

Smart Contract Security Audit V1

Floof Fi Smart Contract Audit

Feb 24, 2025



<https://saferico.com/>

business@saferico.com

https://t.me/SFI_ANN

—

Table of Contents

Table of Contents

Background

Project Information

Token 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)

Functions signature

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:** Binance Smart Chain
- **Name:** Floof Fi
- **Language :** Solidity
- **Contract Address:** 0xa2e09E20eb3Ffa3339c168169c18c4509C45d566
- **Code Source:** <https://bscscan.com/address/0xa2e09E20eb3Ffa3339c168169c18c4509C45d566#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:

FloofFi.sol

Audit Score:

99% secure



File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
FloofFi.sol	a4e97dce610a24e8e6e99adde95ecc09dd8edb0a	0xa2e09E20eb3Ffa3339c168169c18c4509C45d566

- Contract: FloofFi
- Inherit: ERC20, Ownable, ERC20Burnable
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
balanceOf	✓	Read / public	Passed
allowance	✓	Read / public	Passed
isExcludedFromTax	✓	Read / public	Passed
decimal	✓	Read / public	Passed
totalSupply	✓	Read / public	Passed
name	✓	Read / public	Passed
owner	✓	Read / public	Passed
symbol	✓	Read / public	Passed
transferOwnership	✓	Write / public	Passed
renounceOwnership	✓	Write / public	Passed
burn	✓	Write / public	Passed
burnFrom	✓	Write / public	Passed
addToExcludedList	✓	Write / public	Passed
transfer	✓	Write / public	Passed

approve	✓	Write / public	Passed
transferFrom	✓	Write / public	Passed
addUniswapV2PairAddress	✓	Write / public	Passed
changeTaxes	✓	Write / public	Passed
clearBNB	✓	Write / public	Passed
airdrop	✓	Write / public	Passed
airdropwithSinglevalue	✓	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	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:

#Owner privileges (In the period when the owner isn't renounced)

Description

The owner can change the taxes.

The owner can burn the tokens.

The owner can Exclude any address from Tax.

```
function changeTaxes(uint8 _sellTax, uint8 _buyTax) public onlyOwner {
    require(_sellTax <= 10 && _sellTax > 0, "tax must be in range 1% - 25% !");
    require(_buyTax <= 10 && _buyTax > 0, "tax must be in range 1%- 25%!");
    sellFeePercentage = _sellTax;
    buyFeePercentage = _buyTax;
    emit TaxChange(_buyTax, _sellTax) }
function addToExcludedList(address _address) external onlyOwner{
    isExcludedFromTax[_address] = true; }
```

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](#).

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

No Notes were found.

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

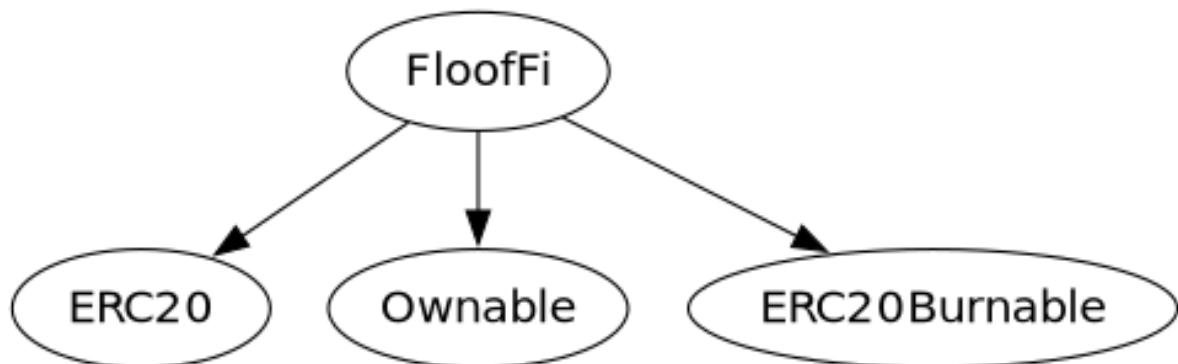
- ☒ **ERC20:**
'decimals' should be 'uint8'

