Smart Contract
Security Audit
V1

LP Management Smart Contract Audit

Jan 12, 2025

https://saferico.com/

business@saferico.com

–

Table of Contents

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**: Ethereum
- **Name**: LPManagement
- **Language** : Solidity
- **Contract Address**: 0x6481313882701c279ff598215B7ba16ae3ED076b
- **Code Source**: https://sepolia.etherscan.io/address/0x6481313882701c279ff598215B7ba16ae3ED076b#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, 3 low, 0 very low-level issues and 1 note in all solidity files of the contract

The files:

LPManagement.sol

Audit Score:

99% secure

99

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
LPManagement.sol	5e5b4e936069ff3e2402ade058d921ca9294b3be	0x6481313882701c279ff598215B7ba16ae3ED076b

- Contract: LPManagement
- Inherit: Pausable, ReentrancyGuard
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
cashCalls	✓	Read / public	Passed
getCashCalls	✓	Read / public	Passed
getETHUSDCExchangeRate	✓	Read / public	Passed
getAdmins	✓	Read / public	Passed
isAdmin	✓	Read / public	Passed
lpData	✓	Read / public	Passed
admins	✓	Read / public	Passed
minCommitmentAmountUSD	✓	Read / public	Passed
defaultAdmin	✓	Read / public	Passed
paused	✓	Read / public	Passed
addAdmin	✓	Write / public	Passed
applyPenalty	✓	Write / public	Passed
createCashCall	✓	Write / public	Passed
executeCashCall	✓	Write / public	Passed

revertExecution	✓	Write / public	**Passed**
makePayment	✓	Write / payable	**Passed**
pause	✓	Write / public	**Passed**
setCommitment	✓	Write / public	**Passed**
setMinCommitmentAmountUSD	✓	Write / public	**Passed**
unPause	✓	Write / public	**Passed**
withdraw	✓	Write / public	**Passed**
removeAdmin	✓	Write / public	**Passed**
setDefaultAdmin	✓	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	**Not 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:

#Admin privileges (In the period when the admin isn’t renounced)

Description

The admin can pause / un pause the smart contract.
The admin set the commitment and the minimum amount of it.
The admin applies penalties for missed deadlines.

```
    function pause() external onlyAdmin {
        _pause();
    }
    // Unpause the contract (Admin only)
    function unpause() external onlyAdmin {
        _unpause();
    }

 function setMinCommitmentAmountUSD(uint256 _minCommitmentAmountUSD) external
 onlyDefaultAdmin {
        require(_minCommitmentAmountUSD > 0, "Minimum commitment amount must be
greater than zero");
        minCommitmentAmountUSD = _minCommitmentAmountUSD;
    }

function applyPenalty(address _lp, uint256 _penaltyAmount, bool _revokeAccess)
external whenNotPaused onlyAdmin {
        LPData storage lpInfo = lpData[_lp];
        require(lpInfo.commitmentAmount > 0, "Invalid LP");

        // Apply late fee
        lpInfo.totalPaid -= _penaltyAmount;
        emit PenaltyApplied(_lp, _penaltyAmount);
        // Revoke access if applicable
        if (_revokeAccess) {
            lpInfo.commitmentAmount = 0;
            lpInfo.totalPaid = 0;
            emit AccessRevoked(_lp); }}
```

Remediation

Make these functions internal in next version or the team should announce the investors before doing anything to give them time if they want to do anything.

P.S: This issue is common to the majority of those smart contracts.

Status: Acknowledged.

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.

```
function setCommitment(
        address _lp,
        uint256 _amountETH,
        uint256 _endTime
    ) external whenNotPaused onlyAdmin {
        require(_lp != address(0), "Invalid LP address");
        require(!isLP(_lp), "LP already exists");
        require(_amountETH * getETHUSDCExchangeRate() >=
minCommitmentAmountUSD * 10**18, "Commitment amount must be greater
than minimum amount");
        require(_endTime > block.timestamp, "End Time must be later
than the current time.");
        LPData storage lpInfo = lpData[_lp];
        lpInfo.commitmentAmount = _amountETH;
        lpInfo.totalPaid = 0;
        lpInfo.endTime = _endTime;
 emit CommitmentSet(_lp, _amountETH, _endTime);
    }
```

Recommendation

Avoid use of block.timestamp.

Status

Acknowledged.

#Pragam version not fixed

Description

It is a good practice to lock the solidity version for a live deployment (use 0.8.28 instead of ^0.8.26). contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs. Contracts may also be deployed by others and the pragma indicates the compiler version intended by the original authors. And avoid Solidity compiler Bugs check here

https://sepolia.etherscan.io/solcbuginfo

Remediation

Remove the ^ sign to lock the pragma version.

