Smart Contract Security Audit V1

S39 Token Smart Contract Audit

Aug 5, 2023

https://saferico.com/

business@saferico.com
https://t.me/SFI_ANN

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 and Token Information

- **Platform**: Binance Smart Chain
- **Website**: https://s39token.com/
- **Name**: S39 Token
- **Language** : solidity
- **Contract Address**: 0x722327604bE7CF1B3d9B111a87605c56512112c3
- **Code Source**: https://bscscan.com/address/0x722327604bE7CF1B3d9B111a87605c56512112c3#code

Executive Summary

According to our assessment, the customer`s solidity smart contract is **Insecured**.

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, 1 high, 0 medium, 2 low, 0 very low-level issues and 1 note in all solidity files of the contract

The files:

S39Token.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
S39Token.sol	cad78dc8860c256e774ce23f598dc9ea05f8dd4 4	0x722327604bE7CF1B3d9B111a87605c56512112c3

- Contract: S39Token
- Inherit: Context, IBEP20, Ownable
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
name	✓	Read / public	**Passed**
symbol	✓	Read / public	**Passed**
decimals	✓	Read / public	**Passed**
totalSupply	✓	Read / public	**Passed**
allowance	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
decimals	✓	Read / public	**Passed**
getOwner	✓	Read / public	**Passed**
totalSupply	✓	Read / public	**Passed**
approve	✓	Write / public	**Passed**
mint	✓	Write / public	**Passed**
transferFrom	✓	Write / public	**Passed**
transfer	✓	Write / public	**Passed**
transferOwnership	✓	Write / public	**Passed**

decreaseAllowance	✓	Write / public	**Passed**
increaseAllowance	✓	Write / public	**Passed**

Issues Checking Status

No.	Issue Description	Checking Status
1	Compiler warnings.	**Passed**
2	Race conditions and Reentrancy. Cross-function race conditions.	**Passed**
3	Possible delays in data delivery.	**Passed**
4	Oracle calls.	**Passed**
5	Design Logic.	**Passed**
6	Timestamp dependence.	**Passed**
7	Integer Overflow and Underflow.	**Passed**
8	DoS with Revert.	**Passed**
9	DoS with block gas limit.	**Passed with notes**
10	Methods execution permissions.	**Passed**
11	Economy model. If application logic is based on an incorrect economic model, the application would not function correctly and participants would incur financial losses. This type of issue is most often found in bonus rewards systems, Staking and Farming contracts, Vault and Vesting contracts, etc.	**Passed**
12	The impact of the exchange rate on the logic.	**Passed**
13	Private user data leaks.	**Passed**
14	Malicious Event log.	**Passed**
15	Scoping and Declarations.	**Passed**
16	Uninitialized storage pointers.	**Passed**
17	Arithmetic accuracy.	**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:

the owner can mint new tokens

SaferICO tests if the owner of the smart contract can mint new tokens. If the contract contains a mint function, we refer to the token's total supply as non-fixed, allowing the token owner to "mint" more tokens whenever they want.
A mint function in the smart contract allows minting tokens at a later stage. A method to disable minting can also be added to stop the minting process irreversibly.
Minting tokens is done by sending a transaction that creates new tokens inside of the token smart contract. With the help of the smart contract function, an unlimited number of tokens can be created without spending additional energy or money.

```
function mint(uint256 amount) public onlyOwner returns (bool) {
    _mint(_msgSender(), amount);
    return true;
  }
function _mint(address account, uint256 amount) internal {
    require(account != address(0), "BEP20: mint to the zero
address");

    _totalSupply = _totalSupply.add(amount);
    _balances[account] = _balances[account].add(amount);
    emit Transfer(address(0), account, amount);
  }
```

Recommendation

Avoid use of mint function.

Status

Opened.

Medium:

No Medium severity vulnerabilities were found.

Low:

Approve Race

The standard BEP20 implementation contains a widely-known racing condition in it approve function, where in a spender is able to witness the token owner broadcast a transaction altering their approval and quickly sign and broadcast a transaction using transferFrom to move the current approved amount from the owner's balance to the spender. If the spender's transaction is validated before the owner's, the spender will be able to get both approval amounts of both transactions.

```
function approve(address spender, uint256 amount) external returns
(bool) {
    _approve(_msgSender(), spender, amount);
    return true;
  }
```

Recommendation

Use increaseAllowance and decreaseAllowance functions to modify the approval amount instead of using the approve function to modify it.

Status

Opened.

#Pragam version not fixed

Description

It is a good practice to lock the solidity version for a live deployment (use 0.8.20 instead of ^0.5.6). 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.

Remediation

Remove the ^ sign to lock the pragma version.

Status: Acknowledged.

