

Smart Contract Security Audit V1

Final Staking V3 Smart Contract Audit

JAN 5, 2025



<https://saferico.com/>

business@saferico.com

—

Table of Contents

Table of Contents

Background

Project Information

Smart Contract Information

Executive Summary

File and Function Level Report

File in Scope:

Issues Checking Status

SWC Attack Analysis

Severity Definitions

Audit Findings

Automatic testing

Testing proves

Inheritance graph

Call graph

Source lines

Risk level

Source units in scope

Capabilities

Unified Modeling Language (UML)

Automatic general report

Conclusion

Disclaimer

Background

The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Project Information

- **Platform:** Base Network
- **Name:** Final Staking V3
- **Language :** Solidity
- **Contract Address:** 0xe935fab9fd068039ad1ecce125ec7c2e3f052068
- **Code Source:** <https://sepolia.basescan.org/address/0xe935fab9fd068039ad1ecce125ec7c2e3f052068#code>

Executive Summary

According to our assessment, the customer`s solidity smart contract is **Well-Secured**.

Well Secured	✓
Secured	
Poor Secured	
Insecure	

Automated checks are with remix IDE. All issues were performed by the team, which included the analysis of code functionality, manual audit found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the audit overview section. The general overview is presented in the Project Information section and all issues found are located in the audit overview section.

Team found 0 critical, 0 high, 0 medium, 1 low, 0 very low-level issues and 0 note in all solidity files of the contract

The files:

FinalStakingV3.sol

Audit Score:

99% secure



File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
FinalStakingV3.sol	69ede4562a6c5c2f3638d8f4354855081e9d8d74	0xe935fab9fd068039ad1ecce125ec7c2e3f052068

- Contract: FinalStakingV3
- Inherit: Ownable
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
_calRem	✓	Read / public	Passed
afStk	✓	Read / public	Passed
aiTreasury	✓	Read / public	Passed
checkUsersWithRank	✓	Read / public	Passed
dailyYield	✓	Read / public	Passed
DAI	✓	Read / public	Passed
DAY	✓	Read / public	Passed
getPFromUni	✓	Read / public	Passed
getPFromUniswapV2	✓	Read / public	Passed
getTmUsrLvlYc	✓	Read / public	Passed
getUserTeam	✓	Read / public	Passed
getTx	✓	Read / public	Passed
getUsrAfInc	✓	Read / public	Passed
getUsrDiv	✓	Read / public	Passed

getUsrDivLst	✓	Read / public	Passed
getUsrStk	✓	Read / public	Passed
getUsrStkCnt	✓	Read / public	Passed
isElgMnRwrld	✓	Read / public	Passed
isSupportedToken	✓	Read / public	Passed
isUserExists	✓	Read / public	Passed
LST_LVL	✓	Read / public	Passed
lstRstTmstmpMth	✓	Read / public	Passed
lstRstTmstmpMthOld	✓	Read / public	Passed
lstStkAmnt	✓	Read / public	Passed
mthBusClm	✓	Read / public	Passed
mtr	✓	Read / public	Passed
mult	✓	Read / public	Passed
owner	✓	Read / public	Passed
PCT_DIV	✓	Read / public	Passed
rAch	✓	Read / public	Passed
rAchOld	✓	Read / public	Passed
ror	✓	Read / public	Passed
rI	✓	Read / public	Passed
rorX	✓	Read / public	Passed
rT	✓	Read / public	Passed
rUC	✓	Read / public	Passed
totStkd	✓	Read / public	Passed
t_o	✓	Read / public	Passed
totWthdrwl	✓	Read / public	Passed
treasury	✓	Read / public	Passed
tU	✓	Read / public	Passed
uC	✓	Read / public	Passed
uniswapRt	✓	Read / public	Passed

uRR	✓	Read / public	Passed
uS	✓	Read / public	Passed
uS2	✓	Read / public	Passed
usdX	✓	Read / public	Passed
usrent	✓	Read / public	Passed
wmr	✓	Read / public	Passed
WPOL	✓	Read / public	Passed
yidkg	✓	Read / public	Passed
renounceOwnership	✓	Write / public	Passed
rnkRwd	✓	Write / public	Passed
stake	✓	Write / payable	Passed
supportedToken	✓	Write / public	Passed
transferOwnership	✓	Write / public	Passed
updRwd	✓	Write / public	Passed
wdrLvl	✓	Write / public	Passed
wdrStk	✓	Write / public	Passed

