or dashes as separators instead.

Recommendation:

- replace 5000000000000000000000000000 to **5e6 ether** or **5_000_000 ether**
- replace 1000000 to **1e6** or **1_000_000**

Fixed before second review

6. **Vulnerability:** Public function that could be declared external

public functions that are never called by the contract should be declared **external** to save gas.

Fixed before second review



Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

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

Security engineers found **1** low and **6** informational issues during the first review.

Security engineers found **1** informational issue during the second review.

Security engineers found **no issues** during the second review.



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 the 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.