Status: Acknowledged.

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

#USE SELFBALANCE() INSTEAD OF ADDRESS(THIS).BALANCE

In Solidity, efficient use of gas is paramount to ensure cost-effective execution on the Ethereum blockchain. Gas can be optimized when obtaining contract balance by using selfbalance() rather than address(this).balance because it bypasses gas costs and refunds, which are not required for obtaining the contract's balance.

```
function withdraw(uint256 _amount, address _to) external onlyAdmin
nonReentrant {
        require(_amount <= address(this).balance, "Insufficient
balance in contract");
        payable(_to).transfer(_amount);
        emit Withdrawal(_to, _amount);
    }
```

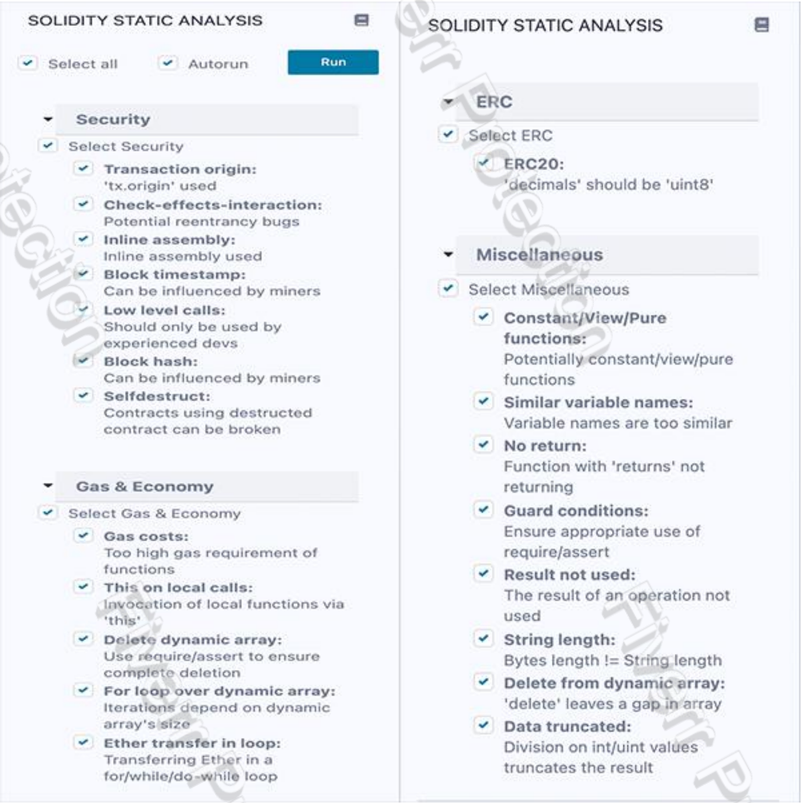
Remediation

To rectify this issue, developers are encouraged to replace instances of address(this).balance with selfbalance() wherever applicable. This optimization not only ensures streamlined gas operations but also contributes to substantial cost savings during contract execution.

Status: Acknowledged.