Issues Checking Status

SWC Attack Analysis

The Smart Contract Weakness Classification Registry (SWC Registry) is an implementation of the weakness classification scheme proposed in EIP-1470. It is loosely aligned to the terminologies and structure used in the Common Weakness Enumeration (CWE) for more info check

<https://swcregistry.io/>

No.	Issue Description	Checking Status
136	Unencrypted Private Data On-Chain	Passed
135	Code With No Effects	Passed
134	Message call with hardcoded gas amount	Passed
133	Hash Collisions With Multiple Variable Length Arguments	Passed
132	Unexpected Ether balance	Passed
131	Presence of unused variables	Passed
130	Right-To-Left-Override control character (U+202E)	Passed
129	Typographical Error	Passed
128	DoS with block gas limit.	Passed
127	Arbitrary Jump with Function Type Variable	Passed
126	Insufficient Gas Griefing	Passed
125	Incorrect Inheritance Order	Passed
124	Write to Arbitrary Storage Location	Passed
123	Requirement Violation	Passed
122	Lack of Proper Signature Verification	Passed
121	Missing Protection against Signature Replay Attacks	Passed
120	Weak Sources of Randomness from Chain Attributes	Passed
119	Shadowing State Variables	Passed

118	Incorrect Constructor Name	Passed
117	Signature Malleability	Passed
116	Block values as a proxy for time	Not Passed
115	Authorization through tx.origin	Passed
114	Transaction Order Dependence	Passed
113	DoS with Failed Call	Passed
112	Delegatecall to Untrusted Callee	Passed
111	Use of Deprecated Solidity Functions	Passed
110	Assert Violation	Passed
109	Uninitialized Storage Pointer	Passed
108	State Variable Default Visibility	Passed
107	Reentrancy	Passed
106	Unprotected SELFDESTRUCT Instruction	Passed
105	Unprotected Ether Withdrawal	Passed
104	Unchecked Call Return Value	Passed
103	Floating Pragma	Passed
102	Outdated Compiler Version	Passed
101	Integer Overflow and Underflow	Passed
100	Function Default Visibility	Passed

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have 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 tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Note	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Audit Findings

Critical:

No Critical severity vulnerabilities were found.

High:

No High severity vulnerabilities were found.

Medium:

No Medium severity vulnerabilities were found.

Low:

Use of block.timestamp for comparisons

The value of block.timestamp can be manipulated by the miner. And conditions with strict equality is difficult to achieve - block.timestamp.

```
// Operator initialization
    uS[_eco].rwdLmt = type(uint256).max; // More readable than
~uint256(0)
    uS2[_eco].rk = Rnk.EMPIRE_X;
    uS2[_eco].regTime = block.timestamp;
    uC[_eco].mnthRcd = true;

    Ustr2 storage userInfo2 = uS2[_aiBot];
    userInfo2.rk = Rnk.EMPIRE_X;
    userInfo2.regTime = block.timestamp;
    userInfo2.fstStkTm = block.timestamp;
    Stk memory newStake = Stk(1000000e18, block.timestamp);
    userInfo2.stks.push(newStake);
    uS[_aiBot].ref = _eco;
    uS[_aiBot].rwdLmt = type(uint256).max; // More readable
than ~uint256(0)
    uC[_aiBot].mnthRcd = true;
```

Recommendation

Avoid use of block.timestamp.

Status

Acknowledged.

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

No Notes were found.