Very Low:

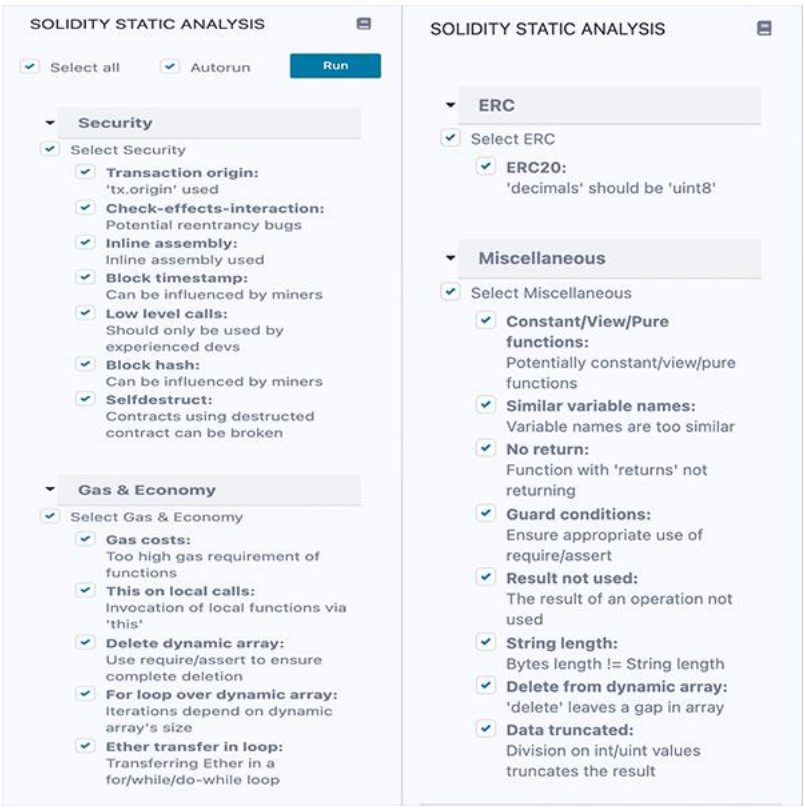
No Very Low severity vulnerabilities were found.

Notes:

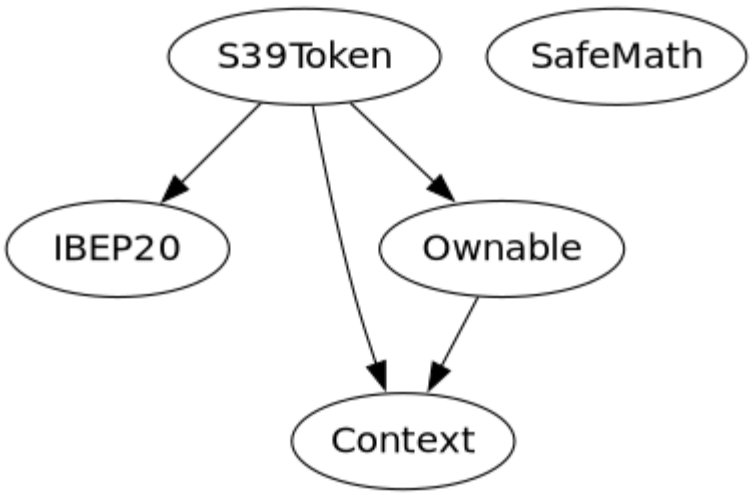
No Notes were found.