Automatic Testing

1- SOLIDITY STATIC ANALYSIS



SOLIDITY STATIC ANALYSIS

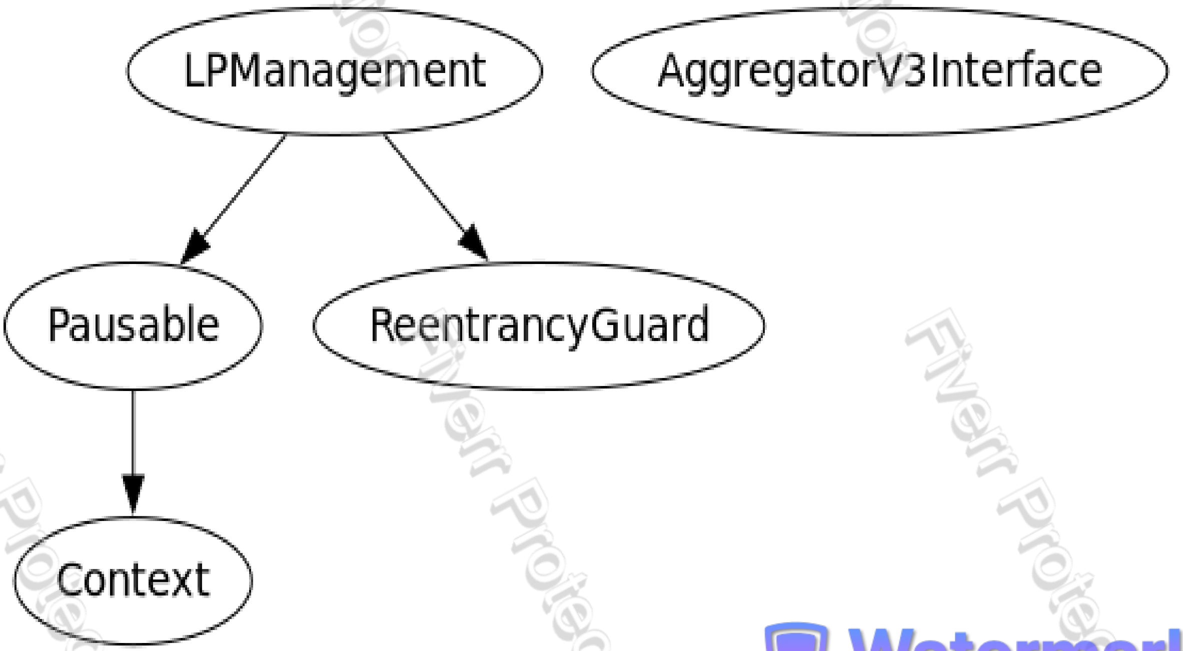
☑ Select all ☑ Autorun Run

- 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

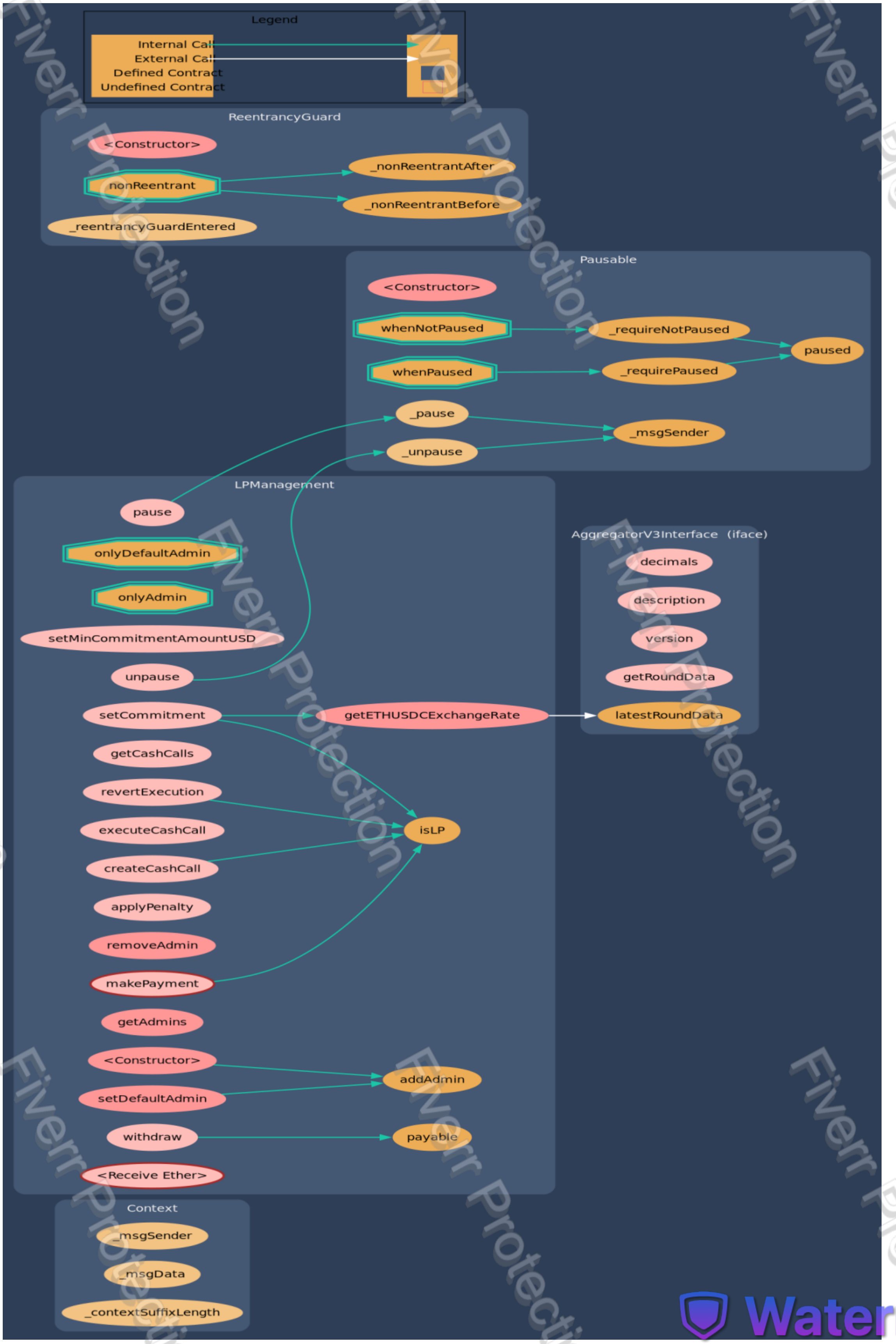
SOLIDITY STATIC ANALYSIS

- 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

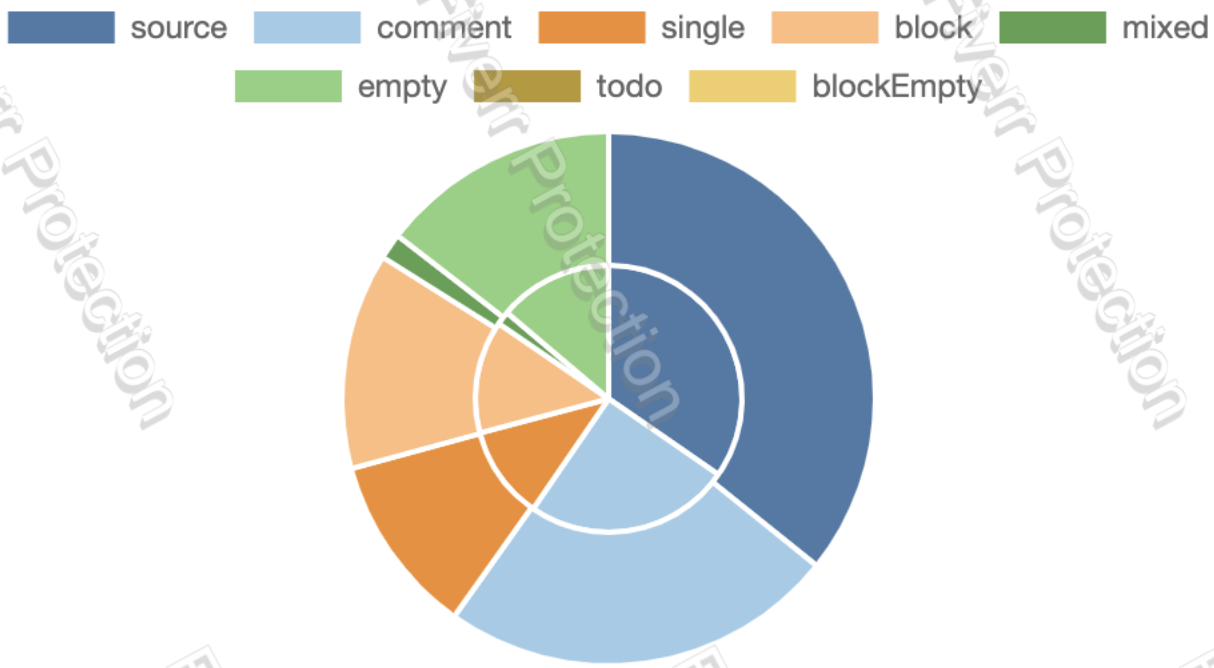
2- Inheritance graph