Automatic Testing

1- SOLIDITY STATIC ANALYSIS

The image shows two side-by-side panels of the Solidity Static Analysis tool. Both panels have a title bar 'SOLIDITY STATIC ANALYSIS' and a 'Run' button. The left panel has checkboxes for 'Select all' and 'Autorun'. It contains two main sections: 'Security' and 'Gas & Economy'. The 'Security' section includes rules like 'Transaction origin', 'Check-effects-interaction', 'Inline assembly', 'Block timestamp', 'Low level calls', 'Block hash', and 'Selfdestruct'. The 'Gas & Economy' section includes 'Gas costs', 'This on local calls', 'Delete dynamic array', 'For loop over dynamic array', and 'Ether transfer in loop'. The right panel has a 'Select ERC' checkbox and an 'ERC20' rule. It also has a 'Select Miscellaneous' checkbox and a list of miscellaneous rules including 'Constant/View/Pure functions', 'Similar variable names', 'No return', 'Guard conditions', 'Result not used', 'String length', 'Delete from dynamic array', and 'Data truncated'.

SOLIDITY STATIC ANALYSIS

☒ Select all ☒ Autorun **Run**

Security

☒ Select Security

- ☒ **Transaction origin:**
'tx.origin' used
- ☒ **Check-effects-interaction:**
Potential reentrancy bugs
- ☒ **Inline assembly:**
Inline assembly used
- ☒ **Block timestamp:**
Can be influenced by miners
- ☒ **Low level calls:**
Should only be used by experienced devs
- ☒ **Block hash:**
Can be influenced by miners
- ☒ **Selfdestruct:**
Contracts using destructed contract can be broken

Gas & Economy

☒ Select Gas & Economy

- ☒ **Gas costs:**
Too high gas requirement of functions
- ☒ **This on local calls:**
Invocation of local functions via 'this'
- ☒ **Delete dynamic array:**
Use require/assert to ensure complete deletion
- ☒ **For loop over dynamic array:**
Iterations depend on dynamic array's size
- ☒ **Ether transfer in loop:**
Transferring Ether in a for/while/do-while loop

SOLIDITY STATIC ANALYSIS

ERC

☒ Select ERC

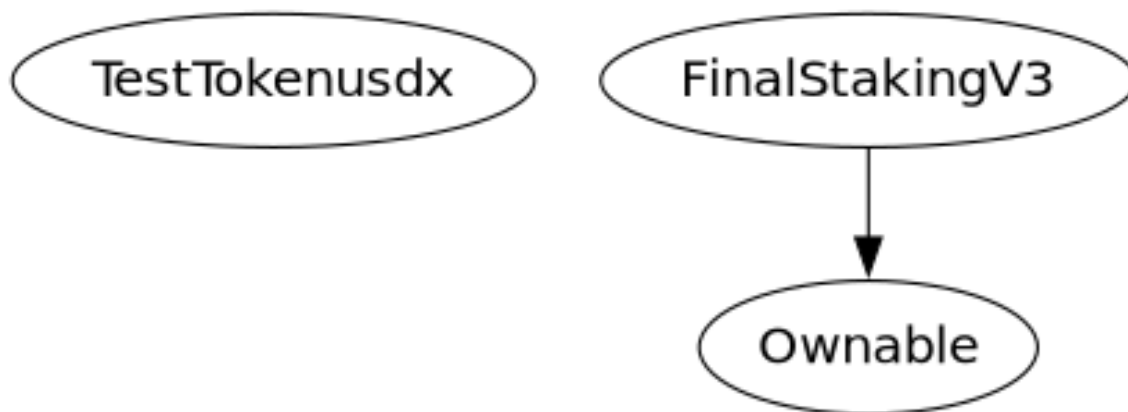
- ☒ **ERC20:**
'decimals' should be 'uint8'

