



UniLayer

LAYERx Farm

SMART CONTRACT AUDIT

01.05.2021

Made in Germany by Chainsulting.de



Table of contents

1. Disclaimer	4
2. About the Project and Company	5
2.1 Project Overview	6
3. Vulnerability & Risk Level.....	7
4. Auditing Strategy and Techniques Applied	8
4.1 Methodology.....	8
4.2 Used Code from other Frameworks/Smart Contracts	9
4.3 Tested Contract Files.....	10
4.4 Metrics / CallGraph	11
4.5 Metrics / Source Lines	12
4.6 Metrics / Capabilities.....	13
4.7 Metrics / Source Unites in Scope.....	13
5. Scope of Work.....	14
5.1 Manual and Automated Vulnerability Test	15
5.1.1 Pragma version not fixed	16
5.2. SWC Attacks & Special Checks	17
5.3. Testnet Deployment.....	21
5.3.1 Deployment LAYER token contract	21
5.3.2 Deployment LAYERx contract	22
5.3.3 Default contract variables	23
5.3.4 Reducing staking period for testing.....	24
5.3.5 Approve LAYERx contract to spend token	24



5.3.6 Lock tokens for staking	25
5.3.7 Approve LAYER token contract for staking	26
5.3.8 Lock tokens for staking	26
5.3.9 Unlock staked tokens from pool	27
5.3.10 Check stake count variable	27
5.3.11 Finalize the staking by closing without ether rewards.....	28
5.3.12 Send Ether to contract	29
5.3.13 Finalize staking with ether rewards.....	29
5.3.14 Withdraw LAYERx tokens and ether	30
5.3.15 Total stakes count.....	30
5.4 Verify Claims.....	31
6. Executive Summary	32
7. Deployed Smart Contract.....	33



1. Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of UniLayer. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (15.02.2021)	Layout
0.5 (16.02.2021)	Automated Security Testing Manual Security Testing
0.6 (17.02.2021)	Testing SWC Checks
0.7 (17.02.2021)	Verify Claims
0.9 (18.02.2021)	Summary and Recommendation
1.0 (19.02.2021)	Final document
1.1 (01.05.2021)	Deployed contract

2. About the Project and Company

Company address:

The Unilayer Project LTD
44 Church St.
St. John's
Antigua & Barbuda

Website: <https://unilayer.info>

DApp: <https://unilayer.app>

Twitter: https://twitter.com/unilayer_

Telegram: <https://t.me/Unilayer>

Etherscan (LAYER Token): <https://etherscan.io/token/0x0fF6fcFDa92c53F615a4A75D982f399C989366b>

Medium: <https://medium.com/@UniLayer/unilayer-next-generation-decentralised-trading-platform-524e458ec7ff>

Discord: <https://discord.com/invite/BV5y3dd>

2.1 Project Overview

UniLayer is a next generation decentralised trading platform built on top of Uniswap that enables key features for professional-level trading with it's LAYER utility token, focusing on automated swaps and liquidity management, flash staking, charts and analytics, live order books, and a lot more. On top of these features, the LAYER token is used to facilitate transactions on UniLayer where all transaction fees are transferred to a token pool. 92% of fees will be distributed to stakers of the platform in addition to liquidity providers to the ETH/LAYER liquidity pool, with the remaining 8% going to the foundation as a reserve.

What is LAYERx?

A new token that will have 0 premine, governance features integrated in the Unilayer platform, and can only be minted while staking LAYER in the staking platform or by Providing liquidity on Uniswap.

LAYERx Tokenomics

- Total Supply 40,000
- Inflation rate 10,000 year for a period of 4 years
- Can be minted while staking LAYER in the staking platform
- Can be minted while providing Liquidity to LAYERX/ETH LPs in Uniswap

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
 - i. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the smart contract.
 - ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 - iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Chainsulting describe.
2. Testing and automated analysis that includes the following:
 - i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
 - ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

4.2 Used Code from other Frameworks/Smart Contracts (direct imports)

1. SafeMath.sol (0.5.0)

<https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.3.0/contracts/math/SafeMath.sol>

2. IERC20.sol (0.5.0)

<https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.3.0/contracts/token/ERC20/IERC20.sol>

3. Address.sol (0.5.0)

<https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.3.0/contracts/utils/Address.sol>