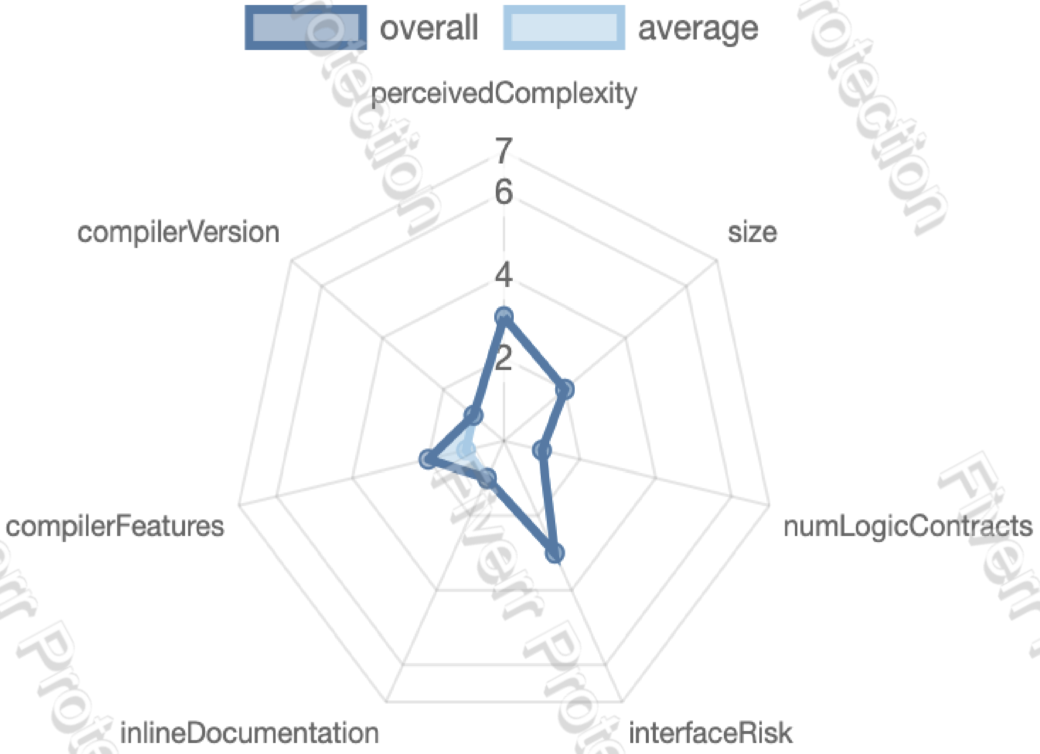
3- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
ReentrancyGuard
<Constructor>
nonReentrant
_nonReentrantAfter
_nonReentrantBefore
_reentrancyGuardEntered
Pausable
<Constructor>
whenNotPaused
_requireNotPaused
paused
whenPaused
_requirePaused
_pause
_msgSender
_unpause
LPManagement
pause
onlyDefaultAdmin
onlyAdmin
setMinCommitmentAmountUSD
unpause
setCommitment
getETHUSDCExchangeRate
getCashCalls
revertExecution
isLP
executeCashCall
createCashCall
applyPenalty
removeAdmin
makePayment
getAdmins
<Constructor>
addAdmin
setDefaultAdmin
withdraw
payable
<Receive Ether>
AggregatorV3Interface (iface)
decimals
description
version
getRoundData
latestRoundData
Context
_msgSender
_msgData
_contextSuffixLength

