

# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

This document may contain confidential information about IT systems and the intellectual property of the Customer and 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 fixed - upon a decision of the Customer.

## Document

<b>Name</b>	Smart Contract Code Review and Security Analysis Report for Layerx (10 pages)
<b>Approved by</b>	Andrew Matiukhin   CTO Hacken OU
<b>Type</b>	Launchpad
<b>Platform</b>	Ethereum / Solidity
<b>Methods</b>	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
<b>Contract address</b>	<a href="https://etherscan.io/address/0xe39F1a62AabcFAC847E412a4CC4Db3Dde61c002E">https://etherscan.io/address/0xe39F1a62AabcFAC847E412a4CC4Db3Dde61c002E</a>
<b>Contract for remediation</b>	<a href="https://github.com/UNILAYER/SMARTCONTRACT/blob/main/launchpad">HTTPS://GITHUB.COM/UNILAYER/SMARTCONTRACT/blob/main/launchpad</a>
<b>Commit for remediation</b>	ED5FBDD7534989D30CCFB2B5B63DE385324E1F32
<b>Timeline</b>	9 DEC 2020 - 10 DEC 2020
<b>Changelog</b>	10 DEC 2020 - Initial Audit 10 DEC 2020 - Remediation check



## Table of contents

Introduction.....	4
Scope.....	4
Executive Summary.....	5
Severity Definitions.....	6
Conclusion.....	9
Disclaimers.....	10

## Introduction

Hacken OÜ (Consultant) was contracted by Unilayer (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 9<sup>th</sup>, 2020 - December 10<sup>th</sup>, 2020.

## Scope

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

Contract address:

<https://etherscan.io/address/0xe39F1a62AabcFAC847E412a4CC4Db3Dde61c002E>

Contract provided for remediation test:

[HTTPS://GITHUB.COM/UNILAYER/SMARTCONTRACT/BLOB/MAIN/LAUNCHPAD](https://github.com/unilayer/smartcontract/blob/main/launchpad)

Commit: ED5FBDD7534989D30CCFB2B5B63DE385324E1F32

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

Functional review	<ul style="list-style-type: none"> <li>■ Business Logics Review</li> <li>■ Functionality Checks</li> <li>■ Access Control &amp; Authorization</li> <li>■ Escrow manipulation</li> <li>■ Token Supply manipulation</li> <li>■ Assets integrity</li> <li>■ User Balances manipulation</li> <li>■ Data Consistency manipulation</li> <li>■ Kill-Switch Mechanism</li> <li>■ Operation Trails &amp; Event Generation</li> </ul>
-------------------	---

## Executive Summary

According to the assessment, the Customer's smart contracts are required minor improvements.

**Note:** At a moment of remediation check, most of the issue was addressed.

We described issues in the conclusion of these documents. Please read the whole document to estimate the risks well.

Insecure	Poor secured	Secured	Well-secured
----------	--------------	---------	--------------



You are here<sup>1</sup>

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.

During initial audit security engineers found 1 critical, 2 medium, 1 low and 2 lowest severity issue during the audit.

<sup>1</sup> Look for details and justification in Audit review and conclusion sections

## 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 essential 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

1. *setDetails* function has no access modifiers, thus, any account can change all parameters set in the function, including token total supply, token name etc.

It's critical issue, for example, attacker can constantly change token name to "scam token" or set total supply to 0

```
173     function setDetails(  
174         string memory tokenName,  
175         string memory siteUrl,  
176         string memory paperUrl,  
177         string memory twitterUrl,  
178         string memory telegramUrl,  
179         string memory mediumUrl,  
180         string memory gitUrl,  
181         string memory discordUrl,  
182         string memory tokenDesc,  
183         uint256 tokensForSale,  
184         uint256 minContribution,  
185         uint256 maxContribution,  
186         uint256 tokenTotalSupply,  
187         bool uniListing,  
188         bool tokenMint  
189     ) external {  
190         _tokenName = tokenName;  
191         _siteUrl = siteUrl;  
192         _paperUrl = paperUrl;  
193         _twitterUrl = twitterUrl;  
194         _telegramUrl = telegramUrl;  
195         _mediumUrl = mediumUrl;  
196         _gitUrl = gitUrl;  
197         _discordUrl = discordUrl;  
198         _tokenDesc = tokenDesc;  
199         _tokensForSale = tokensForSale;  
200         _minContribution = minContribution;  
201         _maxContribution = maxContribution;  
202         _uniListing = uniListing;  
203         _tokenMint = tokenMint;  
204         _tokenTotalSupply = tokenTotalSupply;  
205     }
```

**State:** Fixed at a moment of remediation test

### High

No high issues were found.

### Medium

1. Raised ETH may be different from `_raisedETH` value. If `_raisedETH >= _maxCap`, the difference will not be deducted from `_raisedETH` and it'll be different from final ETH balance

**State:** Fixed at a moment of remediation test

2. If project cap not reached claim function will send the tokens to participant in any case

**State:** Fixed at a moment of remediation test

#### ■ Low

1. Receive function name doesn't describe it's functionality - it should have better name, for example, sendEthToLaunchpad

**State:** Fixed at a moment of remediation test

#### ■ Lowest / Code style / Best Practice

1. It's recommended for all getters to start with "get" for cleaner code style, for example, getBalanceToClaim or getBalanceToClaimTokens
2. Code is partially documented in comments, it's recommended to add comments to all functions For example, balanceToClaimTokens function comments describe function with parameters and return value; setDetails function has no comments.



## 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.

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

Security engineers found 1 critical, 2 medium, 1 low and 2 lowest severity issues during the initial audit.

Violations in the following categories were found and addressed to the Customer:

Category	Check Item	Comments
Code review	<ul style="list-style-type: none"><li>Style and Naming guide violation</li></ul>	Recommended to follow best practices.

## 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, 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 the 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 own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.