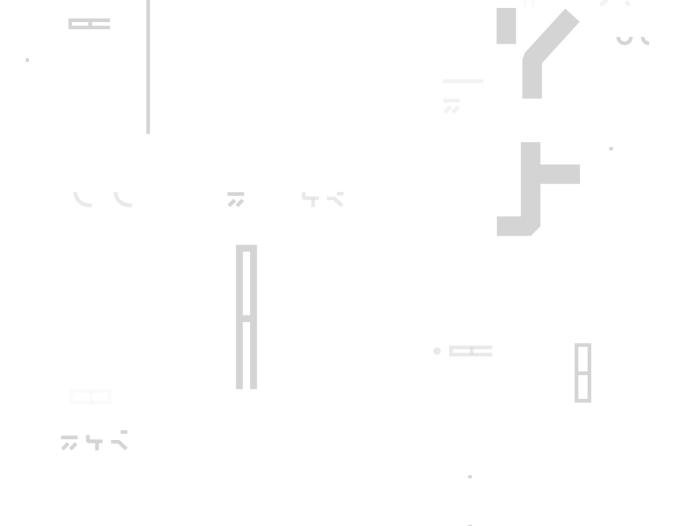


SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT



Customer: Covalent

Date: January 10th, 2022



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 Covalent.		
Approved by	Andrew Matiukhin CTO Hacken OU		
Туре	Staking		
Platform	Ethereum / Solidity		
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review		
Repository	https://github.com/covalenthq/Hacken-Covalent-Staking-Contract		
Commit	5f522eb8299c78ebd0bbbf8287441c95ec0e5ab0		
Technical	YES		
Documentation			
JS tests	YES		
Website	covalenthq.com		
Timeline	01 DECEMBER 2021 - 10 JANUARY 2022		
Changelog	06 DECEMBER 2021 - INITIAL AUDIT		
	14 DECEMBER 2021 - Second Review		
	10 JANUARY 2022 - THIRD REVIEW		

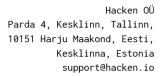




Table of contents

Introduction	
Scope	4
Executive Summary	6
Severity Definitions	7
Audit overview	8
Conclusion	10
Disclaimers	11



Introduction

Hacken OÜ (Consultant) was contracted by Covalent (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contract and its code review conducted between December 1st, 2021, 2021 - December 6th, 2021.

Second review conducted on December 14th, 2021.

Third review conducted on January 10th, 2022.

Scope

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

Repository:

https://github.com/covalenthq/Hacken-Covalent-Staking-Contract

Commit:

5f522eb8299c78ebd0bbbf8287441c95ec0e5ab0

Technical Documentation: Yes

- docs: https://github.com/covalenthq/Hacken-Covalent-Staking-Contract/blob/master/README.md

- wp:

https://www.covalenthq.com/static/documents/Covalent%20Whitepaper%20A
pr%202021%20v1%20Branded.pdf)

JS tests: Yes (included: https://github.com/covalenthq/Hacken-Covalent-Staking-Contract/blob/master/test)

Contracts:

DelegatedStaking.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	
Tunctional Teview	Business Logics Review
	Functionality Checks
	Access Control & Authorization
	Escrow manipulation
	Token Supply manipulation
	Assets integrity
	User Balances manipulation
	 Data Consistency manipulation
	Kill-Switch Mechanism
	Operation Trails & Event Generation



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

As a result of the audit, security engineers found 1 medium and 1 low severity issue.

After the second review security engineers found 1 medium severity issue.

For the third review, security engineers modified the config to change the forking URL to the own alchemy URL to use the archive state node, therefore still 1 medium severity issue is not resolved.



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		



Audit overview

■ ■ ■ Critical

No critical issues were found.

High

No high severity issues were found.

■ Medium

Tests are failing.

4 tests out of 60 are failing. Two of them are failing because of timeout of 20 seconds is reached, but two of them because of unexpected return values.

- allocatedTokensPerEpoch is changed only by calling "setAllocatedTokensPerEpoch" directly, while test "Should emit redeem reward event with correct number of rewards" expects this to be a sum of "rewardsAvailable" and "commissionRewards" divided by 100
- in the "Should return number of rewards earned by validator with delegators" it looks like the "getValidatorsDetails" was mistakenly called with validator's address provided.

Contracts: DelegatedStaking.sol

Tests: redeemAllRewards.js

Recommendation: Please either correct tests or fix the code's logic to make sure all tests are passed.

Status: Running "redeemAllRewards" separately, as proposed by the customer, still one test is failing:

- 1 failing
 - 1) Redeem Rewards

Should return number of rewards earned by validator with delegators: ProviderError: project ID does not have access to archive state

Status update: Changing the forking URL to the alchemy node changed the result, not one test is still failing but the error is different:



Low

A public function that could be declared external

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

Contracts: DelegatedStaking.sol

Function: initialize, depositRewardTokens, takeOutRewardTokens, stake, unstake, recoverUnstaking, addValidator, disableValidator, setAllocatedTokensPerEpoch, setMaxCapMultiplier, setValidatorCommissionRate, setValidatorMinStakedRequired, redelegateUnstaked, transferUnstakedOut

Recommendation: Use the **external** attribute for functions never called from the contract.

Status: Fixed



Conclusion

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

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

As a result of the audit, security engineers found 1 medium and 1 low severity issue.

After the second review security engineers found 1 medium severity issue.

For the third review, security engineers modified the config to change the forking URL to the own alchemy URL to use the archive state node, therefore still 1 medium severity issue is not resolved.



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 the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free 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 the security of smart contracts.

Technical Disclaimer

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