Miscellaneous

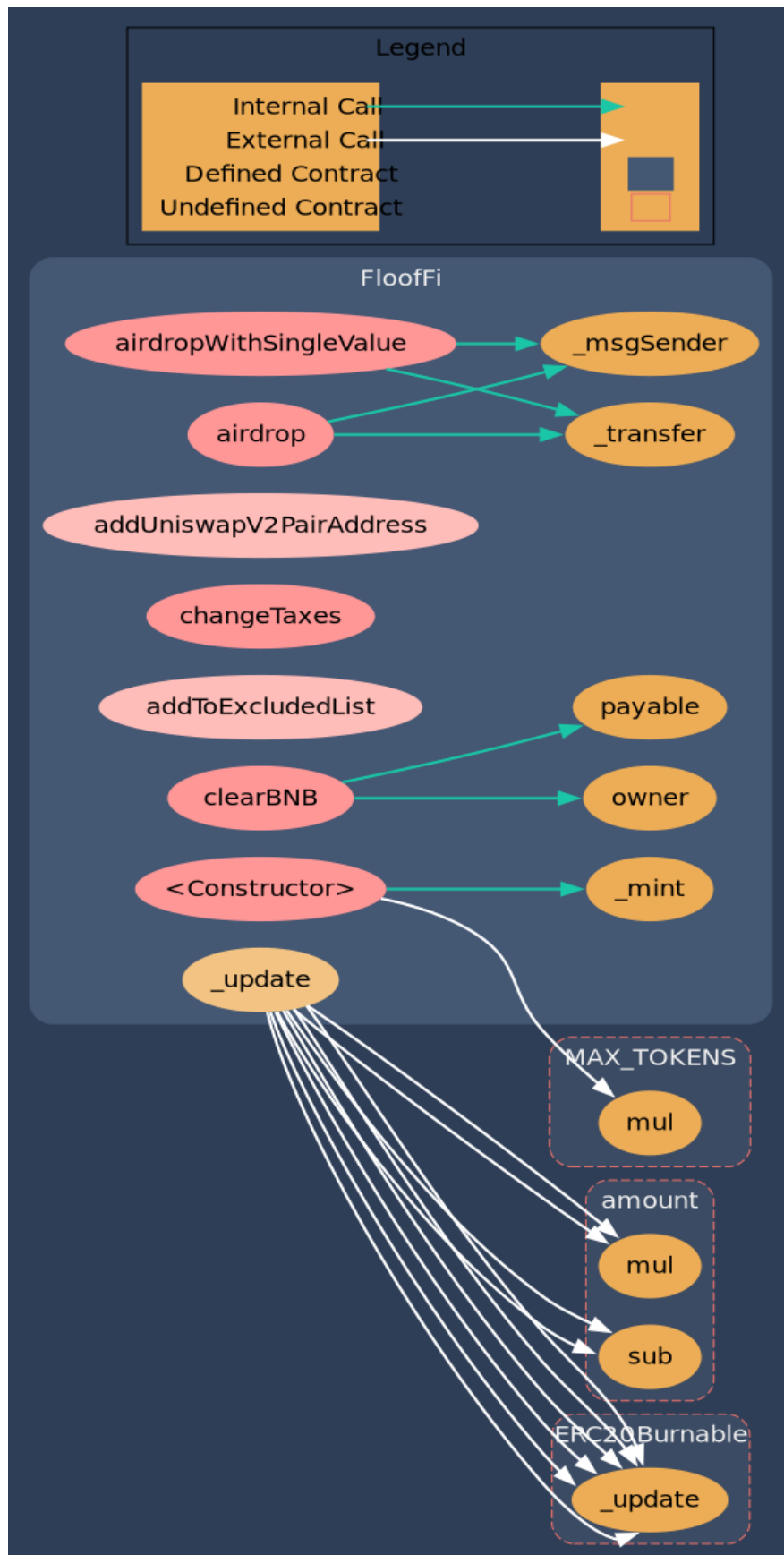
☒ Select Miscellaneous

- ☒ **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

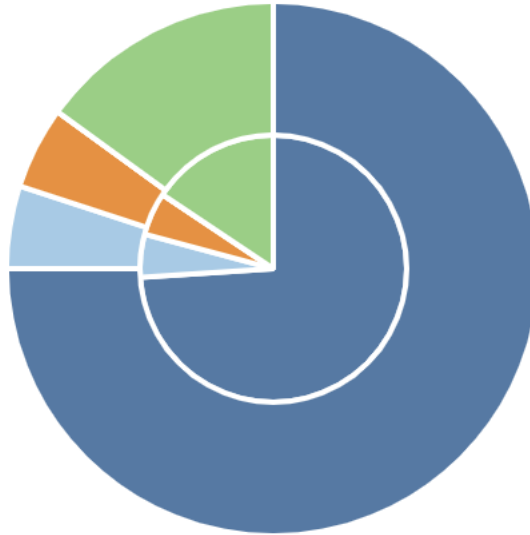
2- Inheritance graph



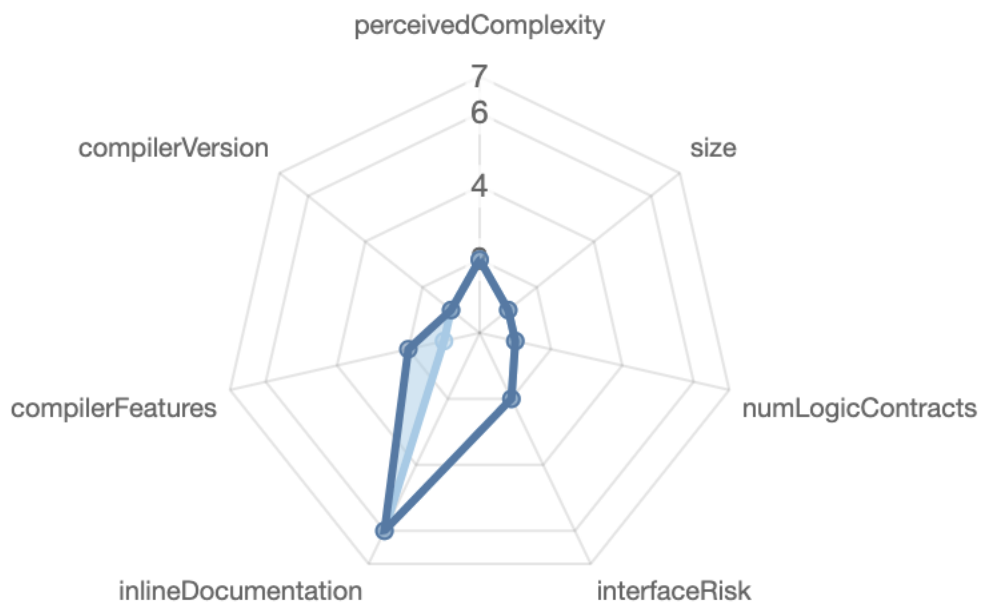
3- Call graph



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
	Floof Fi.sol	1	_____	95	91	71	5	74	
	Totals	1	_____	95	91	71	5	74	

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	0	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
6	0

External	Internal	Private	Pure	View
2	9	0	0	0

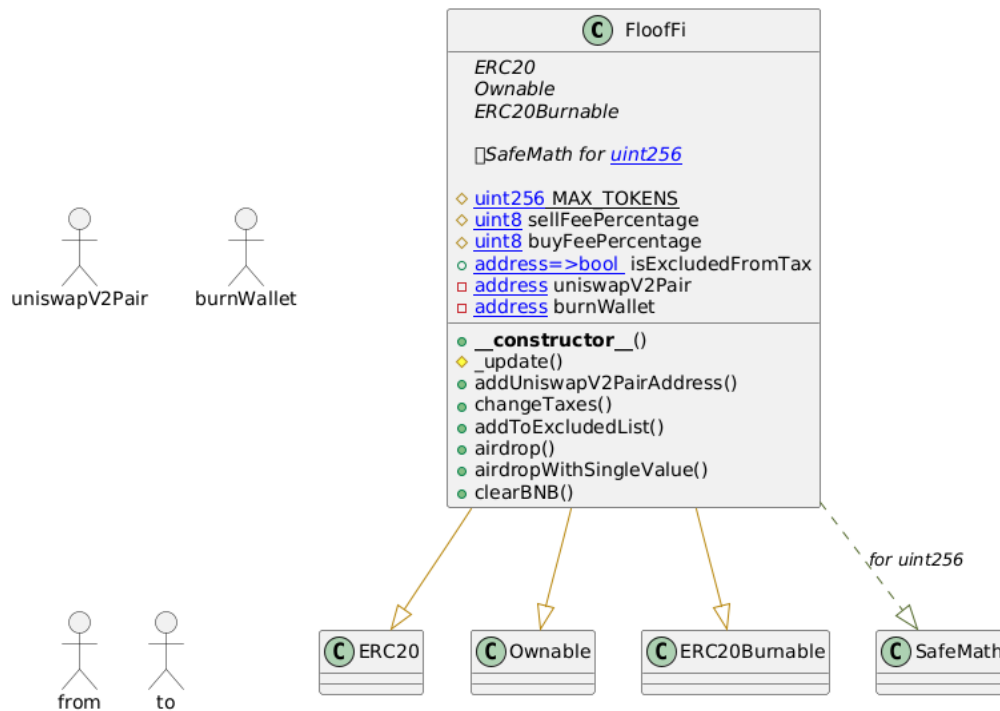
StateVariables

Total	 Public
6	1

Capabilities

Solidity Versions observed	🔧 Experimental Features	💰 Can Receive Funds	💻 Uses Assembly	💣 Has Destroyable Contracts	
0.8.28					
👉 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
addUniswapV2PairAddress	3fb7c67e	addUniswapV2PairAddress(address)
changeTaxes	9de7ed73	changeTaxes(uint8,uint8)
addToExcludedList	e164836d	addToExcludedList(address)
airdrop	67243482	airdrop(address[],uint256[])
airdropWithSingleValue	379d72bb	airdropWithSingleValue(address[],uint256)
clearBNB	f6c6dadf	clearBNB()

Automatic general report



Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/Floof Fi.sol	a4e97dce610a24e8e6e99adde95ecc09dd8edb0a

Contracts Description Table

Contract	Type	Bases	
:-----:-----:-----:-----			
L	**Function Name**	**Visibility**	**Mutability**
Modifiers			
FloofFi	Implementation	ERC20, Ownable, ERC20Burnable	
L	<Constructor>	Public !	Ownable ERC20
L	_update	Internal	
L	addUniswapV2PairAddress	External !	onlyOwner
L	changeTaxes	Public !	onlyOwner
L	addToExcludedList	External !	onlyOwner
L	airdrop	Public !	NO !
L	airdropWithSingleValue	Public !	NO !
L	clearBNB	Public !	onlyOwner

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.