Automatic Testing

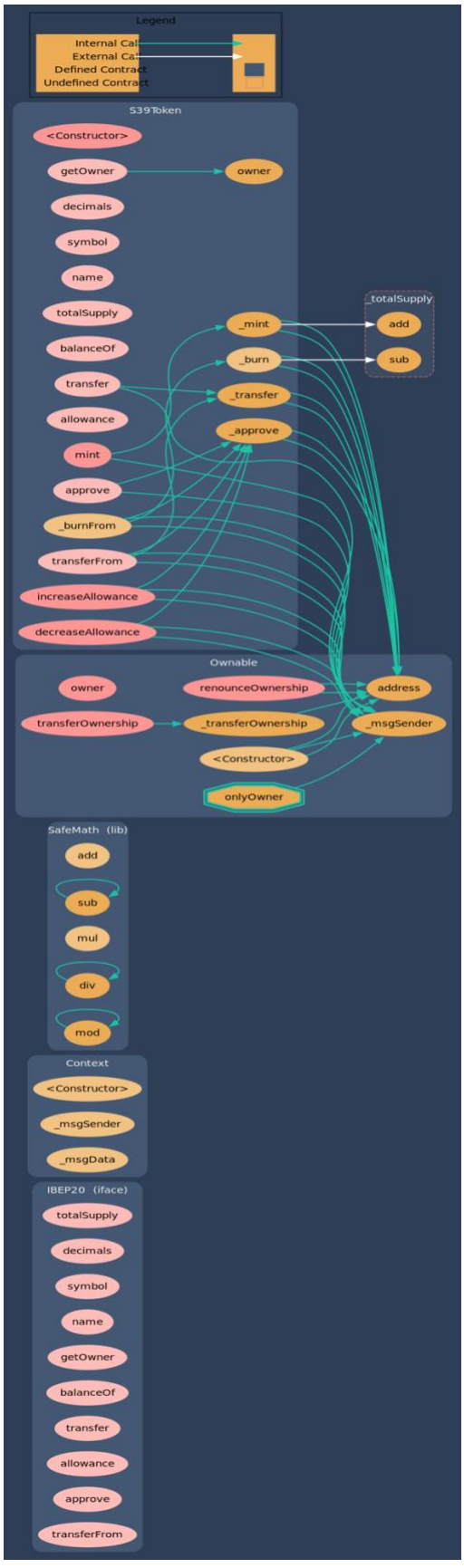
1- SOLIDITY STATIC ANALYSIS



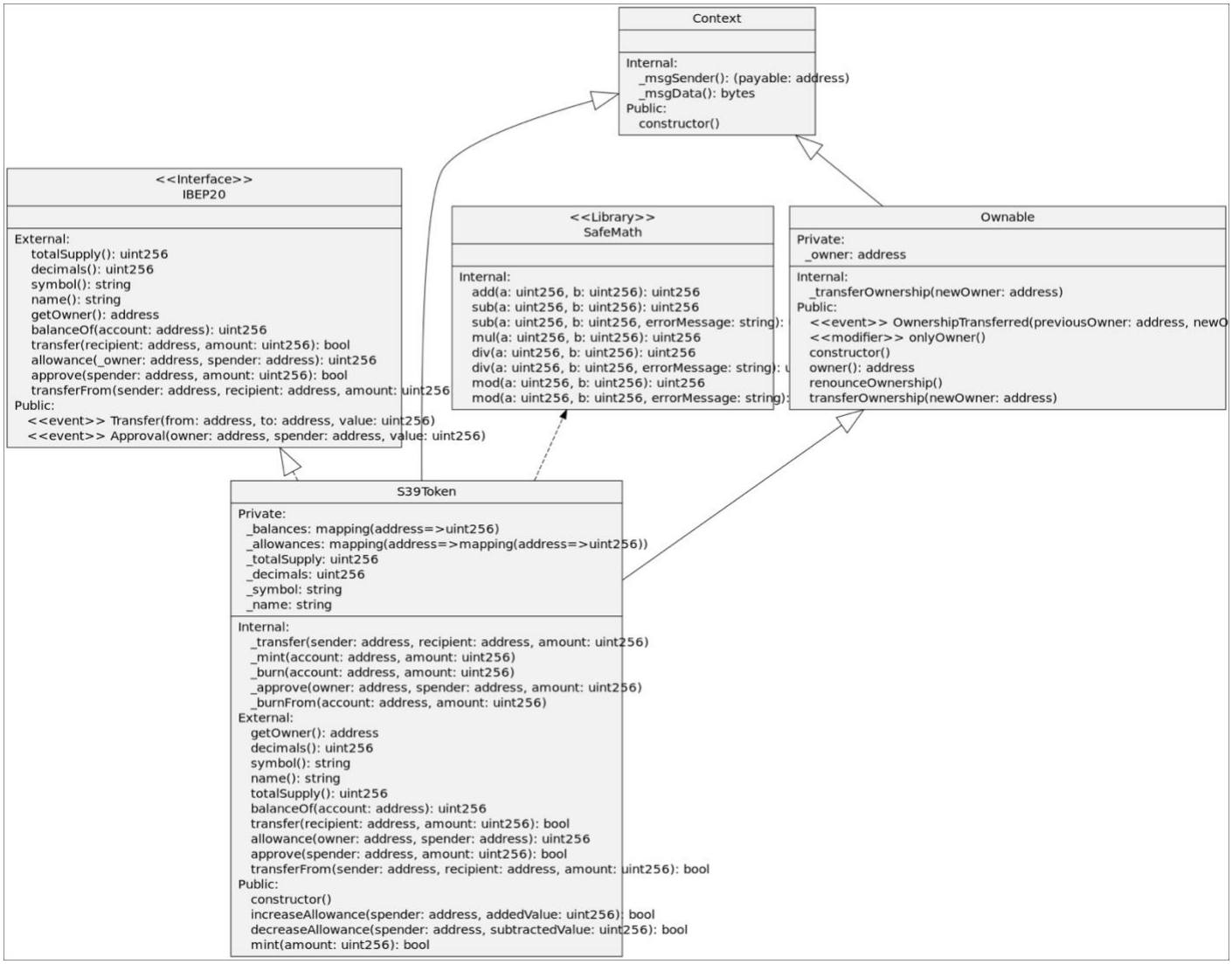
2- Inheritance graph



3- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
S39Token
<Constructor>
getOwner
owner
decimals
symbol
name
totalSupply
balanceOf
transfer
allowance
mint
approve
_burnFrom
transferFrom
increaseAllowance
decreaseAllowance
_mint
_burn
_transfer
_approve
_totalSupply
add
sub
Ownable
owner
renounceOwnership
address
transferOwnership
_transferOwnership
_msgSender
<Constructor>
onlyOwner
SafeMath (lib)
add
sub
mul
div
mod
Context
<Constructor>
_msgSender
_msgData
IBEP20 (iface)
totalSupply
decimals
symbol
name
getOwner
balanceOf
transfer
allowance
approve
transferFrom

Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
39509351  =>  increaseAllowance(address,uint256)
18160ddd  =>  totalSupply()
313ce567  =>  decimals()
95d89b41  =>  symbol()
06fdde03  =>  name()
893d20e8  =>  getOwner()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
e31bdc0a  =>  sub(uint256,uint256,string)
c8a4ac9c  =>  mul(uint256,uint256)
a391c15b  =>  div(uint256,uint256)
b745d336  =>  div(uint256,uint256,string)
f43f523a  =>  mod(uint256,uint256)
71af23e8  =>  mod(uint256,uint256,string)
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
d29d44ee  =>  _transferOwnership(address)
a457c2d7  =>  decreaseAllowance(address,uint256)
a0712d68  =>  mint(uint256)
30e0789e  =>  _transfer(address,address,uint256)
4e6ec247  =>  _mint(address,uint256)
6161eb18  =>  _burn(address,uint256)
104e81ff  =>  _approve(address,address,uint256)
a22b35ce  =>  _burnFrom(address,uint256)
```

Automatic general report

```
Files Description Table
|  File Name  |  SHA-1 Hash  |
|-------------|--------------|
| /Users/macbook/Desktop/smart contracts/S39 Token.sol |
cad78dc8860c256e774ce23f598dc9ea05f8dd44 |
 Contracts Description Table
|  Contract  |         Type        |       Bases      |                  |
|
