HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Constellation Network
Date: July 9th, 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 Lattice, Powered by Constellation - Third Review
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Staking Pool
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Contract address	https://etherscan.io/address/0x74299a718b2c44483a27325d7725f0b2646de3b1#code
Timeline	29 JUNE 2021 – 9 JULY 2021
Changelog	1 JULY 2021 – INITIAL AUDIT 9 JULY 2021 – SECOND REVIEW 11 AUGUST 2021 – THIRD REVIEW

Table of contents

Introduction 4
Scope 4
Executive Summary 5
Severity Definitions 7
Audit overview 8
Conclusion 10
Disclaimers 11

Introduction

Hacken OÜ (Consultant) was contracted by Constellation Network (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 July 9th, 2021.

Scope

The scope of the project is next smart contract:

https://etherscan.io/address/0x74299a718b2c44483a27325d7725f0b2646de3b1#code

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	▪ 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 ▪ Asset’s integrity ▪ User Balances manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Executive Summary

According to the assessment, the Customer's smart contract is well-secured but having some issues with gas consumptions and possible view function inaccessibility.

Insecure | Poor secured | Secured | Well-secured

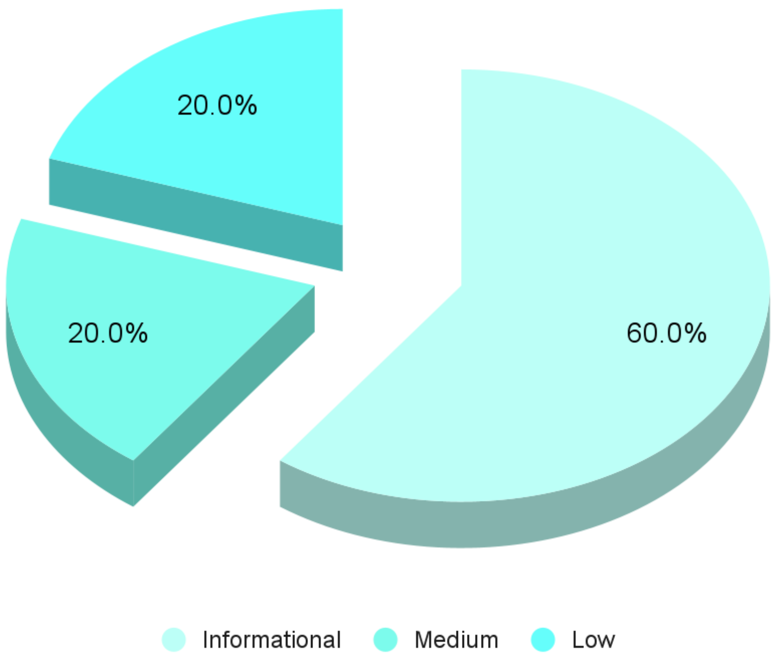
You are here

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. All found issues can be found in the Audit overview section.

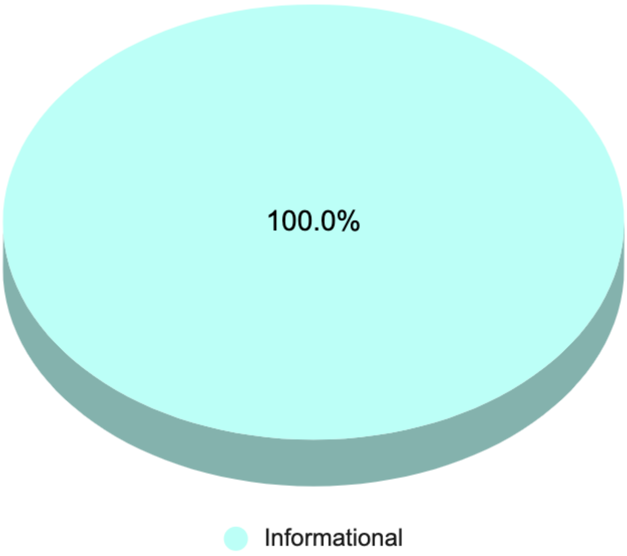
Security engineers found 1 medium, 1 low and 3 informational issues during the first review.

Security engineers found 1 informational issue 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

1. Vulnerability: Possible unreachable view function

   Using nested for-loops on the unpredictable amount of iterations may cause to great gas usage and make view-functions inaccessible

   Recommendation: If it really needs to have this functionality on-chain, please consider adding a kind of pagination in the method, when you're specifying the starting index and amount of elements to return.

   Otherwise you may want to calculate everything off-chain using simple view methods.

   Fixed before second review

🟦 Low

1. Vulnerability: Multiple access to state variable

   Putting the state variable in the condition block of the `for` operator leads to continuous calls to this state variable for checking the condition which increases gas usage.

   Recommendation: we recommend to save `projects[_projectId].numberOfPools` into the local variable and then use it in the condition block.

   Fixed before second review

🟦 Lowest / Code style / Best Practice

1. Vulnerability: Boolean equality

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

Recommendation: remove the equality to the boolean constant.

Fixed before second review

2. Vulnerability: Code optimization

   Multiple usage of the same state variable increases gas usage

   Recommendation: we recommend to store the value of `projects.length` as local variable before pushing new project into it and then use it to save in the `projectNameToProjectId` and for emitting `ProjectAdded` event

   Fixed before second review

3. Vulnerability: Maximum line size

   Line length of lines 274, 247 and 160 exceed maximum line length recommendation.

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 medium, 1 low and 3 informational issues during the first review.

Security engineers found 1 informational issue during the second review.

Category	Check Items	Comments
➔ Code Review	➔ Style guide violation	➔ Maximum line size

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.