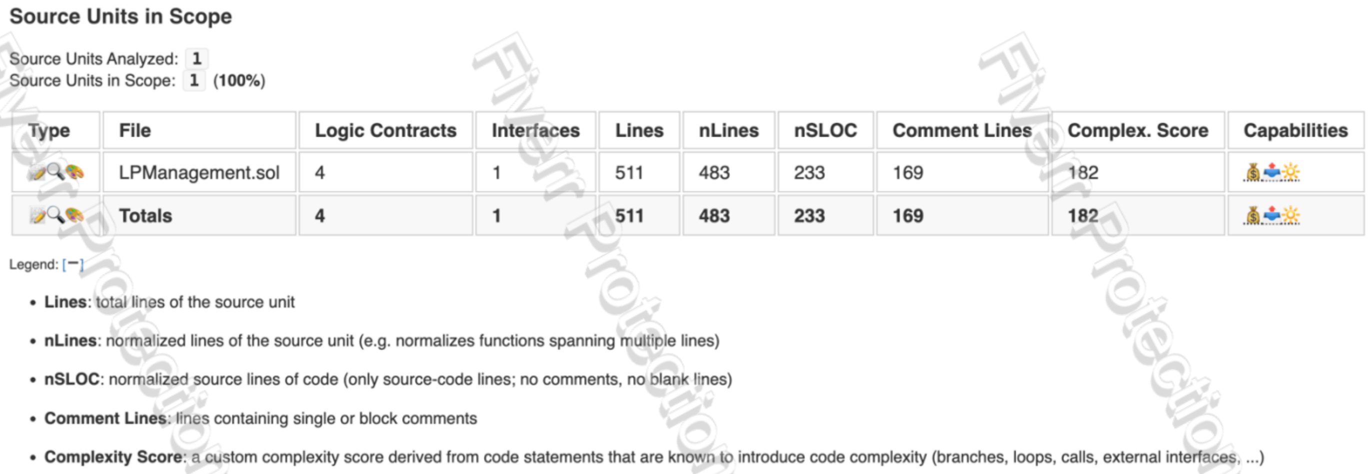
Source lines



Risk level



Source units in scope



Source Units in Scope

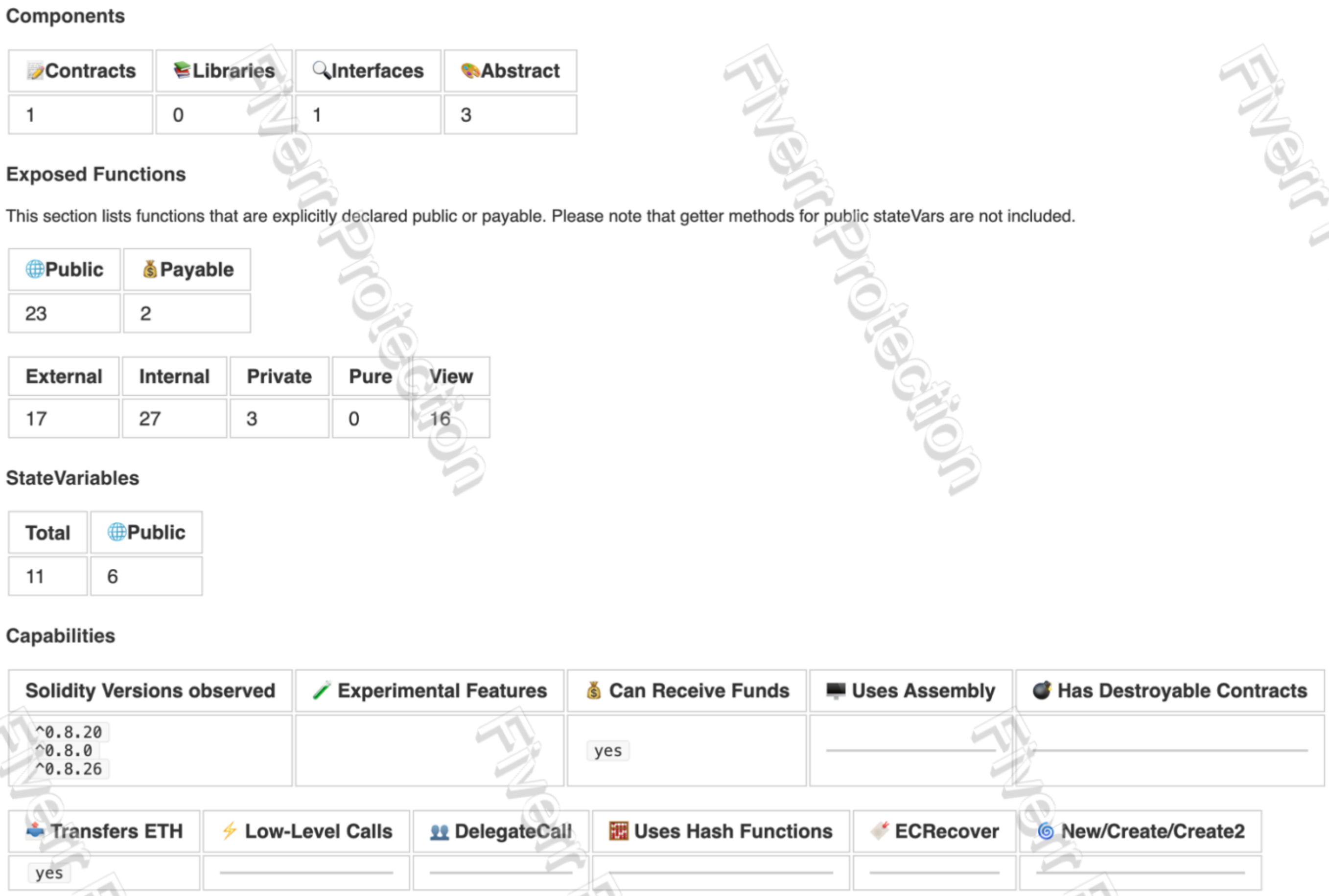
Source Units Analyzed: 1
Source Units in Scope: 1 (100%)

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝🔍🎨	LPManagement.sol	4	1	511	483	233	169	182	💰📤☀
📝🔍🎨	**Totals**	**4**	**1**	**511**	**483**	**233**	**169**	**182**	💰📤☀