Miscellaneous

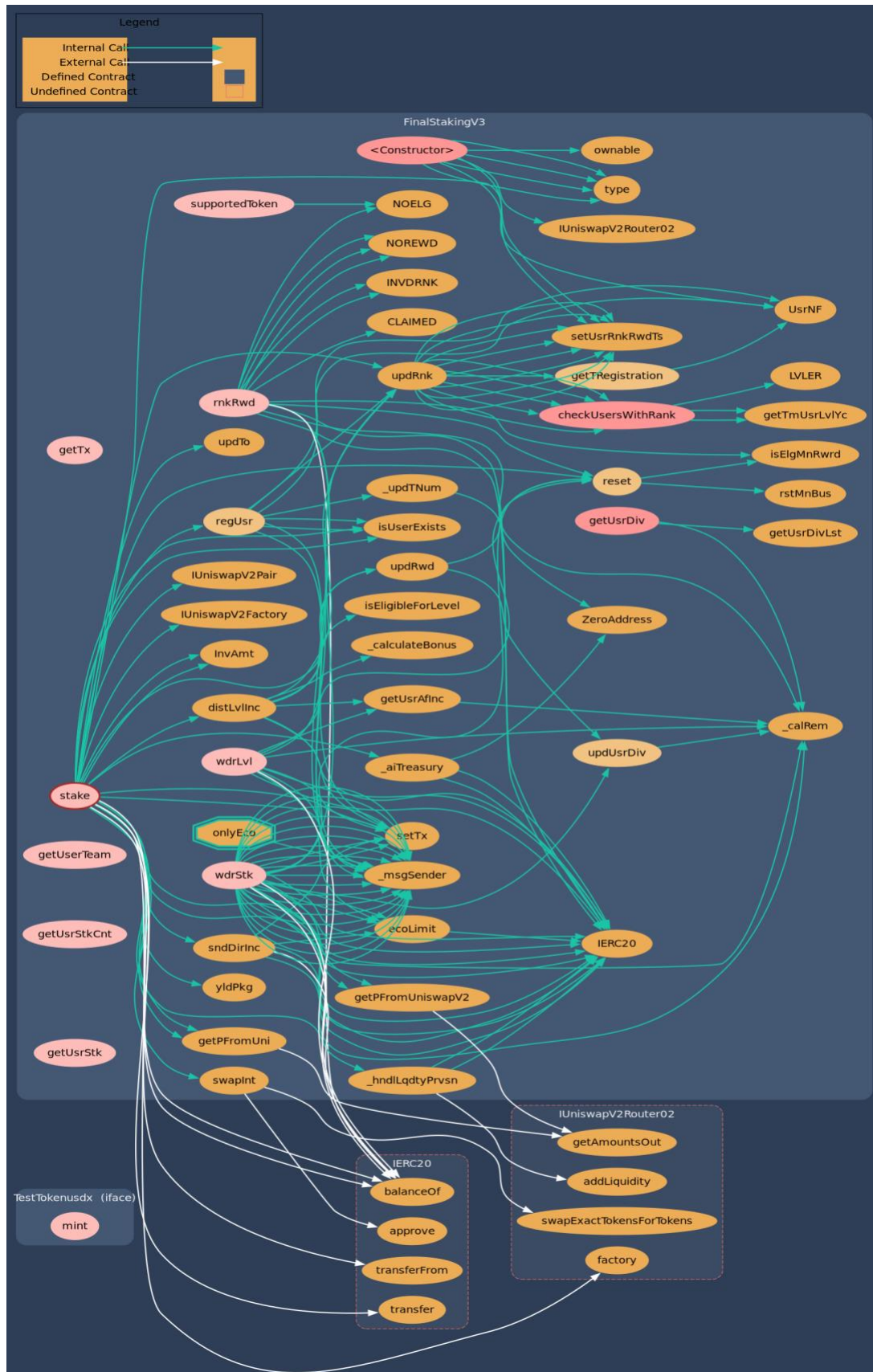
☒ Select Miscellaneous

- ☒ **Constant/View/Pure functions:**
Potentially constant/view/pure functions
- ☒ **Similar variable names:**
Variable names are too similar
- ☒ **No return:**
Function with 'returns' not returning
- ☒ **Guard conditions:**
Ensure appropriate use of require/assert
- ☒ **Result not used:**
The result of an operation not used
- ☒ **String length:**
Bytes length != String length
- ☒ **Delete from dynamic array:**
'delete' leaves a gap in array
- ☒ **Data truncated:**
Division on int/uint values truncates the result