4. Ownable.sol (0.5.0)

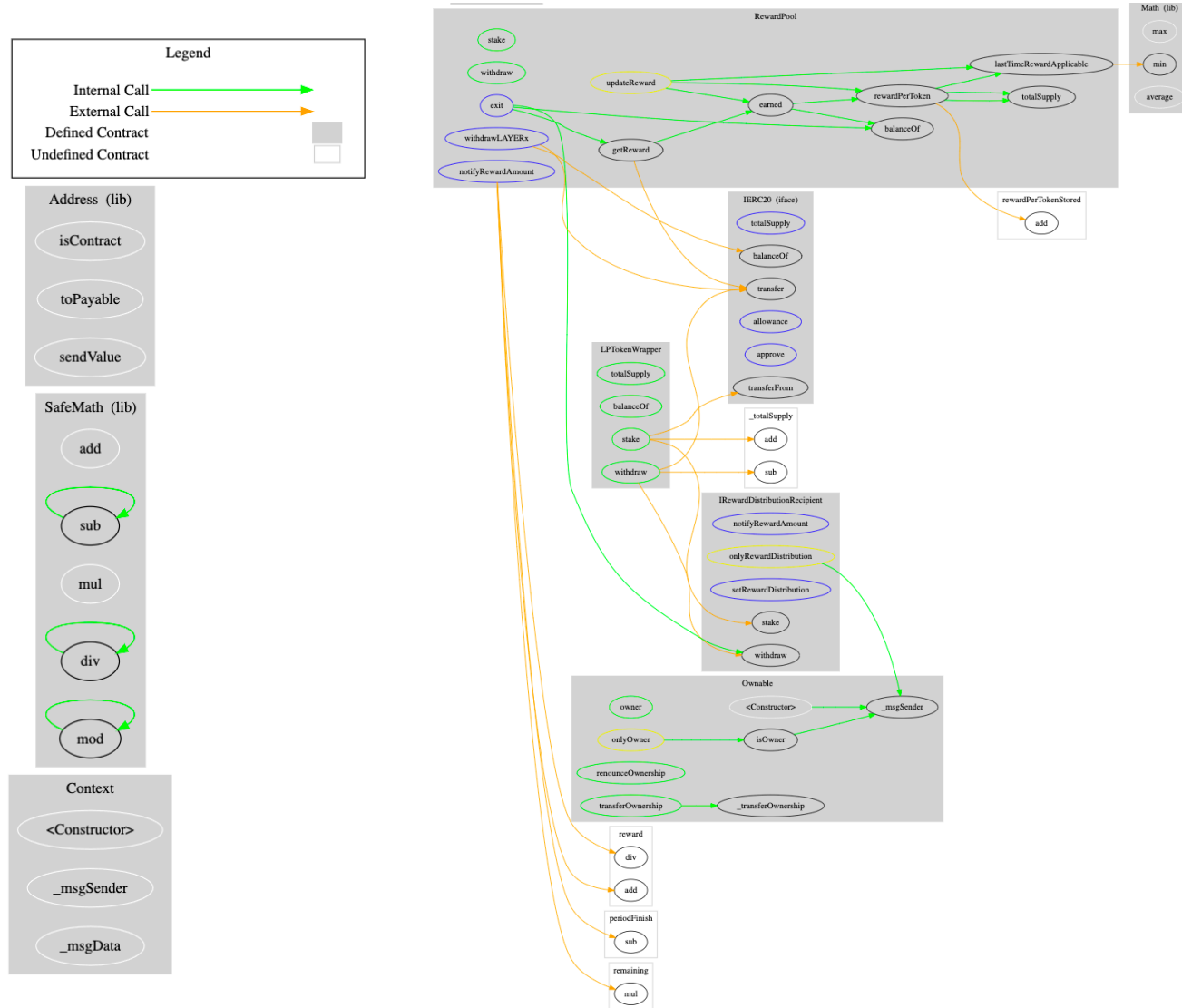
<https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.3.0/contracts/ownership/Ownable.sol>