|:----------:|:-------------------:|:----------------:|:----------------:|:-------
--------:|
|     └      |  **Function Name**  |  **Visibility**  |  **Mutability**  |
**Modifiers**  |
||||||
| **IBEP20** | Interface |  |||
| └ | totalSupply | External ❗ |  |NO❗ |
| └ | decimals | External ❗ |  |NO❗ |
| └ | symbol | External ❗ |  |NO❗ |
| └ | name | External ❗ |  |NO❗ |
| └ | getOwner | External ❗ |  |NO❗ |
| └ | balanceOf | External ❗ |  |NO❗ |
| └ | transfer | External ❗ | 🛑 |NO❗ |
| └ | allowance | External ❗ |  |NO❗ |
| └ | approve | External ❗ | 🛑 |NO❗ |
| └ | transferFrom | External ❗ | 🛑 |NO❗ |
||||||
| **Context** | Implementation |  |||
| └ | <Constructor> | Internal 🔒 | 🛑 | |
| └ | _msgSender | Internal 🔒 |  | |
| └ | _msgData | Internal 🔒 |  | |
||||||
| **SafeMath** | Library |  |||
| └ | add | Internal 🔒 |  | |
| └ | sub | Internal 🔒 |  | |
| └ | sub | Internal 🔒 |  | |
| └ | mul | Internal 🔒 |  | |
| └ | div | Internal 🔒 |  | |
| └ | div | Internal 🔒 |  | |
| └ | mod | Internal 🔒 |  | |
| └ | mod | Internal 🔒 |  | |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Internal 🔒 | 🛑 | |
| └ | owner | Public ❗ |  |NO❗ |
| └ | renounceOwnership | Public ❗ | 🛑 | onlyOwner |
```

└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑	
S39Token	Implementation	Context, IBEP20, Ownable		
└	<Constructor>	Public ❗	🛑	NO❗
└	getOwner	External ❗		NO❗
└	decimals	External ❗		NO❗
└	symbol	External ❗		NO❗
└	name	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
└	increaseAllowance	Public ❗	🛑	NO❗
└	decreaseAllowance	Public ❗	🛑	NO❗
└	mint	Public ❗	🛑	onlyOwner
└	_transfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_burnFrom	Internal 🔒	🛑	

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Conclusion

The contracts arenot written systematically. Team found high issues. So, it isn’t 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 “In Secured”.

✓ volatile code.
✓ 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.