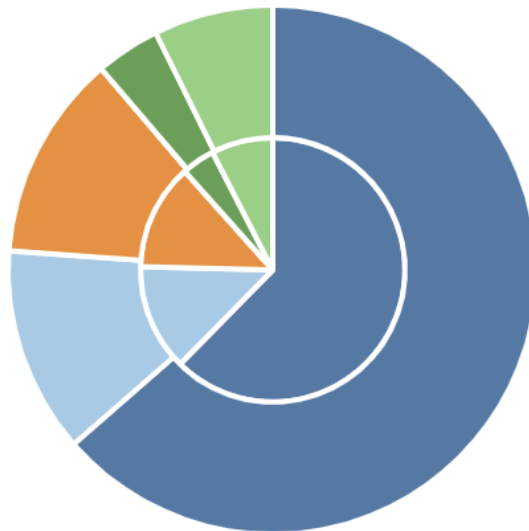
2- Inheritance graph



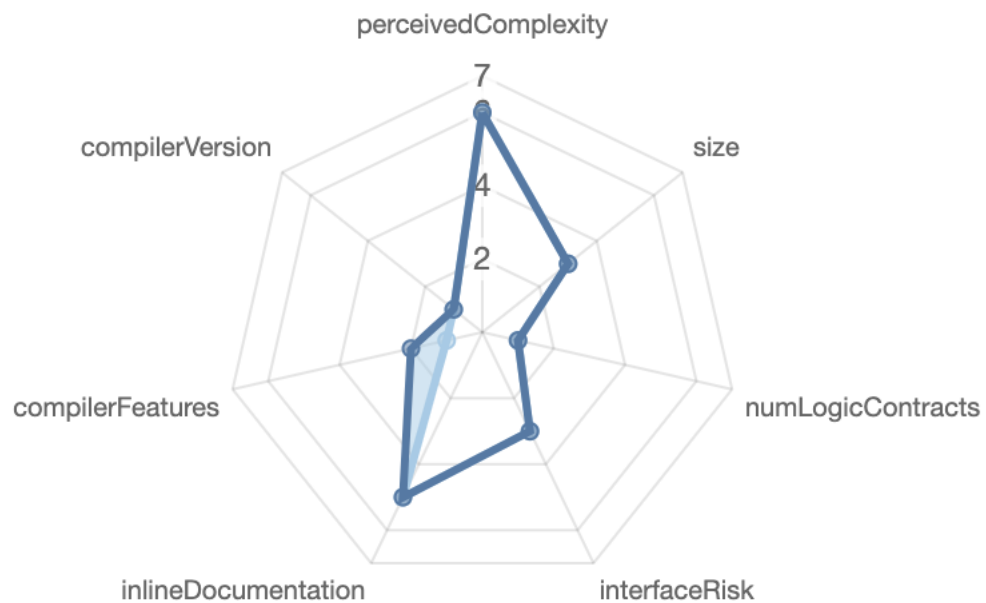
3- Call graph



Source lines



Risk level



Source units in scope

Source Units in Scope

Source Units Analyzed: 1
Source Units in Scope: 1 (100%)

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	FinalStakingV3.sol	1	1	1438	1381	1092	228	636	  
	Totals	1	1	1438	1381	1092	228	636	  

Legend: [-]