4.3 Tested Contract Files

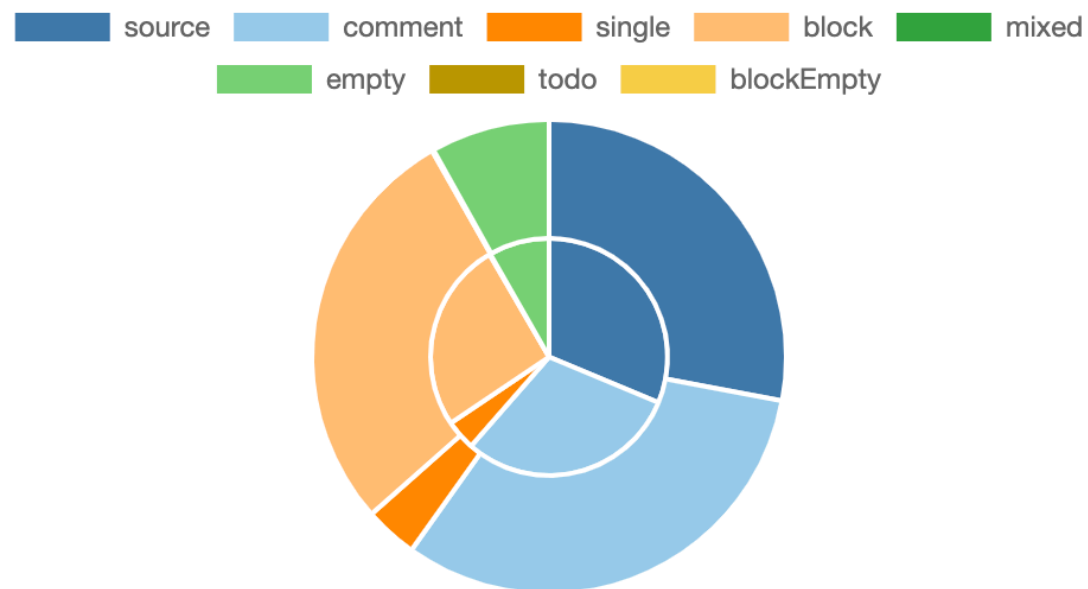
The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5)
Farm.sol	2b58a25c45301ed9de282e2c2f7c8da5











4.4 Metrics / CallGraph







4.5 Metrics / Source Lines



4.6 Metrics / Capabilities

Solidity Versions observed		 Experimental Features		 Can Receive Funds		 Uses Assembly		 Has Destroyable Contracts			
<div>^0.5.0</div> <div>^0.5.5</div>						<div>yes</div> <div>(1 asm blocks)</div>					
 Transfers ETH		 Low-Level Calls		 DelegateCall		 Uses Hash Functions		 ECRrecover		 New/Create/Create2	
<div>yes</div>		<div>yes</div>									

4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	farm.sol	8	1	619	556	250	291	187	
	Totals	8	1	619	556	250	291	187	

5. Scope of Work

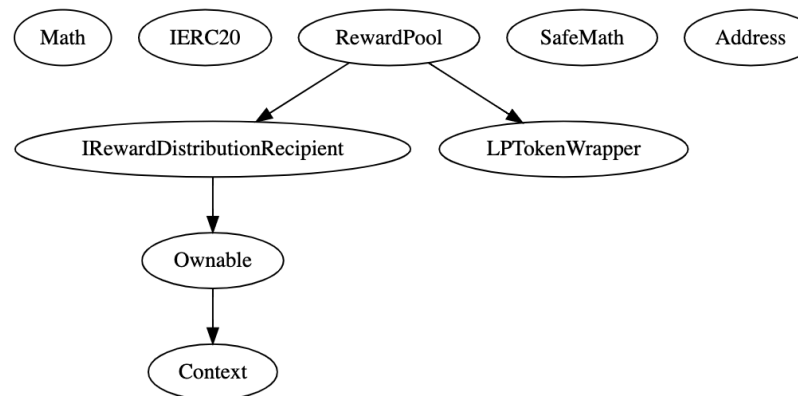
The Unilayer team provided us with the files that needs to be tested. The scope of the audit is the LAYERx farm contract.

Following contracts with the direct imports been tested
farm.sol

The team put forward the following assumptions regarding the security, usage of the contracts:

- User is able to withdraw staked LP Token
- User is able to withdraw rewards
- Emergency withdraw is working
- Deployer cannot steal any user funds
- Deployer cannot burn or lock user funds
- Deployer cannot pause the contract

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.



5.1 Manual and Automated Vulnerability Test

CRITICAL ISSUES

During the audit, Chainsulting's experts found **no Critical issues** in the code of the smart contract.

HIGH ISSUES

During the audit, Chainsulting's experts found **no High issues** in the code of the smart contract.

MEDIUM ISSUES

During the audit, Chainsulting's experts found **no Medium issues** in the code of the smart contract.

LOW ISSUES

During the audit, Chainsulting's experts found **no Low issues** in the code of the smart contract.