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	3

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
23	2

External	Internal	Private	Pure	View
17	27	3	0	16

StateVariables

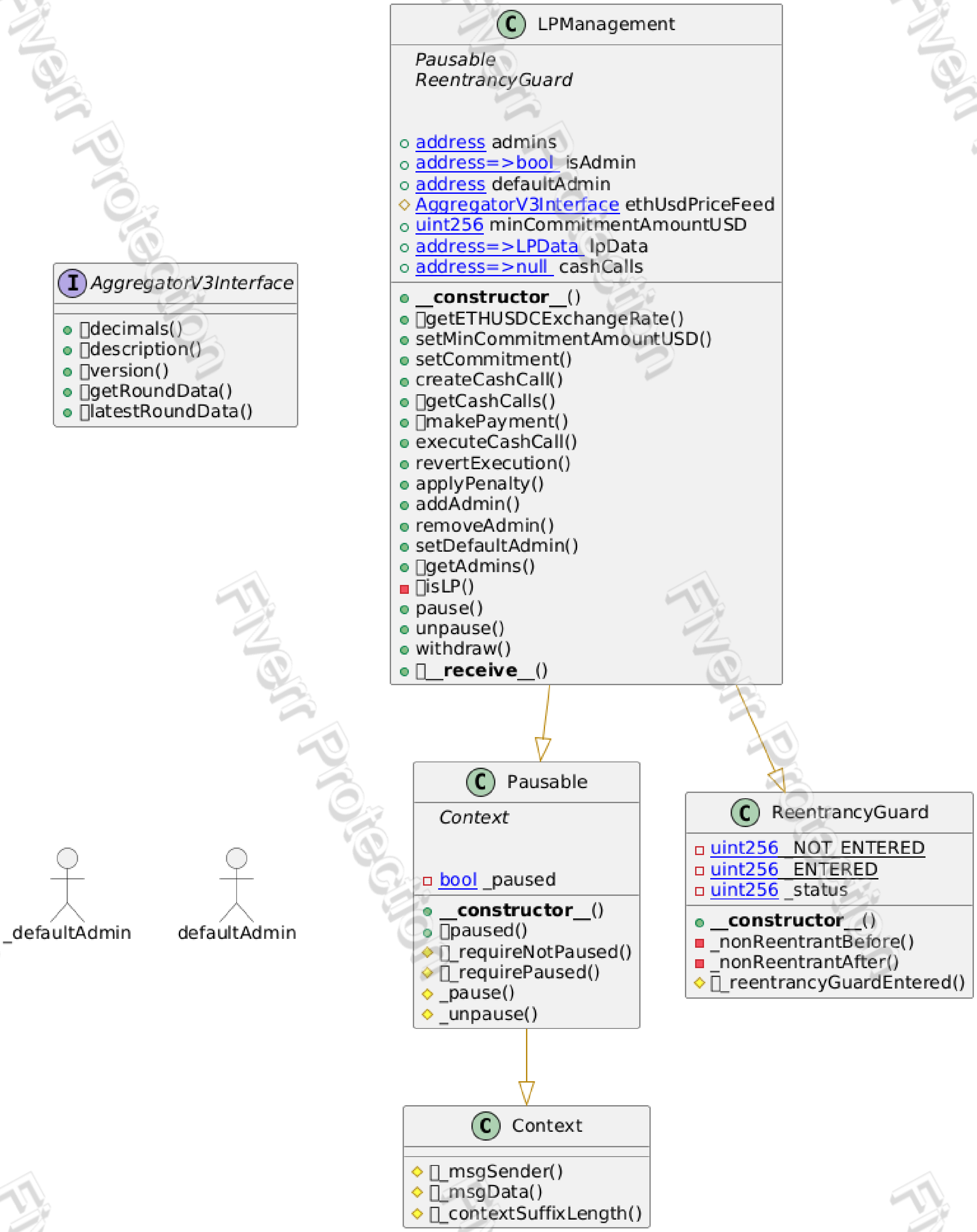
Total	🌐Public
11	6

Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
^0.8.20 ^0.8.0 ^0.8.26		yes		

📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	🔖 ECRecover	🌀 New/Create/Create2
yes					

Unified Modeling Language (UML)