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

Capabilities

Components

 Contracts	 Libraries	 Interfaces	 Abstract
1	0	1	0


Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

 Public	 Payable
22	1

External	Internal	Private	Pure	View
10	35	5	0	18

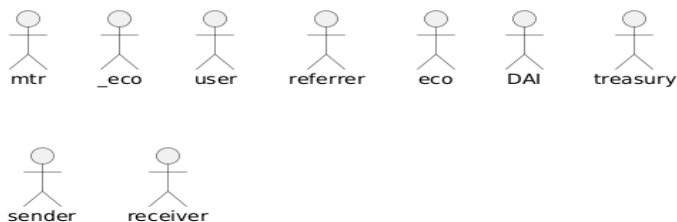
StateVariables

Total	 Public
40	36

Capabilities

Solidity Versions observed	✔ Experimental Features	💰 Can Receive Funds	📜 Uses Assembly	🍏 Has Destroyable Contracts	
0.8.24		yes			
📡 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🏠 Uses Hash Functions	🗑️ ECRrecover	🌀 New/Create/Create2
yes					

Unified Modeling Language (UML)



Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/FinalStakingV3.sol	69ede4562a6c5c2f3638d8f4354855081e9d8d74

Contracts Description Table

Contract	Type	Bases	
:-----: :-----: :-----:			
L	**Function Name**	**Visibility**	**Mutability**
Modifiers			
TestTokenusdx	Interface		
L mint	External !	⬛	NO!
FinalStakingV3	Implementation	Ownable	
L <Constructor>	Public !	⬛	NO!
L setTx	Internal	🔒	⬛
L getTx	External	!	NO!
L supportedToken	External	!	⬛ NO!
L regUsr	Internal	🔒	⬛
L getTRegistration	Internal	🔒	
L reset	Internal	🔒	⬛
L _updTNum	Private	🔒	⬛
L updTo	Internal	🔒	⬛
L rstMnBus	Internal	🔒	⬛
L isElgMnRwrdr	Public	!	NO!
L rnkrwd	External	!	⬛ NO!
L stake	External	!	🔒 NO!
L _hndllqdtPrvsn	Private	🔒	⬛
L _aiTreasury	Private	🔒	⬛
L yldPkg	Public	!	NO!
L getTmUsrLvlYc	Public	!	NO!
L getUserTeam	External	!	NO!
L distLvlInc	Internal	🔒	⬛
L _calculateBonus	Private	🔒	
L isEligibleForLevel	Internal	🔒	
L checkUsersWithRank	Public	!	NO!
L updRnk	Internal	🔒	⬛
L setUsrRnkRwdTs	Internal	🔒	⬛
L sndDirInc	Private	🔒	⬛
L _calRem	Public	!	NO!
L getUsrDivLst	Public	!	NO!
L getUsrDiv	Public	!	NO!
L updUsrDiv	Internal	🔒	⬛

L	updRwd	Public	!	⬛	NO	!
L	wdrStk	External	!	⬛	NO	!
L	wdrLvl	External	!	⬛	NO	!
L	isUserExists	Public	!		NO	!
L	getPFromUniswapV2	Public	!		NO	!
L	getPFromUni	Public	!		NO	!
L	getUshrStkCnt	External	!		NO	!
L	getUshrStk	External	!		NO	!
L	swapInt	Internal	🔒	⬛		
L	ecoLimit	Internal	🔒	⬛		
L	getUshrAfInc	Public	!		NO	!

Legend

Symbol	Meaning
:-----:	-----
⬛	Function can modify state
🔒	Function is payable

Conclusion

The contracts are written systematically. Team found no critical issues. So, it is good to go for production.

Since possible test cases can be unlimited and developer level documentation (code flow diagram with function level description) not provided, for such an extensive smart contract protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan Everything.

Security state of the reviewed contract is “Well Secured”.

- ✓ No volatile code.
- ✓ No high severity issues were found.

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against the team on the basis of what it says or doesn't say, or how team produced it, and it is important for you to conduct your own independent investigations before making any decisions. team go into more detail on this in the below disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Saferico and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Saferico s) owe no duty of care towards you or any other person, nor does Saferico make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Saferico hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Saferico hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Saferico, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.