INFORMATIONAL ISSUES

5.1.1 Pragma version not fixed

Severity: INFORMATIONAL

Status: **ADDRESSED**

Update: Doesn't affect the overall security

File(s) affected: farm.sol

Attack / Description	Code Snippet	Result/Recommendation
Due to the fact that compiler upgrades might bring unexpected compatibility or inter-version consistencies, it is always suggested to use fixed compiler versions whenever possible.	Line: 1 <code>pragma solidity ^0.5.0;</code>	As an example, we highly encourage to explicitly indicate the Solidity compiler version, e.g., <code>pragma solidity 0.5.0;</code> instead of <code>pragma solidity ^0.5.0;</code> .

5.2. SWC Attacks & Special Checks

ID	Title	Relationships	Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	

ID	Title	Relationships	Test Result
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	✓
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	✓
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	✓
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	✓
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	✓
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	✓
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✓
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	✓
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	✓

ID	Title	Relationships	Test Result
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✓
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✓
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✓
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✓
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✓
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✓
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✓
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✗
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✓
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✓

ID	Title	Relationships	Test Result
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	✗
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	✓
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	✓
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	✓

5.3. Testnet Deployment

5.3.1 Deployment LAYER token contract

CONTRACT

Unilayer - browser/contracts/1_Layer.sol

DEPLOY


NAME: Unilayer

SYMBOL: LAYER

DECIMALS: 18

TOTALSUPPLY: 40000000

OWNER: 0x2F1602FD37228b32Ad8D13137b18620862771909

 **transact**

Tx: <https://ropsten.etherscan.io/tx/0x73f8f3f3b335ef68b35cd5332f372c5a147dbe48082ead5db24221fcb4854cfe>

Contract: [0x21bb8a338b68fe0e86eabccadc65e97a91520ef57](https://ropsten.etherscan.io/address/0x21bb8a338b68fe0e86eabccadc65e97a91520ef57)

5.3.2 Deployment LAYERx contract

CONTRACT

Layerx - browser/contracts/2_Layerx.sol

DEPLOY

_OWNER: 0x2F1602FD37228b32Ad8D13137b1B620862771909





LAYER_TOKEN: 0x21bb8a338b68fe0e86eabccadc65e97a91520ef57

transact

Tx: <https://ropsten.etherscan.io/tx/0x322d57158d509eb1301d9fc288e38e47c7a47c8d0abbaa05f984ae58a15a6fb8>

Contract: [0x5da2bbc48aa50421bca0a7d164c15c97ca4e1b56](https://ropsten.etherscan.io/address/0x5da2bbc48aa50421bca0a7d164c15c97ca4e1b56)


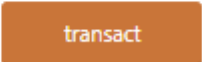
5.3.3 Default contract variables

Address	 0x5Da2bbc48aa50421bcA0a7D164c15C97cA4E1B56
ETH Balance	0
UNILAYER	 0x21bB8A338b68fE0E86eabcADc65E97a91520ef57
decimals	18
isPaused	X False
name	UNILAYERX
owner	 0x2F1602FD37228b32Ad8D13137b1B620862771909
stakeCreator	 0x2F1602FD37228b32Ad8D13137b1B620862771909
stakePeriod	604800
symbol	LAYERX
totalEthRewards	0
totalSupply	4000000000000000000000
getStakesNum	1

5.3.4 Reducing staking period for testing

setStakePeriod

newStakePeriod:

stakePeriod

0: uint256: 30


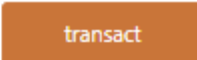
Tx: <https://ropsten.etherscan.io/tx/0x54059eaf8e4ac55a601488647a1536ec27c8700c0b66708f693e0b33c3f59b4f>

5.3.5 Approve LAYERx contract to spend token

approve

spender:

tokens:



 

Tx: <https://ropsten.etherscan.io/tx/0x4e54c241f10e69b364df8a4cb2e34db30a5cf546a204d9d6f151aaa5d092d83b>

allowance

tokenOwner:

spender:

0: uint256: remaining 2000000000000000000

5.3.6 Lock tokens for staking

lock

```
amount: 1000000000000000000
```


transact

Gas estimation failed

x

Gas estimation errored with the following message (see below). The transaction execution will likely fail. Do you want to force sending?

```
execution reverted: Call Approve function firstly. { "originalError": { "code": 3,
```

"data":

[illegible]