Functions signature

Function Name	Sighash	Function Signature
paused	5c975abb	paused()
decimals	313ce567	decimals()
description	7284e416	description()
version	54fd4d50	version()
getRoundData	9a6fc8f5	getRoundData(uint80)
latestRoundData	feaf968c	latestRoundData()
getETHUSDCExchangeRate	43bc5cd3	getETHUSDCExchangeRate()
setMinCommitmentAmountUSD	161b1349	setMinCommitmentAmountUSD(uint256)
setCommitment	8dd64b00	setCommitment(address,uint256,uint256)
createCashCall	87d4377c	createCashCall(address,uint256,uint256)
getCashCalls	8c28a23a	getCashCalls(address)
makePayment	f98cf07c	makePayment(address,uint256)
executeCashCall	f92a5dcd	executeCashCall(address,uint256)
revertExecution	b9721b2e	revertExecution(address,uint256)
applyPenalty	2ebf8bd6	applyPenalty(address,uint256,bool)
addAdmin	70480275	addAdmin(address)
removeAdmin	1785f53c	removeAdmin(address)
setDefaultAdmin	15a41150	setDefaultAdmin(address)
getAdmins	31ae450b	getAdmins()
pause	8456cb59	pause()
unpause	3f4ba83a	unpause()
withdraw	00f714ce	withdraw(uint256,address)

Automatic general report

```
Files Description Table

|  File Name  |  SHA-1 Hash  |
|-------------|--------------|
| /Users/macbook/Desktop/smart contracts/LPManagement.sol |
5e5b4e936069ff3e2402ade058d921ca9294b3be |

 Contracts Description Table

|  Contract  |        Type        |       Bases      |
|                  |
|:----------:|:-------------------:|:----------------:|:----------------
:|:----------------:|
|     └      |  **Function Name**  |  **Visibility**  |  **Mutability**
|  **Modifiers**  |
||||||
| **Context** | Implementation |  |||
| └ | _msgSender | Internal 🔒 |   | |
| └ | _msgData | Internal 🔒 |   | |
| └ | _contextSuffixLength | Internal 🔒 |   | |
||||||
| **Pausable** | Implementation | Context |||
| └ | <Constructor> | Public ❗️ | 🛑  |NO❗️ |
| └ | paused | Public ❗️ |   |NO❗️ |
| └ | _requireNotPaused | Internal 🔒 |   | |
| └ | _requirePaused | Internal 🔒 |   | |
| └ | _pause | Internal 🔒 | 🛑  | whenNotPaused |
| └ | _unpause | Internal 🔒 | 🛑  | whenPaused |
||||||
| **ReentrancyGuard** | Implementation |  |||
| └ | <Constructor> | Public ❗️ | 🛑  |NO❗️ |
| └ | _nonReentrantBefore | Private 🔐 | 🛑  | |
| └ | _nonReentrantAfter | Private 🔐 | 🛑  | |
| └ | _reentrancyGuardEntered | Internal 🔒 |   | |
||||||
| **AggregatorV3Interface** | Interface |  |||
| └ | decimals | External ❗️ |   |NO❗️ |
| └ | description | External ❗️ |   |NO❗️ |
| └ | version | External ❗️ |   |NO❗️ |
| └ | getRoundData | External ❗️ |   |NO❗️ |
| └ | latestRoundData | External ❗️ |   |NO❗️ |
||||||
| **LPManagement** | Implementation | Pausable, ReentrancyGuard |||
| └ | <Constructor> | Public ❗️ | 🛑  |NO❗️ |
| └ | getETHUSDCExchangeRate | Public ❗️ |   |NO❗️ |
| └ | setMinCommitmentAmountUSD | External ❗️ | 🛑  | onlyDefaultAdmin |
| └ | setCommitment | External ❗️ | 🛑  | whenNotPaused onlyAdmin |
| └ | createCashCall | External ❗️ | 🛑  | whenNotPaused onlyAdmin |
| └ | getCashCalls | External ❗️ |   |NO❗️ |
| └ | makePayment | External ❗ |  💵 | whenNotPaused nonReentrant |
| └ | executeCashCall | External ❗ | 🛑  | whenNotPaused onlyAdmin |
| └ | revertExecution | External ❗ | 🛑  | whenNotPaused onlyAdmin |
| └ | applyPenalty | External ❗ | 🛑  | whenNotPaused onlyAdmin |
| └ | addAdmin | Public ❗ | 🛑  | onlyDefaultAdmin |
| └ | removeAdmin | Public ❗ | 🛑  | onlyDefaultAdmin |
| └ | setDefaultAdmin | Public ❗ | 🛑  | onlyDefaultAdmin |
| └ | getAdmins | Public ❗ |   |NO❗ |
| └ | isLP | Private 🔐 |   | |
| └ | pause | External ❗ | 🛑  | onlyAdmin |
| └ | unpause | External ❗ | 🛑  | onlyAdmin |
| └ | withdraw | External ❗ | 🛑  | onlyAdmin nonReentrant |
| └ | <Receive Ether> | External ❗ |  💵 | whenNotPaused |

 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.