Send Transaction

Cancel Transaction



5.3.7 Approve LAYER token contract for staking

approve

spender: 0x5da2bbc48aa50421bca0a7d164c15c97ca4e1b56

value: 3000000000000000000



transact

Tx: <https://ropsten.etherscan.io/tx/0xe84906b5fef5231c9f2430355bb5e7944e9bf96a45bf6437c56386ceceb673f9>

5.3.8 Lock tokens for staking

lock

amount: 1000000000000000000



transact

Tx: <https://ropsten.etherscan.io/tx/0x0902e976ccaeab98ff03a4d945f6f8340667ae471ad5e52395f07cdc14c799e9>

stakeOf

account: "0x2F1602FD37228b32Ad8D13137b1B620862771909"



call

0: uint256: 1000000000000000000

5.3.9 Unlock staked tokens from pool

stakeOf

account: "0x2F1602FD37228b32Ad8D13137b1B620862771909"



call

0: uint256: 0

Tx: <https://ropsten.etherscan.io/tx/0x13ccd9e863739478a4e7b529bde4cddc651187ed30381e0725c343a326b261b2>

5.3.10 Check stake count variable

getStakesCount

holder: "0x2F1602FD37228b32Ad8D13137b1B620862771909"



call

0: uint256: 0

lock

amount: 1000000000000000000



transact

Tx: <https://ropsten.etherscan.io/tx/0x475d2ba349c72a866379ab74768c3d9982764d32f4f453cb9259eb37efb12759>

getStakesCount

holder: "0x2F1602FD37228b32Ad8D13137b1B620862771909"



call

0: uint256: 1
stakeOf

account: "0x2F1602FD37228b32Ad8D13137b1B620862771909"



call

0: uint256: 1000000000000000000

5.3.11 Finalize the staking by closing without ether rewards

rewards

0x2F1602FD37228b32Ad8D13137b1B620862771909

0: uint256: layerx 836187214612708332

1: uint256: eth 0

Tx: <https://ropsten.etherscan.io/tx/0xaa1040ded46d95cba6f70c295d717bd47c0f8e1281da373770265ab495993a64>



5.3.12 Send Ether to contract

Sending 0.1 Ether to the Layerx smart contract for staking rewards.

totalEthRewards

0: uint256: 1000000000000000000

Tx: <https://ropsten.etherscan.io/tx/0x86b08161d7384f9e45dcfe7a4d8d9b413e1ab8e1e956b9be5210808b118a3306>

5.3.13 Finalize staking with ether rewards

rewards

0x2F1602FD37228b32Ad8D13137b18620862771909

0: uint256: layerx 1268391679351851849

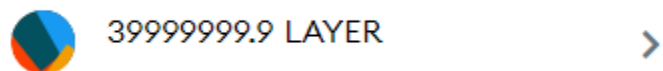
1: uint256: eth 999999999999999999

totalEthRewards

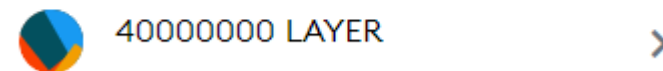
0: uint256: 0

Tx: <https://ropsten.etherscan.io/tx/0x6407d2cdcf55d394fdf1081076e47bd89db95145e484e025a491c13fa2153a91>

5.3.14 Withdraw LAYERx tokens and ether



Tx: <https://ropsten.etherscan.io/tx/0x2a020f1a09d7a962fee697fc86104d855bc0087b6a6e91d73b45faa133f7c139>



5.3.15 Total stakes count



0: uint256: 3

5.4 Verify Claims

5.4.1 **User is able to withdraw staked LP Token** ✓

Status: tested and verified

5.4.2 **User is able to withdraw rewards** ✓

Status: tested and verified

5.4.3 **Emergency withdraw is working** ✓

Status: tested and verified

5.4.4 **Deployer cannot steal any user funds** ✓

Status: tested and verified

5.4.5 **Deployer cannot burn or lock user funds** ✓

Status: tested and verified

5.4.6 **Deployer cannot pause the contract** ✓

Status: tested and verified

6. Executive Summary

The overall code quality of the project is good, not overloaded with unnecessary functions, these is greatly benefiting the security of the contract. It has not correctly implemented widely-used and reviewed contracts from OpenZeppelin and for safe mathematical operations and we recommend as well, to specific all integer types.

The main goal of the audit was to verify the claims regarding the security of the smart contract and the functions. During the audit, no issues were found after the manual and automated security testing.

7. Deployed Smart Contract

VERIFIED

Farm

<https://etherscan.io/address/0xB238d2407134f14E1fB8a9E449c6c5D0e88CD04D#code>